Game Theoretical Analysis of Blockchain Users' Behaviors

Marianna Belotti

École Doctorale Sciences des Métiers de l'Ingénieur
Laboratoire CEDRIC
Centre d'études et de recherche en informatique et communications

THÉSE

présentée par : **Marianna Belotti**
soutenue le : 16 décembre 2021

pour obtenir le grade de : **Docteur d'HESAM Université**

préparée au : **Conservatoire National des Arts et Métiers**

Discipline : **Informatique**

Spécialité : **Mathématiques appliquées et applications des mathématiques**

Game Theoretical Analysis of Blockchain Users' Behaviors

Thése dirigée par :
Stefano Secci, Professeur, Cnam

Maria Potop-Butucaru, Professeur, Sorbonne Université

et co-dirigée par :
Stefano Moretti, Directeur de recherche, CNRS, Paris Dauphine University PSL

JURY

Nicolas Maudet	Professeur, Sorbonne Université, France	Président
Maurice Herlihy	Professeur, Brown University, USA	Rapporteur
Emmanuelle Anceaume	Directrice de recherche, IRISA, France	Rapporteuse
Julien Prat	Professeur, ENSAE, France	Examinateur
Silvia Bonomi	Professeur, La Sapienza University, Italie	Examinatrice
Henning Ahnert	Banque Centrale Européenne, Allemagne	Invité
Nadia Filali	Caisse des Dépôts, France	Invitée

T H È S E

Affidavit

Je soussignée, Marianna Belotti, déclare que le travail présenté dans ce manuscrit est mon propre travail, réalisé sous la direction scientifique de M. Stefano Secci (directeur de thèse), de Mme Maria Potop-Butucaru (co-directrice de thèse) et de M. Stefano Moretti (co-encadrant), conformément aux principes d'honnêteté, d'intégrité et de responsabilité inhérents à la mission de recherche. Les travaux de recherche et la rédaction de ce manuscrit ont été réalisés dans le respect de la charte française d'intégrité de la recherche. Ce travail n'a pas été soumis antérieurement, en France ou à l'étranger, dans la même version ou dans une version similaire, à un autre organisme d'examen.

Paris, 16 décembre 2021

Marianna Belotti

Affidavit

I, undersigned, Marianna Belotti, hereby declare that the work presented in this manuscript is my own work, carried out under the scientific direction of Mr Stefano Secci (thesis director), Mrs Maria Potop-Butucaru (co-thesis director) and of Mr Stefano Moretti (supervisor), in accordance with the principles of honesty, integrity and responsibility inherent to the research mission. The research work and the writing of this manuscript have been carried out in compliance with the French charter for Research Integrity. This work has not been submitted previously either in France or abroad in the same or in a similar version to any other examination body.

Paris, December 16th 2021

Marianna Belotti

Vorrei dedicare questa tesi ai miei desideri, sogni e progetti
che possono cominciare ora a realizzarsi.

I would like to dedicate this thesis to my wishes, dreams and projects
that can now start to come true.

Je souhaiterais dédier cette thèse à mes souhaits, mes rêves et mes projets
qui peuvent maintenant commencer à se réaliser.

Abstract

The disruptive technology born in 2008 with Bitcoin and known as blockchain represents a significant quality leap from the distributed database technology. Distributed systems theory provides then models and techniques to analyze some protocols characterizing the technology, however in order to analyze a blockchain system additional considerations on its users need to be done. This thesis aims at analyzing the different behaviors of the users operating in blockchains or more in general in DLTs (i.e., Distributed Ledger Technologies). The latter are considered as rational agents, fully aware of all actions available to them and capable of choosing the one they feel is the best for themselves. Game theory is then used to model situations where users are called to choose and perform certain actions within the DLT environment. This thesis analyzes different users as well as different blockchains with the scope of providing a general overview on the topic and formal results on their behaviors ; users may indeed be honest *vis-à-vis* of other users or they may behave maliciously (as Byzantine nodes) attacking the blockchain system.

Keywords : Blockchain, Consensus, Game Theory, Crypto-assets.

Résumé

La technologie disruptive née en 2008 avec Bitcoin et connue sous le nom de blockchain représente un saut qualitatif important par rapport à la technologie des bases de données distribuées. La théorie des systèmes distribués fournit alors des modèles et des techniques pour analyser certains protocoles caractérisant la technologie, cependant afin d'analyser un système blockchain des considérations supplémentaires sur ses utilisateurs doivent être faites. Cette thèse vise à analyser les différents comportements des utilisateurs opérant dans les blockchains ou plus largement dans les DLTs (i.e., Distributed Ledger Technologies). Ces derniers sont considérés comme des agents rationnels, pleinement conscients de toutes les actions à leur disposition et capables de choisir celle qui leur semble la meilleure pour eux-mêmes. La théorie des jeux est alors utilisée pour modéliser des situations où les utilisateurs sont appelés à choisir et à effectuer certaines actions dans l'environnement DLT. Cette thèse analyse différents utilisateurs ainsi que différentes blockchains dans le but de fournir une vue d'ensemble sur le sujet et des résultats formels sur leurs comportements ; les utilisateurs peuvent en effet être honnêtes vis-à-vis des autres utilisateurs ou se comporter de manière malveillante (comme des nœuds byzantins) en attaquant le système blockchain.

Mots-clés : Blockchain, Consensus, Théorie des Jeux, Crypto-actifs

Table of contents

List of Tables

List of Figures

Publications

Marianna Belotti, Nikola Božić, Guy Pujolle, and Stefano Secci. A vademecum on blockchain technologies : When, which, and how. *IEEE Communications Surveys & Tutorials*, 21(4) :3796–3838, 2019.

Marianna Belotti, Sofiane Kirati, and Stefano Secci. Bitcoin pool-hopping detection. In *2018 IEEE 4th International Forum on Research and Technology for Society and Industry (RTSI)*, pages 1–6. IEEE, 2018.

Marianna Belotti, Stefano Moretti, Maria Potop-Butucaru, and Stefano Secci. Game theoretical analysis of atomic cross-chain swaps. In *40th IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2020.

Marianna Belotti, Stefano Moretti, and Paolo Zappalà. Rewarding miners : bankruptcy situations and pooling strategies. In *Multi-Agent Systems and Agreement Technologies*, pages 85–99. Springer, 2020.

Paolo Zappalà, Marianna Belotti, Maria Potop-Butucaru, and Stefano Secci. Game theoretical framework for analyzing blockchains robustness. In *International Symposium on Distributed Computing (DISC)*, 2021.

Introduction

Looking back to the last half century of computer technologies, architectures and related design practices, we can observe a fluctuation trend between the centralization and subsequent decentralization of computing resources such as computing power, storage, infrastructure, protocols, and code. Mainframe computers are largely centralized, housing most of computing resources. Today, computational capabilities are distributed on the clients, the clients facilities, and on distant servers. This approach gave rise to the 'client-server' architecture which supported the development of the Internet and relational database systems. Massive data sets, originally housed on mainframes, can move onto a distributed architecture, with data replicated from node to node, or server to server, and subsets of the data can be accessed and processed on clients, and then, synced back to one of the servers.

Over time, Internet and cloud computing architectures enabled global access from a variety of computing devices ; whereas mainframes were largely designed to address the needs of large corporations and governments. Even though such an Internet/Cloud architecture is decentralized in terms of hardware, it has given rise to application-level centralization. Currently, we are witnessing the transition from centralized computing, storage, and processing to decentralized architectures and systems. The DLT is the key innovation making this shift possible. Some distributed systems (e.g., permissionless blockchains) aim to give the control of digital assets to end users without the need for intermediate nodes. Others (e.g., permissioned blockchains), attempt at maintaining a logical centralization of some information while adopting a decentralized architecture. Not all DLTs make use of a block architecture and can therefore be defined as 'blockchains' (e.g., *The Tangle* and *BigchainDB* [9; 10]). Blockchain can hence be used in diverse sectors with several applications. Indeed, it is crucial for adopters to understand whether the technology fits the problems they are aiming to solve or not.

The disruptive technology born in 2008 with Bitcoin and known as blockchain represents a significant quality leap from the distributed database technology studied since June of 1970 (when the first paper on the subject [11] was published). Blockchain allows sharing a ledger of transactions that are read, validated and stored in a chain of blocks. Systems based on blockchain technology work in a distributed manner, involving multiple agents or participants that ought to be independent of each other, and which can use peer-to-peer communications (P2P) to structure themselves into a network collectivity (where nodes represent users connected via transactions). The adoption of P2P as communication paradigm adequately supports the goal that resources are shared and dispersed over a network which by construction forbids the existence of providers or servers centralizing tasks. The result is a decentralized ecosystem with no central authority.

In contrast to legacy client-server architectures [12], P2P network nodes do not always a fixed hierarchy ; roles may not be predetermined, or may change over time depending on the actual operation behind a communication, i.e., a blockchain transaction.

Blockchain Transactions and Users' Roles

Whenever a user aims at interacting with another one in a blockchain network, one or multiple transactions are created, propagated, validated and confirmed by the network. This journey starts at the moment in which the transaction is created and ends when the transaction is recorded in the blockchain. Four crucial steps of the journey of a blockchain transaction can be identified :

— *Creation* : the sender of a transaction must define, according to the data model, the origin and the destination of "the object of the transfer" (i.e., the digital asset). Transactions must specify as well the conditions under which the transaction object can be redeemed (i.e., the conditions to update the system state). Depending on the model, redemption criteria can be simple scripts or more generally actual contracts (smart contracts).

— *Propagation* : the transaction (eventually in a block) is propagated to the validating peers. An efficient transaction broadcasting has an impact on the transactions processing speed.

— *Validation* : it is the most crucial step since it characterizes all the existing blockchain-based systems. At this step transactions, collected in blocks, must address the different stages of the consensus mechanism envisaged to be considered valid and therefore executable. Afterwards, the block of transactions can be attached to the blockchain, updating its state. The valid transactions block is propagated throughout the network in order to let all nodes to update their own replica.

— *Confirmation* : blocks of transactions give rise to a real transfer of assets only if, once validated and eventually published on the blockchain, they are confirmed in the final version of the ledger from which they may no longer be discarded. To become part of it, the consensus procedure has to come to the end, i.e, nodes have to agree on a single chain of blocks.

Transitions from one step to another characterize the technology. Cryptography is involved with hashing and key-generation techniques. Verification checks and block formation may connect the two first steps or the central ones. These four steps mark a strong distinction between two levels of technology use : (i) using the technology to perform simple transactions to then propagate to the network (i.e., creation and propagation) and (ii) helping to maintain the system by participating in the validation and confirmation phase of the transactions (collected in blocks).

Validators are all the peers involved from the moment in which the transaction is included in a block (or its outputs are collected in a block) upon its publication on the ledger. Peers collecting transactions (or transactions outputs) in blocks may not enter the validation phase however, since for many blockchain systems transaction collection is a part of the validation process, this separation is not emphasized throughout the manuscript. Section 1.3.5 presents a more detailed division of the roles blockchain users assume, we highlight in the following the key ones.

— *Transacting parties* : a blockchain transaction involves two different types of actors related to single or multiple blockchain users : the *data-sender* and the *data-receiver*. Interactions take place at address level : the *sending-address(es)* and the *receiving-address(es)* digitally track the data-flow (i.e., the transfer of digital assets) between the parties.

— *Validating nodes* : validating actors run the consensus algorithm and are responsible for establishing the agreement on the proposals made by other validators or by leading nodes (see Section 1.3.5). The validation of a block represents the consensus among validating nodes on which block to publish and in which order.

Blockchain Users' Behaviors

As multi-agent systems, blockchain systems characterized by different layers and protocols, can be modeled as *games* to be solved using game theory. Agents characterizing blockchains are human being or systems with behaviors programmed by human being even if referred to as *nodes*, *peers, agents* or *players*. They are indeed agents who can be considered as rational. Being rational does not consist trivially in behaving in a selfish manner without considering the satisfaction of the others. On the other hand, rational agents aim to get the best for themselves and this can sometimes coincide with the best for the community. Rationality has two strong connotations, (i) it assumes agents are able to order outcomes according to some preferences and that (ii) they are able to analyze the problem they are dealing with and take the right decisions. These two strong assumptions are not always met by blockchain users, leading them to be classified as altruistic and Byzantine nodes according to [13].

Blockchain users play specific games depending on their specific roles within the blockchain system ; they may participate in the consensus phase, they may simply interact among each other via transactions or again they may be interested in attacking the system to damage it or with the mere intention of ending up with a gain. The behaviors they may have can be honest, dishonest or armful. Analyzing all these different behaviors of blockchain users helps in building robust blockchain protocols characterizing robust blockchain systems.

This manuscript analyzes the two main types of blockchain users (i.e., transacting parties and validating nodes) as well as different blockchains (i.e., permissionless and permissioned) with the scope of providing a general overview of the topic and formal results on blockchain users' behaviors ; users may indeed be honest *vis-à-vis* of other users or they may behave maliciously (as Byzantine nodes) attacking the blockchain system. Both blockchain users are at first modeled as rational agents. This game-theoretical modeling enables to capture several behaviors. To be more generic (i.e., including also irrational or unexpected malicious behaviors) blockchain users are modeled also as Byzantine agents ; this is to assess the robustness of general protocols deviations in the blockchain context. The manuscript is structured as follows :

- **Chapter 1** presents a comprehensive overview of blockchain technology analyzing its key features as well as all the layers characterizing the blockchain architecture. It particularly focuses on the features that lead possible blockchain users' decision on (i) whether to adopt the technology or not and, (ii) which type of blockchain to choose. The content of the chapter originates from [14].
- **Chapter 2** provides an introduction to Game Theory, focusing on the *rationality of the players taking part in in a game*. This enables to predict players' actions and thus the outcomes of the game that can be more or less stable (i.e., equilibria or solution in the core). Game participants may play as an individual or as a group leading to two different types of modeling.

- **Chapter 3** models and analyzes the behaviors of a particular type of *validating nodes* ; Bitcoin miners. Miners who operate validating tasks in the Bitcoin blockchain are modeled as rational agents who may or may not behave maliciously (i.e., perform an attack) in certain situations. The situation we present in this chapter is the process of remunerating miners for their validating work. Two types of malicious behaviors are analyzed ; *pool-hopping* and *block-withholding.* The content of the chapter originates from [15] and [16].
- **Chapter 4** is devoted to the analysis of the behaviors of *transacting parties* aiming at exchanging assets through cross-chain swaps. Cross-chain swaps enable interactions between multiple blockchains i.e., blockchain interoperability. Game theory is used to model transacting parties (who swap crypto-assets) as rational agents and characterize the equilibria of existing cross-chain protocols. The content of the chapter originates from [17].
- **Chapter 5** brings together the two types of users that are analyzed in Section 5.3 (i.e., validating nodes for *layer-1* blockchain protocols) and Section 5.4 (i.e., transacting parties for *layer-2* blockchain protocols), respectively, in a more general way, going beyond the strong assumptions (and repetitive limitations) of the rationality of the agents present in the previous chapters. This chapter proposes a game theoretical framework that characterizes the robustness of general blockchains systems in terms of resilience to rational deviations and immunity to Byzantine behaviors. Rational blockchain users faced with a blockchain protocol to follow may act altruistically (following the protocol) or maliciously (deviating from the protocol) according to their personal utility. On the other hand, Byzantine users deviate from the protocol in any situation (even acting irrationally). We prove the practical interest of our formal framework by characterizing the robustness of two layer-1 blockchain protocols (i.e., Bitcoin [2] and Tendermint [1]) and three layer-2 blockchain protocols (i.e., Lightning Network [3], a side-chain protocol [4] and a cross-chain swap protocol [5])). The content of the chapter originates from [18].

The following table represents the analysis structure of the manuscript for Chapters 3-5. The first two chapters introduce the topics of “Blockchain” and “Game Theory”.

		Behavior Model	
		Rational	*Byzantine*
Blockchain	*Validating nodes*	Chapter 3	Chapter 5 (Section 5.3)
User	*Transacting parties*	Chapter 4	Chapter 5 (Section 5.4)

Chapter 1

State of the Art on Blockchain Technology and its Users

Content

This chapter provides a comprehensive overview of blockchain technology analyzing its key features as well as all the layers characterizing the blockchain architecture particularly focusing on those that help a potential blockchain users to decide (i) whether to adopt the technology or not and, (ii) which type of blockchain to choose. The content of the chapter originates from [14].

1.1 Introduction on Blockchain and DLTs

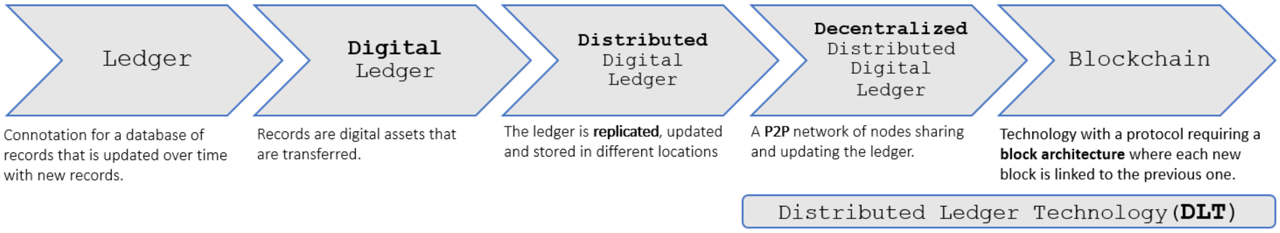
Blockchain can be regarded as a quality leap from the distributed database technology [19] studied since the seventies, which consists in a transaction database shared by different users. Generally, Distributed Ledger Technologies (DLTs) are designed to deal with database in the form of data shared in a distributed manner, and blockchain represents one possible DLT to do it (see Fig. 1.1).

Fundamental bricks in the design of a blockchain technology are as follows : (i) communications and transaction data storage are regulated by cryptographic security, network nodes have to agree on both the validity and the order in which transactions are listed in the blockchain and (ii) distributed consensus protocols solve these issues in a scenario where each node comes to vote.

The first example of such a blockchain is Bitcoin, proposed in 2008 by its anonymous identity [20]. The Bitcoin behavior traces what can be defined as the 'classical' blockchain, consisting in a *permissionless* blockchain alternative enabling a digital, distributed and decentralized payment system. The Bitcoin blockchain is structured in order to protect the ecosystem against attacks launched by malicious or simply rational nodes of the network. As attackers may exploit blockchain vulnerabilities in several ways to achieve a privileged position on the network, the Bitcoin blockchain was designed primarily for preventing the so called *double spending* and *Sybil* attacks, without addressing other important aspects [21; 22] such as :

(i) complete anonymity – Bitcoin provides its users with only pseudonymity ;

(ii) blocks have a limited size, limiting both the number of transactions that can be validated with one block and the number of validated transactions per second (*tps*) – Bitcoin has a 1 MB limit with a transaction rate ranging from 3.3 to 7, incomparable to current credit card systems managing tens of thousands tps [23] ;


Ledger
Digital Ledger
Distributed Digital Ledger
Decentralized Distributed Digital Ledger
Blockchain
Distributed Ledger Technology (DLT)

Connotation for a database of records that is updated over time with new records.

Records are digital assets that are transferred.

The ledger is **replicated**, updated and stored in different locations

A **P2P** network of nodes sharing and updating the ledger.

Technology with a protocol requiring a **block architecture** where each new block is linked to the previous one.

Figure 1.1 – DLT evolution : from the traditional ledger to blockchain.

(iii) eco-sustainability of the validation process – Bitcoin is designed to make it difficult to validate blocks with validating agents or *miners* required to solve computationally heavy crypto-puzzles, and therefore consuming energy.

As a consequence, even if Bitcoin remains the most successful cryptocurrency in circulation, a large number blockchain-based cryptocurrencies have been defined – as of [24], more than 50 alternative cryptocurrencies exist. Some of these 'Altcoins' [25] can guarantee anonymity, solve the energy consumption issue, reduce the price volatility (Stablecoins [26]) or rely on a *permissioned* blockchain – accessible only to authorized nodes, in order to offer a more scalable and fast system.

Going beyond the Bitcoin case, the general blockchain technology aims at assuring the third party benefits such as integrity, authenticity, security and non-repudiation in a distributed and decentralized environment. In addition to auditability and transparency, it offers immutability [1]. Besides being evident for currency systems, these features are useful for any transactional system that is to be used by multiple independent trustless parties.

With the introduction of permissioned blockchains, users may opt for its adoption by placing constraints and customizing the behavior of network nodes. While with classical blockchains it is possible to build a completely open and decentralized system, permissioned ones allow only a limited number of users to have the right of validating transactions. Validators constitute a set of nodes that can be publicly elected or selected by a central authority. Limiting the number of participants in the validation procedure can grant significant scalability improvements by using appropriate consensus mechanisms. Moreover, protocols changes (in both the blockchain data and consensus structure) made to support the execution of Turing-complete codes, facilitate the deployment of distributed applications ('dapps') based on *smart contracts*. However, since full-permissioned blockchains have many similarities with classic shared databases, there can be situations where such a complex architecture is not indispensable.

The literature published after 2014 widely covers blockchain technology in a comprehensive manner but the majority of the articles and surveys analyze it without dwelling on the permissionless part rather than on the permissioned one. Concerning DLT, a term coined in [28] in 2016, many works also

[1] With the term immutability we refer to the concept of "immutability unless the adversary thresholds exceedance" : a permissionless blockchain become mutable whenever the majority of the network efforts are devoted for the purpose of replacing validated blocks, a permissioned one can become mutable following an attack by $\frac{1}{3}$ of the network (see Section 1.4.1.3). (stored transactions are not editable once published) and pseudonymity [27] to its users.

address the comparison between blockchain and previous technologies.

Most of the articles focus on cryptocurrency blockchain-based systems, with different focus on all their aspects. *Tschorsch* and *Scheuermann* [29] present a complete work covering all aspects of the Bitcoin protocols, addressing security, network and privacy aspects. *Conti* et al. [21] survey security and privacy issues of the Bitcoin blockchain, while *Khalilov* et al. [27] focus on surveying techniques enhancing anonymity and privacy in blockchains based on PoW consensus with an emphasize on Bitcoin. Network aspects and related attacks are surveyed by *Neudecker* and *Hartenstein* [30]. Mining procedures for cryptocurrency are presented by *Mukhopadhyay* et al. [31]. Consensus mechanisms constructed using the Bitcoin architecture are surveyed by *Sankar* et al. [32] and *Garay* et al. [33].

Besides crypotocurrency-oriented works, general technology aspects are also covered by other articles presenting differences among permissioned and permissionless blockchains. *Zheng* et al. [34; 35] presented a key features overview for blockchains, covering both public and private modes. Consensus protocols in blockchains are surveyed in [36] and [37], the latter focusing on consensus evolution from the Bitcoin blockchains to the private ones. *Wang* et al. [38] presented the design methodologies for consensus incentive mechanisms in blockchain. *Li* et al. [39] surveyed attacks against blockchain networks, while security issues and challenges are briefly presented in [40].

Furthermore, a comprehensive overview of blockchain applications and use-cases is provided by [41]. Generic IoT (Internet of Things) blockchain applications are presented by *Ferrag* in [42]. Recently published, two tutorials [43; 44] present comparisons between permissionless and permissioned blockchains relating them to technology use-cases.

In this chapter, as in our article [14], we explore all the layers characterizing the blockchain architecture (i.e., network layer, data model layer, execution layer, consensus layer and, application layer), particularly focusing on those that are crucial for deciding (i) whether to adopt the technology or not and, (ii) which of the available blockchain solution come closest to a certain use-case. The chapter is organized as follows. Section 1.1 provides an overview of blockchain and Distributed Ledger Technology (DLT) while Section 1.2 presents the basic features of the technologies and its architecture. In Section 1.3 we present the journey of a generic blockchain transaction ; we go through creation, propagation and validation steps. Section 1.4 describes the agreement problem, its history in the blockchain context and presents the different types of algorithms adopted by existing DLTs. The decision steps leading a potential blockchain user to adopt the technology are presented in Section 1.5.

1.2 Blockchain Structure and Features

A fundamental element beyond the innovation brought by blockchain to the DLT ecosystem is its intelligent mix of encryption techniques [45] in data storage – preserving block structure through timestamping [46] – and in transacting – authenticating transfers with digital signatures. A blockchain ledger consists of a history of validated digital transactions collected in blocks ; each block of transactions is linked to the immediately previous one (known as *parent block*) through a hash value ; hence by traversing the transactions ledger one can trace back the genesis block, which has no parent block and contains the first processed transactions in the blockchain history. Cryptography characterizes the technology and attributes important properties to it. Let us start familiarizing the reader with the terminology and the different blockchain participation modes. Afterwards, we present the features of the technology and its architecture.

1.2.1 Terminology

— *A distributed ledger* is a type of digital data structure residing across multiple computer devices, generally at geographically distinguished locations [47].

— *Distributed Ledger Technology (DLT)* designs a type of technology enabling storing and updating a distributed ledger in a decentralized manner. As shown in Fig. 1.1, the blockchain and all its variations belong to the spectrum of DLTs. While distributed ledgers existed prior to Bitcoin, the Bitcoin blockchain was novel in that since marking the convergence of a set of existing technologies (including timestamping of transactions, P2P networks, cryptography, and shared computational power) and enabling data sharing and storage without entrusting any central party for the ledger maintenance. DLTs consist of three basic components :

1. a data model that captures the current ledger state ;
2. a communication language defined by transactions that change the ledger state ;
3. a protocol used to build consensus among participants around which transactions are accepted by the ledger and in which order.

— *A blockchain* is a P2P DLT structured as a chain of blocks, forged by consensus, which can be combined with a data model and a communication language enabling smart contracts and other assisting technologies. Cryptography lets blockchains overcome former DLTs by offering secure data-transmission and by enabling records immutability, in a decentralized environment (see Section 1.2.5). Hence, a blockchain is an immutable read-only data structure, where new entries (blocks) get appended onto the end of the ledger by linkage with the previous block's 'hash' identifier.

The collection of these features can be used to build a new generation of transactional applications that establish trust, accountability, and transparency at their core, while streamlining business processes and legal constraints. In all DLTs, there is an initial record - in a blockchain it is called a *genesis block*. Each block includes one or more transactions. Connecting to a blockchain involves users connecting to this distributed ledger via, typically, an application. The blockchain ledger consists of digital transactions representing interactions between nodes of a P2P network.

- *Transactions* are individual and indivisible operations that involve exchange or transfer of digital assets. The latter can be information, goods, services, funds or set of rules which can trigger another transaction.
- *Blockchain nodes* are computing device connected to the blockchain that support the network by maintaining a copy of the ledger. Records replicas are stored by *full nodes* which verify blockchain data integrity. There can be nodes that, when connecting to the blockchain, do not download the whole ledger but just a subset of it ; these *lightweight nodes* – served by full nodes allowing them to transmit their transactions to the network – download the headers of all blocks on the blockchain in order to verify only if a transaction has been included in a block. Whenever blockchain nodes exchange assets via transactions in the network they are considered as *blockchain users*. In order to transact with the network peers [2], they generate a cryptographic key-pair (see Section 1.2.4). If the private key is used to sign transactions, the public key is the one identifying the user(s) *address* storing exchangeable assets (e.g., addresses with tokens defined as accounts or wallets).

Blockchain transactions are grouped into blocks, and there can be any number of transactions per block while respecting a given block size limit. Nodes on a blockchain network group up these transactions and send them throughout the network. Eventually peers synchronize to an exact copy

[2] The term "peer" denotes those blockchain nodes that are directly connected. Nodes that are initially alone seek to establish new connections with a certain number of peers (e.g., 8 for Bitcoin) in order to be part of the network. The terms node and peer are therefore interchangeably used.

of the blockchain throughout the network. The blockchain updating procedure needs an agreement, e.g., a consensus among the network peers.

— *Consensus* or more generally *agreement* in the network refers to the process of achieving agreement among the network participants as to the correct state of data on the system. It leads to all nodes sharing the exact same data. Therefore an agreement algorithm (i) ensures that the data on the ledger is the same for all network nodes, and (ii) prevents malicious actors from manipulating the data.

The agreement procedure varies with different blockchain implementations. While the Bitcoin blockchain uses a *PoW*-based consensus mechanism, other blockchains and distributed ledgers are deploying a variety of consensus algorithms belonging to two main classes : (i) *Proof-of-X*-based algorithms and (ii) *Byzantine Fault Tolerant* algorithms. We elaborate about agreement and consensus algorithms used in DLTs in Section 1.4.

Early blockchain-based systems were meant for managing digital currencies. However, a generic DLT can fit any digital asset exchange requirement. Contractual aspects of an exchange, involving nodes' rights and obligations, can be digitalized and controlled by proper digital (smart) contracts.

— A *smart contract* is a computer program that executes predefined actions when certain conditions within the system are met. Smart contracts provide the transactions language allowing the ledger state to be modified. They can facilitate the exchange and transfer of any asset (e.g. shares, currency, content, property). They reside into the blockchain structure and are triggered along with transactions. Smart contracts can be imagined as digital protocols used to facilitate and enforce the negotiation of a legal contract. Actions carried out by trusted third-parties during a trade are replaced by pieces of code.

Having acknowledged that blockchain ledgers fit within a wider spectrum of technologies, our contribution focuses on the analysis of blockchain systems characterized by a permissionless or permissioned participation mode.

1.2.2 Permissionless and Permissioned Participation Modes

In conventional central data storage systems, only a single entity, the owner or the administrator, keeps a copy of the database. Consequently, this entity controls what data is contributed and what other entities are permitted to contribute. With the advent of DLT this radically changes in favor of distributed data storage where multiple entities hold a copy of the underlying database and are naturally permitted to contribute. Data is replicated for all entities participating in a distributed ledger in a network of so-called peers. Due to distributed data storage, the difficulty arises to ensure that all nodes agree upon a common truth, i.e., the correctness of a ledger, as changes made by one node have to be propagated to all other peer nodes in the network. The result of arriving at a common truth is referred to as consensus among nodes.

With respect to accessing the blockchain network, there are two main modes of operation : ***permissionless*** and ***permissioned*** – it is worth noting that in the literature, these are often referred to as public and private blockchains, respectively, but we use in this thesis a more precise taxonomy as explained hereafter. The same division is adopted regarding the participation to the ledger maintenance procedures, i.e., the possibility to modify (update) the network state. In the first mode, participation is public and open-access : anybody is allowed to participate in the network and in the consensus process [48] ; this mode is the one adopted by first generation blockchains (e.g., Bitcoin). On the other hand, if participation is permissioned, participants have either restrictions on writing (validation) rights only, or on both reading (access) and writing rights. In the first case, permissions concern the participation to the phases of the transaction journey (see Section 1.3) amending the log ; any modification of the transaction ledger is entrusted to a selected set of nodes. Instead, the so-called ***full-permissioned*** blockchains select participants in advance and restrict any sort of activity in the network to these only.

The participation mode differentiates between decentralized blockchain-based ledgers and those that additionally offer disintermediation namely, that cut out any middleman (i.e., permissionless blockchains). It is worth stressing that in permissionless blockchains anyone with an Internet connection can join the network, as well as write and read transactions ; this is why permissionless, public and open-access are terms used interchangeably to refer to such technologies. Participants here are pseudonymous, which is not preventing malicious nodes to act within the network. Contrariwise, full-permissioned blockchains, reduces these security risks by whitelisting authorizations to join the network. In this way, rather than displaying the transactions record to the entire Internet community, transactions remain visible only to a private network of nodes.

The differentiating points in the previous two paragraphs allow us to support what authors in [49] propose, i.e., differentiating full-permissioned blockchains from those allowing anyone to read the

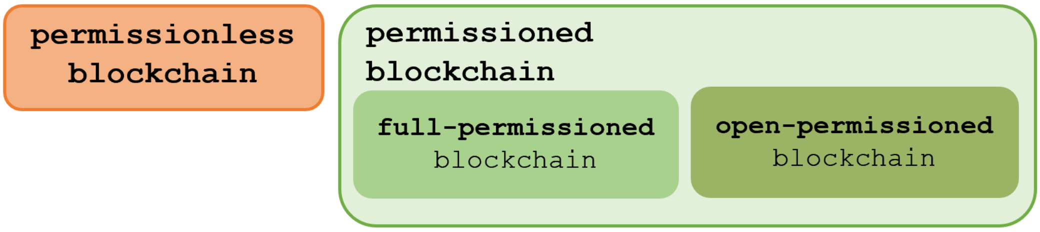

permissionless blockchain
permissioned blockchain
full-permissioned blockchain
open-permissioned blockchain

FIGURE 1.2 – The different participation modes characterizing blockchains and DLTs. After a first distinction based on the reading rights (i.e., permissionless and permissioned), the second distinction is based on writing rights of the data on the ledger.

blockchain state, denoted in [49] and in the following as ***open-permissioned blockchains***.

With respect to the nature of participants, permissioned blockchains can be further classified in *private* blockchains – where the participants are within the same organization – and *consortium* blockchains – where the permissioned blockchain is deployed among several organizations (consortium). A consortium blockchain represents a joint effort of several entities sharing a common goal or business need. Furthermore, 'private' and 'consortium' attributes can be linked to the blockchain governance system. There are some developed by a single enterprise, and others by a joint effort of several contributors (e.g., Corda and Hyperledger [50; 51]). The latter, for instance, is a cross-industry project led by the Linux Foundation to advance blockchain technology by coming up with common standards. Fig. 1.2 illustrates the different DLT participation modes presented. The participation mode has a braking impact on the decentralization trend in distributed consensus, as we develop in Section 1.4.

1.2.3 Data Structure

A block is the junction of (i) an **outer header** identifying the blockchain and specifying the size of the block, (ii) a **block header** – containing all the information on the block validation and on its parent block – and (iii) a **block body** – consisting in a list of transactions and a transaction counter. While the precise structure of a block varies from one blockchain to another, each blockchain is identified by the *magic number*[3] which is included in any block of transactions at the beginning together with the *blocksize* field reporting the maximum number of bytes in a block. The block header should include for every blockchain system, as in Fig. 1.3, the following elements (whose order can vary from one

[3] The magic number consists in a data-structure identifier characterizing the different blockchain protocols (i.e., 0xD9B4BEF9 is the magic number identifying the Bitcoin blockchain)

blockchain to another one) :

- Block version number : it refers to the blockchain protocol and hence the used consensus algorithm followed by the (majority of the) nodes at the moment of the block validations.
- Parent-block hash : it is the output of the hashing function with the previous block header as input.
- Nonce : it is a string of fixed length crucial in the validation process (Section 1.3.3).
- Timestamp : it indicates the time elapsed since a predefined instant.
- Merkle tree root : it is the hash value descending from a hash tree procedure (patented in 1989 [52]) applied to the transactions present in the block body ; transaction informations are iteratively hashed in pairs as showed in Fig. 1.4 (if the number of transactions is odd, the last transaction, hashed or not, in the list is duplicated).

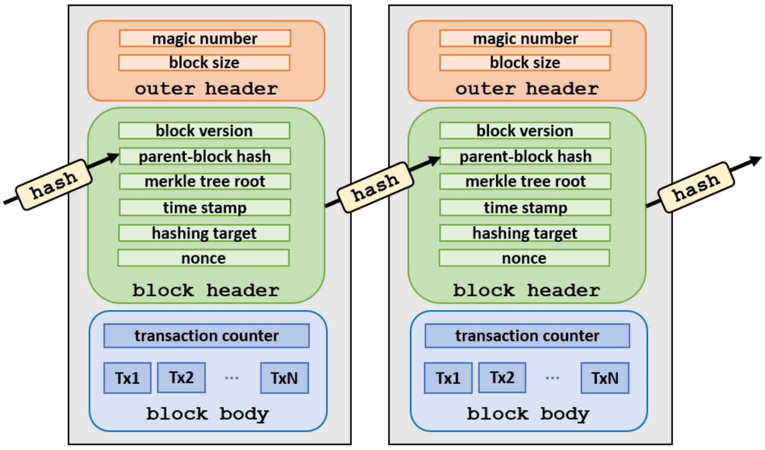


FIGURE 1.3 – Representation of a blockchain structure.

The hash of the block header serves as a link to future new blocks on top of it. The block body consists of all the transactions involved in the Merkle root calculation and of a transaction counter providing the total number of transactions contained in the block. Note that the block size limit has a direct effect on the number of transactions that can be included in the block body.

1.2.4 Cryptography

Cryptography allows sending data trough trustless channels in a secure and verifiable way. Data hashing consists in a basic cryptographic operation that not only compresses data in a fixed-length format, but it does so irreversibly, which is crucial for ensuring the integrity of digital assets when transferred in the network. Asymmetric cryptography authenticates the data source and ensures its reception by the desired user. Blockchain combines asymmetric cryptography with hashing and *digital signature* schemes in order to provide fundamental security guarantees presented later on.

More precisely, a digital signature scheme consists of three phases as depicted in Fig. 1.5 :

— *Key-pair generation phase* : each blockchain user generates a private key to sign a transaction with and a public key by which the receiver can verify the authenticity, integrity and provenance of the received data.

— *Signing phase* : the sender hashes the data and generates the digital signature with its private key ; next, the signed hash is sent together with the encoded original data to the receiver. Data hashing not only makes the signature scheme more streamlined and efficient (data are compressed and have the same format), it also ensures the integrity of the transferred data (transactions contents are protected against being modified).

— *Verification phase* : the signed data is decoded with the sender's public key and compared with the re-computed hash value of the unsigned and uncompressed data.

Note that, in both the signing and verification phases the hashing function used must be the same (e.g., SHA256 for Bitcoin blockchain). Blockchain requires asymmetric algorithms – generating both public and private key – that allow for fast verification (the time taken for signing shall be the same as for the last phase). Digital signature algorithms in blockchains widely use elliptic curves (ECDSA [53; 54]).

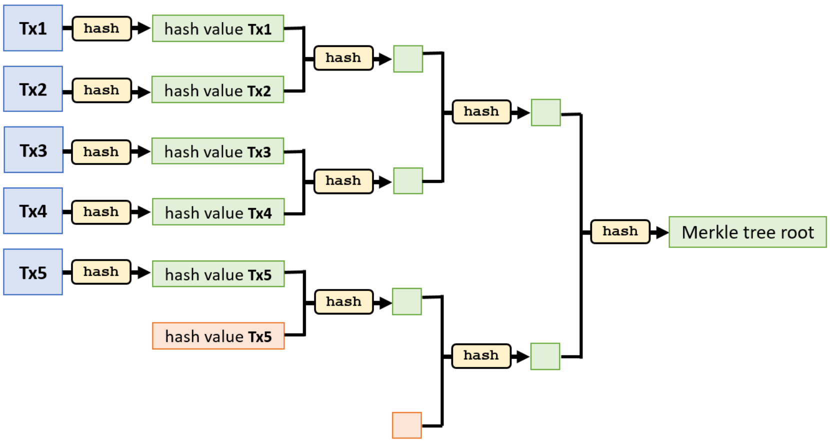


FIGURE 1.4 – Merkle hash tree procedure example : duplicated (hashed) transactions are marked in orange.

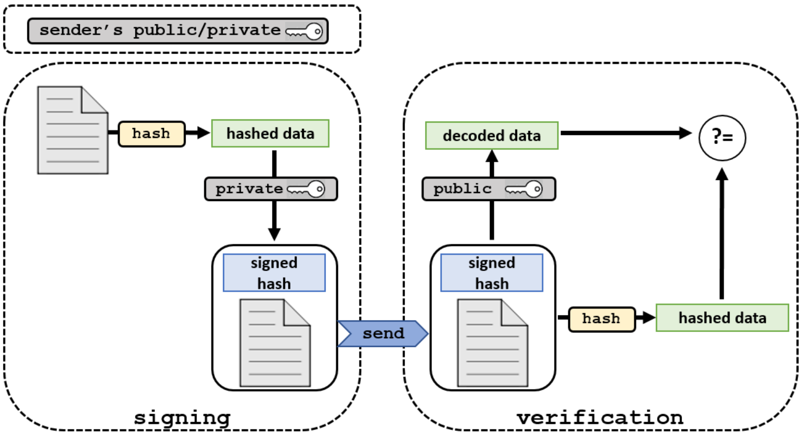


FIGURE 1.5 – Phases of the digital signature protocol : (i) a public/private key pair is created – the public key can be recovered from the private one while the viceversa is not possible, (ii) data are signed – the signature is the result of encoding with the sender's private key the hashed data – and transferred. Once received (iii) the receiver decode data by the usage of the sender's public key and additionally verifies its authenticity.

1.2.5 Blockchain Features

Thanks to the explanations of the previous paragraphs, we can now highlight six fundamental blockchain features, which are obviously dependent upon each others :

— *Decentralization* : DLTs enables P2P data sharing and storage without entrusting the ledger maintenance to any central authority. It does not mean completely cutting out intermediaries that validate transactions (*disintermediation*) like permissionless blockchains do, but rather decentralizing them along with their roles.

— *Immutability* : while shared ledgers allow data manipulation by a central authority, distributed ledgers working with replicated information protect data from any sort of tampering and falsification; except in situations where the majority of the network's efforts are devoted to change the registry [55] (e.g, the Ethereum DAO fork [56]) or where the adversary thresholds are exceeded (see Section 1.4.1.3). Data immutability makes data accessible and manageable by different entities that do not trust each other.

— *Integrity, Authenticity and Non-Repudiation* : the data hashing grants that data is not modified during its transmission (i.e., integrity). Moreover, the origin of a transaction can be ascertained by the senders' public key dissemination, while the evidence of the sending action is represented by the data signing procedure involving the private key (i.e., authenticity and non-repudiation).

Blockchain signing scheme combining asymmetries cryptography and data hashing is presented in Section 1.2.4.

— *Auditability* : Transactions in blockchain systems must be validated and verified thus, each data transfer should be visible to all blockchains participants in its entirety. In this way, all blockchain operations are traceable via audits. Users accessing the first generation blockchains can see the data ledger in its entirety. Indeed, recent implementations enable multiple ledger to be isolated and maintained within the same blockchain system via private channels. Nevertheless, ledgers data is visible to all channel participants, thus the auditability is satisfied at channel level.

The mix of the above features qualifies the technology at a quite high level of dependability, differentiating it from the classic distributed database. Blockchain features result strictly correlated with the consensus mechanism in use.

1.3 Journey of a Transaction

Generally, transactions in blockchains or DLTs are not strictly financial and do not just carry and store transaction data. Hence, the usage of blockchain transactions is not limited to the simple assets exchange, but it also covers the execution of computing instructions such as *storing*, *querying* and *sharing*. Every transaction, once validated, is placed in a new block which is added in the transaction ledger and linked to the previous one. This results in an update of the system state and of users' local copy of the blockchain.

Whenever a user aims at interacting with another one in the network, one or multiple transactions are created, propagated, validated and confirmed by the network. Each blockchain-based system differs from the others by the way in which the steps of the 'transaction journey' are performed. This journey starts at the moment in which the transaction is created and ends when the transaction is recorded in the blockchain. Four crucial steps of the journey of a blockchain transaction can be identified :

— *Creation* : each blockchain adopts a predefined data-structure that determines certain benefits and drawbacks. Some data models are designed for specific blockchain applications, others are designed to be as flexible as possible. The sender of a transaction must define, according to the data model, the origin and the destination of "the object of the transfer" (i.e., the digital asset). Transactions must specify as well the conditions under which the transaction object can be redeemed (i.e., the conditions to update the system state). Depending on the model, redemption criteria can be simple scripts or more generally actual contracts (smart contracts).

— *Propagation* : the transaction (eventually in a block) is propagated to the validating peers. An efficient transaction broadcasting has an impact on the transactions processing speed. The communication protocols adopted by blockchains aim at optimizing the network performance while being resistant to manipulations and attacks.

— *Validation* : it is the most crucial step since it characterizes all the existing blockchain-based systems. At this step transactions, collected in blocks, must address the different stages of the agreement mechanism envisaged to be considered valid and therefore executable. Afterwards, the block of transactions can be attached to the blockchain, updating its state.

— *Block Propagation* : the valid transactions block is propagated throughout the network in order to let all nodes to update their own replica.

— *Confirmation* : blocks of transactions give rise to a real transfer of assets only if, once validated and published on the blockchain, they are confirmed in the final version of the ledger from which they may no longer be discarded. To become part of it, the agreement procedure has to come to the end, i.e., nodes have to agree on a single chain of blocks. Once a block of transactions is confirmed, users can be sure or mostly sure (depending on the agreement mechanism of the blockchain) that the validated block cannot be changed or removed from the blockchain ledger.

1.3.1 Transaction Creation

Whenever a user aims at interacting with another one in the blockchain network, a transaction takes place. In general, a transaction indicates to the network that a user has authorized a data flow. Hence, it has to be properly constructed for its purpose before its propagation.

Firstly, the sender user has to build a transaction *proposal* specifying all the criteria according to which the information can flow to the transaction receiver(s). All blockchain transactions must specify the destination of the operation, in most cases provided with a unique transaction identifier. Moreover, a transaction field reporting the entity of the transfer must exist ; i.e., in the case of cryptocurrencies a certain amount of tokens is specified in the amount field of the transaction. Blockchain technology supports the presence of both multiple *origins* and multiple *destinations* ; a transaction sender may have more receivers and vice-versa.

The transaction proposal must be signed by the sender(s) to prove the ownership of the address(es) instantiating the transaction. Blockchain-based systems use digital signatures as authentication methods (as presented in Section 1.2.4). Once signed, the transaction can move on to be propagated in the P2P network. Privacy-preserving blockchains – trying to hide the source, the destination and the entity of a transaction – can make use of temporary addresses and special cartographic tool to sign and encrypt transactions before the propagation [27].

The data model of a blockchain transaction differs depending on the system implementation and its business application. For instance, the Bitcoin protocol imposes the transfer of *Unspent Transaction Outputs* (UTXOs [57]), presented hereafter. Post-Bitcoin data models have evolved in two different ways. First, blockchains moved to the adoption of an *account-based model*, making use of a completely new transaction syntax (Turing complete) [58] and resulting more 'smart contract friendly' ; Ethereum is one of the so-called 'second generation' cryptocurrencies [59] adopting this record-keeping model.

Subsequently, blockchains' intention was to maintain the original Bitcoin data-structure along with its improvement proposals [60] to which integrate the benefits of an account-based model. General blockchains, going beyond cryptocurrencies and digital assets, may adopt basic models supporting smart contract execution. Offering more and more general operations corresponds to a data model supporting more and more complex logic, hence overcoming both the account and the UTXO models. Blockchain-based systems of this type adopt a *key-value data model* (also called table-data model) where the blockchain registers its state as data-tuples that can be updated. We present in the following these different models in more details ; benefits and drawbacks are summarized in Table 1.1.

UTXO Model.

This record-keeping model associates value to users' addresses as 'unspent' transaction outputs, i.e., cryptocurrency amounts that may be spent in the future through the construction of other transactions ; UTXOs become inputs of a 'spending transaction' transferring the value previously received to another blockchain user. Transactions outputs (TXOs) can only be spent (i.e., transferred) once. Blockchain addresses keeps track of the received UTXOs ; their sum corresponds to the address balance.

A peculiarity of the UTXO model is that transactions inputs and outputs must match ; namely the entire value of the TXOs received in a prior transaction has to be transferred in order to be spent. More precisely, a user aiming at transferring data to another one does nothing more than 'endorsing' a previous received UTXO. Users unlock an output, appropriately transform it and generate a new one ; the procedure, resembling the "compare-and-swap" instruction in computer science, forces a synchronization in data accessing [61]. The problem arises whenever a user has no intention of spending the entire value of a TXO. The issue is solved with the proper use of multiple outputs ; the system creates a transaction with two different outputs : (i) one destined to the receiving user, transferring the aimed value (lower in relation to the TXO) and, (ii) one transferring the difference back to the sender in the form of a new UTXO. In this way, the inputs value corresponds to outputs value. The UTXO model is designed in such a way that each UTXO has to be transferred/spent in its entirety as input of another transaction. That is why operations on UTXO-based blockchains are so reminiscent of exchanging cash. Fig. 1.6 shows how UTXO works in the Bitcoin blockchain marking the difference between TXOs and UTXOs. The state of the whole blockchain is represented by the UTXOs state. Each transaction includes the state of the new output and in order to be updated it

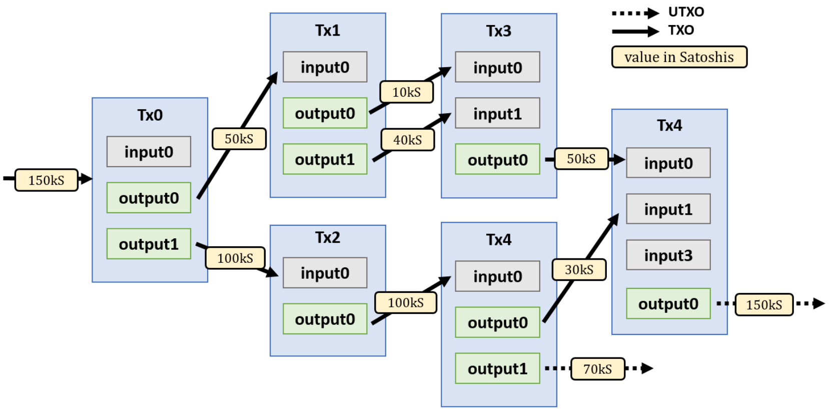


FIGURE 1.6 – An example of UTXO-based transfers in Bitcoin.

has to be included as input of a second transaction. This implies high verification, duplication and transmission costs. Because of these drawbacks, UTXO model forces blockchains to limit the amount of operations impacting the system state.

Bitcoin adopts a transaction structure with three basic fields : (i) the value to be transferred, (ii) a short script specifying the conditions under which the value can be redeemed (i.e., the Locking Script or Redeem Script [62]) and (iii) a *witness* field to unlock the previous transaction output. The script locks the transaction until spending conditions are met, i.e., when a witness is provided. The approach works for simple transactions ("Pay-to-PubKeyHash" [45]) or simple contracts involving a small number of transactions locked with proper locking scripts ("Pay-to-ScriptHash" [63]), however it results not suitable for slightly more complex operations contemplated with smart contracts. UTXO-based applications in Bitcoin should limit the number of transactions involved, because of both the cost in terms of computations required to find a PoW (a *golden nonce* [64]) validating a transaction, and the scripting language supported by the model which is Turing incomplete [65].

Account-Balance Model.

This model results more intuitive in keeping track of the balance of each account as a global state of the blockchain. State replication completely overcomes the concept of transaction input and output ; more precisely, the blockchain state is an outcome of a transaction. Once a transaction is executed the states of the accounts involved in the transferring are updated.

There are different options for creating a transaction depending on the output and the finality ; regular transactions between users have to simply specify the receiving account(s) and the entity of the transfer, while transactions dealing with contracts present rather complex structures.

— *Smart contracts.* The smart contracts idea of Nick Szabo [66], was a key driver in the blockchain evolution since its conception. At the time, he used an analogy with vending machines, that in hardware implements predefined set of rules, an agreement between a machine and a user. Thus, smart contracts are computer codes which digitally represent contracts on a DLT and enable its automatic execution. Smart contracts are distributed to the network of peers characterizing the blockchain and are executed on the blockchain i.e., the outcomes (e.g., exchange of assets, access to property, certification of documents, etc...) are recorded on the ledger. The power of smart contracts is that they remove the counter-party risk making third party services redundant.

Let see it with an example : Alice enters at time T_0 into a contract to buy a land from Bob by providing a deposit upfront and committing to pay the full amount by the time $T : T > T_0$. A smart contract can verify whether the payment took place on time and in that case (i.e., under that condition) it can release the funds and change the title on the property. On the other hand, the smart contract can cancel the contract and liquidate the deposit if Alice did not pay on time. We can see that the code characterizing a smart contract can replace the third party services provided by an escrow service (that keep the deposit), a title company (that verify payments) and a titling (that update the records).

In terms of data model, a smart contract consists of a collection of standard transactions presenting locking conditions : contracts on the blockchain are created as transactions between addresses and they can be executed thanks to *triggering* transactions. For instance, Ethereum works with different types of accounts : Externally Owned Accounts (EOAs) holding only its balance, and Contract Accounts (CAs) holding the code of a smart contract and keeping an internal state. Once a transaction in a contract or a regular one is executed, the ledger is updated together with its state.

Smart contracts play an important role in this thesis contributions as they enable settlement operation between different blockchains (see Chapter 5).

Contrary to UTXO-based blockchain, account-based systems have to deal with several security issues. First of all, the account model is not not immune to double-spending practice. Hence, it is necessary to secure the blockchain adopting this record-keeping model, preventing the same transaction being submitted more than once. Moreover, an anonymity issue arises when accounts are reused ; the account model gives preference to balance updates rather than new account creation.

UTXO$^+$.

The idea beyond the UTXO$^+$ model is to maintain the UTXO structure, to which appropriate changes are made in order to obtain the same benefits granted by the account-based models. There is no notion of 'account' and state is forced to be included in the transactions outputs. Such operations still result quite unnatural and require a deep-level of abstraction together with serious complexities. Corda, Chain Core and Qtum [50; 67; 68] appropriately mix the Bitcoin and the Ethereum data-structures in order to have an UTXO-based model supporting complex contract operations ; both systems adopt powerful virtual machines supporting operations written in Turing-complete code but differently to Ethereum the EVM are stateless.

Key-Value Model.

An evolution of the previous data models consists in including in the state of a blockchain more variables, presenting them as tuples or tables. Such a general approach allows to adopt an UTXO-like or an account-like structure depending on the business constructed on top of the blockchain.

For instance, Hyperledger Fabric offers the possibility to deploy Bitcoin-like currency systems (*Fabric-Coin* [69]), digital assets exchange (i.e., a contract, liabilities, properties) and tangible assets exchange (i.e., real estate and hardware). Fabric represents general assets as collections of key-value pairs (KVP) and it records state changes as transactions outcomes [51]. Kadena [70] adopts a table-based data model operating modification at a *per-row* level. That is, the blockchain registers a columnar history and transactions, both regular and smart contract ones, can update multiple column values at once thanks to a proper object syntax.

Model Comparison.

Major differences between the four models are summarized in Table 1.1. The table reports benefits and drawbacks of the data models and cite some blockchain frameworks adopting them [4]. Transacting using a UTXO model is conceptually equivalent to banknotes exchanging ; the amount of paper bills (UTXOs) in the purse is the balance of our wallet and, whenever users spend money, they pay with a bill covering the cost (existing UTXOs) and they receive a change back consisting in other bills (new UTXOs). Thanks to the analogy, it is easy to note that this record-keeping model provides higher

[4] The DLT platforms we refer to were the mainly adopted ones during the time period of the thesis i.e., 2017-2020

levels of scalability and anonymity ; multiple UTXOs can be processed in parallel and whenever a new address is receiving new UTXOs the identity of the user owning the address is hidden. The account data model is constructed to record each account's balance so as to allow the issue of valid transactions. With accounts resembling traditional banks' debit cards, the blockchain structure results more intuitive and efficient. Adopting a stateful approach, the balance of each debit card is registered in the system and it is not included in the transactions data as for the Bitcoin stateless model.

Table 1.1 – Blockchains data model comparison.

Data model	**Benefits in**	**Drawbacks in**	**Frameworks**
UTXO	scalability, security, anonymity.	applicability, efficiency, intuitiveness.	*Bitcoin*, *Litecoin* [71], *Dodgecoin* [72], *ZCash* [73], *MultiChain* [74].
Account	intuitiveness, applicability, efficiency.	security, anonymity.	*Ethereum*, *Tezos* [75], *IOTA* [9], *Ripple* [76], *Stellar* [77].
UTXO$^+$	scalability, efficiency, security, anonymity.	applicability, intuitiveness, model complexity.	*Corda* [50], *Chain Core* [67], *Qtum* [68].
Key-value	as UTXO and Account.	model complexity.	*Tendermint* [1], *Hyperledger Fabric* [51], *Kadena* [70], *SawtoothLake* [78].

1.3.2 Transaction Propagation

This step results crucial for the correct functioning of the consensus mechanism in the network. In order to establish which transactions are valid or not, all the validating peers must have complete knowledge of the information to be agreed upon. Therefore, transactions must be propagated to validators as fast as possible.

In order to optimize blockchain network performance and scalability, *flooding* or *gossip* protocols [79] are used for the propagation. Transaction propagation is carried out by means of a message exchange amongst peers. Blockchains clients connect only to a limited number of peers (neighbors) ; the message is first propagated to the connecting peers that then propagate it to their neighbors, and so on until it reaches all network nodes. Data present in the messages can be encrypted or not. Blockchain-based systems can require sending peers' authentication via exchange of a public key that can be included in the message or communicated out of band. Hence, receiving peers' can verify the data integrity.

From a networking performance perspective, it is important to establish to which of its neighbors peers have to relay a message. Flooding protocols include message transmission to all neighbors, while according to gossip protocols messages are relayed to a subset of randomly selected neighbor nodes. Both approaches assure a fast information dissemination but they differ in term of bandwidth and delay performance. The design of the transmitted message can impact the transmission delay. Delay-aware or bandwidth-aware neighbor selection can obviously lead to clear forwarding delay and bandwidth gains. A Bitcoin-like *announce-and-request* signaling, adding two more steps in peers communications (i.e., two more round-trip time, RTT, latencies), can consume less network bandwidth at the expense of delayed transmission. Such signaling can also imply a more complex data model : the protocol has to rule peers' request mechanism, peers' access to the data-ledger and peers' verification of the message originality (i.e., whether the information is new or not).

Apart from bandwidth and delay aspects, message propagation has to deal with network privacy and security aspects : multiple connections per node implies a large attack surface, while a limited number of communications facilitates interrupting and avoiding attacks (i.e., eclipse and DoS attacks [80; 81]). Regarding the identity-privacy aspects in permissionless blockchains, P2P protocols can reveal information on nodes identity. Deanonymization practices are related to the blockchain network topology built on top of the P2P overlay network, which can be generally disclosed if global-view P2P network traces are available or can be collected from different peers.

Bitcoin and the first generation of Altcoins work with flooding protocols using an *announce-and-request* signaling, where information is first announced to the neighbors to be sent afterwards, if not already possessed. Even if propagation costs with flooding do increase sub-linearly with the number of neighbors, the dissemination protocol is prone to deanonimization attempts [82] along with destabilizing communication strategies [83] ; starting from withholding (*relay-delay* [84]), ending with net-split and gold-finger attacks [21]. Moreover, even if the announce-and-request signaling can be improved (e.g., compressing information by announcing headers only) or appropriately mixed with the classical push (e.g., Ethereum), the added latencies elapse can be more or less significant.

Permissioned blockchains are superior to permissionless ones also in the communication performance. In permissioned environments where anonymity, message encryption, Sybil attacks do not represent a major issue, the communication security is concentrated on the faulty nodes management, to which gossip dissemination is more resistant with respect to flooding. The dissemination protocol does not require fixed connectivity to work since it operates with an *unsolicited push propagation* [85] mechanism, providing a consistent data synchronization tolerant to node crashes. Permissioned blockchains can count on a fast propagation with low latency (due to the direct *push*) and low bandwidth costs. In order to further speed up the propagation, the push mechanism can be improved reducing the size of the broadcasted messages by disseminating the transactions ID instead of the whole transactions.

Model Comparison.

Table 1.2 summarizes the differences. First generation cryptocurrencies opt for flooding protocols using announce-and-request signaling, leading to higher bandwidth consumption and lower delay performance. Concerning security, the level of attack resistance depends on other factors (e.g., relay-delay). In this respect, Ethereum represents a transition from flooding to gossip adopting a "hybrid" design where some information is pushed and the rest is sent selectively. The gossip protocol promises good performances *pushing* messages ; however, it results more sensible to net-split attacks due to the fewer connections involved in the propagation.

TABLE 1.2 – Blockchains propagation mechanism comparison.

Communication protocol ⟶	**Flooding**	**Hybrid Flooding**	**Gossip**
bandwidth consumption	●●●	●●●	●○○
delay performance	●○○	●○○	●●●
net-split attack resistance	●●○	●●○	●○○
scalability	●●○	●●○	●●●
Basic protocol design ⟶	**Announce-Request**	**Hybrid**	**Unsolicited push**
RTTs	3	2-3	1
delay performance	●○○	●●○	●●●
examples	*Bitcoin*	*Ethereum*	*Hyperledger*

High : ●●●, Medium : ●●○, Low : ●○○.

1.3.3 Transaction (Block) Validation

Before being collected in blocks, transactions must pass the verification checks, i.e., they must have been created in accordance with the network rules. Once verified and inserted in the blocks, validators check whether the blocks meet all the protocol requirements necessary to assign the 'valid' entry and to proceed with the publication. These validation criteria must be deterministic and uniform across the network. While the transactions verification consists in a trivial cryptographic check, the block-validation phase is considered a key passage since it attributes to every blockchain-based system a distinctive character. After verifying that the block proposal has been correctly carried out, nodes have to find an agreement on the validity of the block. More precisely, nodes in the network must agree on a unique record of transactions following a collaborative consensus protocol.

Transactions-ordering and consensus establishment can be considered as separated phases, or can be combined as in most of the existing consensus protocols. Bitcoin combined the two processes in the consensus procedure proposed in [20]. Validators in the Bitcoin network, known as *miners*, have to agree on both the order and the validity of the blocks. Some permissioned blockchains separate these steps (e.g., Hyperledger [51]) : peers can agree on the ordering of the transactions that are validated in a second moment, right before their publication.

The agreement – on both publication and ordering of the transactions in the ledger – is reached through a distributed protocol executed by the nodes involved in the validation procedure. The consensus protocol must solve the Byzantine Generals (BG) problem [86], which consists in reaching consensus among trustless nodes (i.e., generals can be traitors). Since systems must accomplish this agreement state in a distributed manner, protocols should provide a *consistent* (or at least *eventually consistent*) view of the blockchain in the whole network. Thus, protocols adopt data *replication*, meaning that nodes hold replicas of the transaction ledger. Replicating data over nodes in the network makes blockchains resilient.

Building a proper consensus protocol is a challenge, as we develop in detail in Section 1.4. Since blockchain technology has many different use-cases, consensus protocols have been designed to meet specific system requirements. In permissionless blockchain applications, everyone is allowed to participate in the network, executing the consensus protocol and maintaining the shared ledger. The availability of these systems results in a substantial amount of computational power (hence energy) for maintaining a distributed ledger at a large scale (e.g., as in Bitcoin). Permissioned blockchains, with the presence of restrictions on who is allowed to participate in the network, adopt differently designed agreement procedures. More specifically, since the participants using blockchain are whitelisted, consensus protocols in permissioned blockchains guarantee higher performances.

1.3.4 Transaction (Block) Confirmation

Block confirmation coincides with its inclusion in the valid transactions history. Confirmation is the direct consequence of *consensus finality* (i.e., an agreed transactions never change or disappear) characterizing the so-called "consensus-based" blockchains. In this case, confirmation consists of a transaction predicate obtained when the majority of nodes get to decide to validate, and then publish the block containing the given transaction. However, in general, decentralized distributed ledgers may ensure a *probabilistic and economic* consensus finality – since they rely on eventually consistent consensus algorithms [87] – referring to cases in which the block-confirmation probability/cost (depending on the type of consensus) is increasing with the number of validated children blocks . In fact, despite the robustness of permissionless blockchains against double spending attempts (they need the involvement of the majority of the network to be successful), reversals are very common by means of forking attitudes that do not correspond necessarily to malicious intents. Confirmed blocks that cannot be discarded give way to the proposed exchange in the collected transactions. Therefore, in this case block confirmation is not a formal step explicitly notified to blockchain nodes, but it is implicitly inferred by the actual presence of the validated block in the blockchain branch where the majority of nodes concentrate their efforts.

Transitions from one step to another characterize the technology. Cryptography is involved with hashing and key-generation techniques. Verification checks and block formation may connect the two first steps or the central ones. More precisely, (i) transactions are signed once created (i.e., the *signing phase* in Section 1.2) and (ii) their signature authenticity is checked (i.e., the *verification phase* in Section 1.2) when collected in blocks ; this can be done before or after the propagation to the validating nodes. Signing and verification grant to blockchain the fundamental features of integrity, authenticity and non-repudiation mentioned in Section 1.2.5.

The block formation procedure can be an integral part of the validation step or a separate one depending on the blockchain nature. The validation process in blockchains is the expression of the distributed consensus on the transactions to be executed, and on their ordering. Hence, validators are all the peers involved from the moment in which the transaction is included in a block (or its outputs are collected in a block) upon its publication on the ledger. Peers collecting transactions (or transactions outputs) in blocks may not enter the validation phase. Any node of the network can build blocks to its liking. The possibility of subjecting the built blocks to the validation process (i.e., provide a *block-proposal*) can be entrusted to a restricted circle of peers (or even to a single one) denoted as *leading node(s)*. Leading nodes election procedure can be interwoven with the validation procedure or it can be completely separated. Permissioned blockchains adopt direct voting-based consensus protocols enabling a drastic separation between the *leader election* and the validation phase. Incentive-based consensus mechanisms admit the degeneration of the leaders' role into the validators' one – in order to be elected as leaders peers have to do the effort to validate the block they have constructed.

Due to the decentralized nature of blockchain technology, leading nodes, as validating ones, are likely changed once the proposed block of transactions is validated (as in the Bitcoin blockchain where leaders and validators are elected in a random fashion).

Comparisons of blockchains data-models can be found in [88; 89]. Authors survey in [90] the scripting language in both the UTXO and the account model (for Bitcoin and Ethereum). Regarding the transactions propagation, authors present in [91; 82] weaknesses of the propagation model adopted in Bitcoin-like networks and countermeasures to adopt for preventing any type of attack. In [30] authors present how the propagation model can be appropriately modified on the basis of specific strategic decisions. A list of the different block propagation mechanisms used in existing blockchains can be found in [39]. Concerning the different validation and consensus procedures adopted in blockchains few comprehensive works exists [37; 92; 93] ; one may find much more literature focusing respectively on permissionless [29] and permissioned [87; 36] environments. The following section presents the blockchain actors encountered by every blockchain transaction along the journey. Fig. 1.7 illustrates the transaction journey in its entirety : the four crucial steps, the intermediate steps and the key actors in the path. The proposed classification of the blockchain roles nodes can assume follows the logic proposed in [88] and surveys the different classifications presented in the white-papers of the major platforms (see Table 1.3).

1.3.5 Blockchain Actors and Corresponding Roles

We highlight in the following the key roles that blockchain nodes (with at least reading permissions) can assume.

Transacting parties : A blockchain transaction involves two different types of actors related to single or multiple blockchain users : the *data-sender* and the *data-receiver*. Interactions take place at address level : the *sending-address(es)* and the *receiving-address(es)* digitally track the data-flow (i.e., the transfer of digital assets) between the parties.

- *Data-Sender* : The data-sender is the node transferring data through an atomic operation (i.e, transaction) to a receiving node. The data-sender is not necessarily coinciding with (i) the transaction creator, (ii) the node with the right of initiating a data-transfer or, (iii) the data-holder [94]. Smart contracts involve the creation of a 'locked' transactions sequence that can be triggered by an authorized node (or even by a node outside the network) that may not be the owner of the transferred data. However, the data-sender is the one responsible for signing transactions (with its private key) in order to authenticate the origin of the object of the transfer (i.e., digital asset).
- *Data-Receiver* : Any user receiving a signed transaction that can : (i) recover the sender's public-key from the message and (ii) verify the transaction authenticity (i.e., transaction author and signature correspondence), is a data-receiver. Any blockchain node (user or contract account) can recover and verify the signature allowing tamper-proof transfers in the network.

Leading nodes : Agreement can be established by the election of a temporary *leader* node acting as a 'dictator'. The leader is responsible for both deciding which block to propose as a candidate to be included in the blockchain ledger and verifying the block proposal correctness. The leader goes out of power immediately after the validation of its block proposal. During its *round* (i.e., time interval where the leader has decisional power), the peer has no certainty that its block will be confirmed. Whenever a round expires, a new leader election starts.

The leaders election procedure is inherent in the agreement mechanism adopted by blockchain systems. Permissioned and permissionless blockchains adopt different methods to establish the peer in charge of proposing blocks to validators.

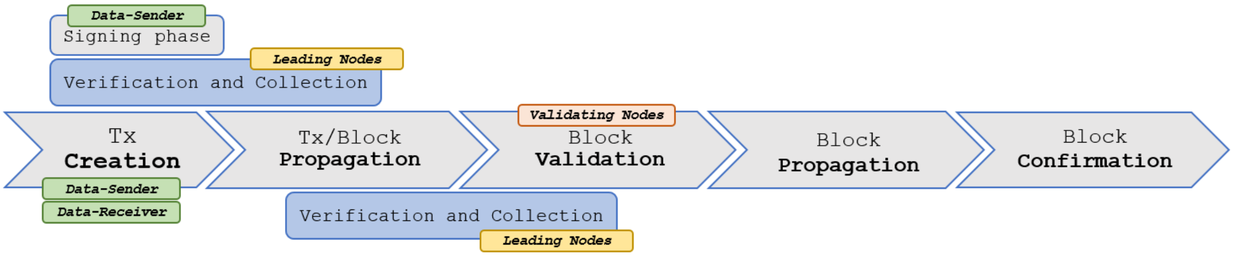


FIGURE 1.7 – The Transaction Journey. Once created the transaction is signed by the data-sender. Verification checks are performed upon block creation by the leading nodes. Transaction can be collected in a block either before being transmitted to the validating nodes or afterwards. The block of transactions is then validated, propagated and confirmed.

Validating nodes : As mentioned before, validating actors run the consensus algorithm and are responsible for establishing the agreement on the proposals made by the leading nodes. The validation of a block corresponds to the consensus among validating nodes on which block to publish and in which order.

Based on the journey of the transaction presented so far, it can be seen that it is nothing but the actors assuming the roles just described to characterize it. At the first stage the transaction meets (i) the transacting parties, namely data-sender and data-receiver ; the transaction is then transmitted to the (ii) leading peers responsible for verifying the correctness of the transactions, collecting them in blocks and proposing the block as a good candidate for the validation ; at the final stage, (iii) validating peers proceed with the validity attribution.

In permissioned environments, each actor has a different role with no overlap in the procedures of block proposal and validation. This is due to the scalable voting-based agreement procedure adopted in permissioned blockchains (see Section 1.4). Instead, open-access blockchains foresee overlapping roles for the mining nodes. Indeed, mining can be interpreted as a simulation of the leader election in traditional consensus protocols. Table 1.3 shows the different actors of widely adopted blockchain platforms[5] assuming the relevant roles previously presented.

A blockchain transaction is intended to meet these three main actors, but not only them. Some permissioned blockchains improve their scalability by designating to other peers different tasks such as execution-verification checks, leader election and ordering (e.g. Hyperledger *endorsers* and *ordering service nodes* [69]).

Table 1.3 – Blockchain peers acting as ‘transacting parties’, ‘leaders’ and ‘validators’ in the different platforms.

Platform	**Senders-Receivers**	**Leaders**	**Validators**
Bitcoin [20]	Users/Clients	Miners	Miners
Ethereum [95]	Accounts	Miners	Miners
Hyperledger Fabric [96]	Clients	Ordering Services	Validating Peers
Corda [97]	Transacting parties	Transaction(s) issuer(s) only	
Tendermint [1]	Accounts	Virtual miners	Committee
Chain Core [67]	Users/Clients	Block Generators	Block Signers
Quorum [98]	Accounts	‘Makers’	‘Voters’

[5] The DLT platforms we refer to were the mainly adopted ones during the time period of the thesis i.e., 2017-2020

1.4 Agreement in Blockchains

1.4.1 Agreement in Multi-Agent Systems

The words “agreement” and “consensus” refer to the convergence to a common *interest*. Consensus is the task of getting multi-agent systems with interacting agents to achieve a common goal. Agents must reach an agreement regarding a certain interest (a value or an action, etc.) depending on their state.

Consensus ensures agents’ agreement on a single request, or a sequence of requests also referred to as *atomic broadcast* [99]. Evidently, in any consensus protocol there are two events : the *proposal* and the *decision*. What nodes propose and decide is the interest they aim to agree upon, that in applications is most of the time a numerical value.

Fault-tolerant protocols are designed to deal with a limited number of faulty agents. According to [100; 101; 102], consensus reliability to halting failures is ensured by the following properties :

— *Agreement* : every correct/honest node must agree on the same proposed value $\mathcal{V}$.

— *Validity* : if all nodes propose the same value $\mathcal{V}$, then all correct nodes decide $\mathcal{V}$.

— *Termination* : every correct node has to take a decision on a value $\mathcal{V}$.

Moreover, atomic broadcasts are reliable broadcasts satisfying the following property :

— *Total order* : if any correct node decides that value $\mathcal{V}_1$ comes before value $\mathcal{V}_2$, then every other correct node must order $\mathcal{V}_1$ and $\mathcal{V}_2$ at the same way.

Therefore atomic broadcasts are also known as *total order broadcasts* [103].

In [104; 36] authors grouped these properties in two classes : *liveness*, grouping validity and termination, and *safety* that incorporates the remaining properties. These properties are analyzed in [36] for atomic broadcasts characterized by a *broadcast* and a *deliver* event.

It is worth noticing that blockchain applications may rise additional properties that can appear more important than those above to the designer. For instance, authors in [92] compare protocols in terms of *network identity management*, *energy consumption* and *adversary tolerated power*. Authors in [37] make comparisons in terms of security and performance ; in particular, security is qualified in terms of *agreement* (i.e., the achievement of a consensus state) and the resistance to *transaction censorship* (i.e., the malicious behavior of suppressing transaction) and Denial of Service attacks [80] ; and performance is qualified in terms of throughput (i.e., the transaction agreement rate), scalability (i.e., the system capability to respond adequately to a growth in the number of nodes) and latency (i.e., the time elapsing between proposal and decision phases during the agreement process). In [36] we find a comparison based on *liveness* and *safety*, while in [105] the comparison is limited to permissioned blockchains. A complete contribution on BFT protocols for replicated systems is provided in [106] where algorithm performances are evaluated in terms of cryptography costs, workloads, network conditions and faults.

Eventually, in order to satisfy the desirable set of properties, an agreement or consensus protocol consists in a set of rules that each database transaction must respect. These rules, embedded in each blockchain node behavior implementation, are therefore application-dependent rules that can vary from system to system [107]. Therefore, agreement in blockchains is crucial since it characterizes the systems ensuring properties such as resilience and security that can be summarized by a desirable level of dependability [108; 109].

1.4.1.1 Dealing with Asynchronous Communications

Networks can be *synchronous*, *asynchronous* or *partially synchronous* [110; 111]. Dealing with synchronous network does not mean dealing with networks where nodes' communications are not delayed in time ; instead, it means considering message delays bounded by some value. In asynchronous networks, this upper bound does not exist or is flexible, as messages are supposed to be delayed arbitrarily. In partially synchronous networks, or *eventually synchronous* networks, asynchronous nodes present time windows where they behave synchronously. Partial synchrony offers a good adaptability to the real network behavior and, at the same time, simplifies network modeling. Both liveness and safety properties are guaranteed during synchronous periods. On the other hand, during periods of asynchrony liveness cannot be ensured as proven by the "impossibility theorem" [112] stating that deterministic protocols do not reach agreement in a fully asynchronous environment.

In order to overcome this limitation, fully synchronous networks opt for relaxing the deterministic constraint ; they introduce randomness by requiring probabilistic termination (i.e., it is improbable for non-terminating executions to collectively occur) [113]. Authors in [114] proposed cryptographic solutions with *computational bounded adversary* (see Appendix 1.4.1.3) to overcome it. In partially synchronous networks, protocols correctly terminate during synchronous phases while they may stall during asynchronous ones, however termination is guaranteed under proper trust assumptions. More precisely, in order to preserve safety and liveness properties, this kind of protocols have to meet specific assumptions on the type and the number of faulty nodes in the network. In particular, fault-tolerant protocols typically work with a number n of nodes (replicas) exceeding twice the number of crashing nodes t and three times the number of Byzantine nodes b.

1.4.1.2 Dealing with Data Consistency and Agreement Finality

An important impact on agreement has the "CAP" (Consistency, Availability, Partitioning) theorem [115; 116] stating that fault-tolerant distributed systems cannot guarantee at the same time full data consistency (i.e., the ability to have nodes storing the latest data version at the same time) and, complete failure independence (or high availability) in presence of a partition.

It is worth recalling that consensus implementation is a mean for transaction validation and systems' resilience to failures. However, availability comes at the expense of consistency [41] whenever a network partition or failure happens. Thus, in general blockchain based systems aim at maintaining *eventual consistency*, i.e., consistency with time lags : all nodes get eventually a consistent view on the shared data, and in the convergence period upon each given change intermediate decisions may be taken, but eventually corrected based on the consistent store. Eventually consistent systems provide probabilistic agreement finality while consistent systems guarantee absolute finality.

1.4.1.3 Integrating Failure Conditions

Summing up, each agreement protocol is characterized both by a *communication model* and a *failure model* which in turn is characterized by *trust assumptions*. Communications among nodes can be synchronous, asynchronous or can lie between the two cases. Failures may be of two types (crash and byzantine) and can characterize a certain number of nodes. Crash failures – where honest nodes may fail – must be distinguished from Byzantine failures – where nodes may act maliciously. Of the two types of failures, the Byzantine class involves several failure subtypes [117; 118; 119], which are far more disruptive than classical crash failures. More precisely, protocols in partially synchronous environments tolerate a number $t < n/2$ of crashing nodes and a number $b < n/3$ of byzantine nodes. Liveness and safety in synchronous or partially synchronous environments are guaranteed for those protocols working with $n \geq 3f + 1$ replicas, where f denotes the number of faulty nodes in general.

In blockchains, properties and features result from a clever choice and implementation of a consensus protocol.

Consensus protocols, aiming at reaching an agreement state in the networks, satisfy their desired features and properties (such as liveness and safety) under some conditions. These are the so called trust assumptions characterizing the failure model of a protocol. These models are typically presenting bounds/threshold on the gap between two parameters referring to honest and malicious nodes respectively. Therefore, they are known in literature as "threshold adversary models" [104; 120]. The typical failure model foresees a threshold on the total number of nodes an adversary can control (f) with respect to the total number of nodes in the network (n). The threshold choice depends on the failure type and is between the half and a third (as previously met). However, this failure model presupposes knowledge of the number of parties involved in the network. Therefore, this classical adversary model works for permissioned networks where parties joining the system follows a specific membership protocol.

Bitcoin and other PoW-based cryptocurrencies consider an alternative failure model bounding no more the number of nodes but the work they may do. More precisely, the *computational threshold adversary model* limits the total amount of computational power that the adversary control (f_c) with respect to the total computational power (n_c). In order to guarantee double-spending resilience Bitcoin selects a threshold of a minority $n_c > 2f_c$, namely the adversary can control a minority of computational power. Bounding computational power does not require knowledge on participating parties, therefore the model well adapts to PoW-based permissionless networks, where anyone can join the system.

Further adversary models can be found in literature ; a new approach is the one of bounding the adversary *stake* (i.e., participation in a finite limited resource) [121], another option may be to adopt a game theoretical approach and therefore bounding adversary utility [122; 123].

1.4.2 Consensus in Distributed Systems and Blockchains

Agreement problems see abundant applications in complex systems dynamics [124] as well as in computer science and communications [125]. In such systems, consensus protocols must deal with dynamic agents that may fail during the agreement process.

The *two phase commit* (2PC) protocol [126], proposed in 1978, enables transaction processing in a distributed environment where nodes can atomically commit transactions trough *pre-commit* and *validation* phases. However, with 2PC any node failure compromises the consensus procedure. In this context, fault-tolerance (see Section 1.4.1.3) is defined as a property such that the system continues operating properly in the event of both process and communication failures caused by both honest nodes (i.e, crash failures) and nodes that act maliciously (i.e, Byzantine failures).

The *state machine replication* (SMR) technique [127] enables the construction of fault-resilient consensus protocols ; robust against crash failures in trusted environments (e.g., Paxos and RAFT [128; 129]) and additionally capable of tolerating Byzantine failures in networks of untrusted parties (e.g., BFT). Any computation is considered as a state machine mutating its state through request receiving. In a distributed environment, state machines are replicated and executed across multiple nodes. Though they do not evolve simultaneously, they have to agree on a common sequence of requests (state transformations) they are going to accept in order to have consistent replicas. A popular class of state-machine replication protocol is the one of *Byzantine Fault Tolerant* (BFT) protocols [86; 130].

We developed in Section 1.4.1.1 desirable behaviors with respect to asynchronous communications and data consistency guarantees, while recalling the strong relationship of these aspects with fault tolerance and the fact that in blockchain the consensus needed is about both on the elements of the ledger and their order.

The first approaches to consensus in distributed databases (2PC, atomic broadcast, SMR, BFT) can be considered as the predecessors of consensus solutions for DLTs. First generation blockchains (e.g., Bitcoin, Litecoin, Ethereum) establish consensus among millions of users in a probabilistic manner [33] thus, eventual consistency [131] took over from the initial need to maintain a coherent view of the system among participants. Failure-resilience characterize the systems as long as malicious nodes remain a minority in the P2P network (see Section 1.4.1.3). The idea is to introduce computational costs – to find a proof-of-work that validates a block of transactions – for charging peers who deviate from the default behavior (e.g., Bitcoin adopts previous approaches for fighting email spam [132] and preventing *Sybil attack* [133]). With the increase in popularity for cryptocurrencies, scalability and performance requirements changed significantly. Weaknesses of first generation blockchains led to a deeper analysis of the underlying technology through the lens of distributed computing. At a closer look PoW consensus procedure with its limited scalability and hight latency wastes too much computational resources. Appropriate amendments to the PoW procedure can guarantee challenging scalability levels without energy waste.

Several alternatives to PoW were proposed in order to compensate for its complexity and scalability issues. The idea was to replace the wasteful computations characterizing the PoW consensus with alternative proofs of a performed effort in validating transactions. PoW consensus together with protocols characterized by an effort-based leader election form the class of *proof-of-X* (PoX) consensus algorithms as defined by *Tschorsch and Scheuermann* in [29].

1.4.2.1 Proof-of-X Consensus

PoX protocols are designed for permissionless blockchains and relay on a probabilistic leader election process. In permissionless environments every node has the chance to become a leader simply proving that it made some "effort". The latter may have a computational, a monetary, or a storage nature or it may be an effort to assert itself on the blockchain network. The elected leader maintains his voting role till the new election's results are available. In the following we detail the PoW consensus, the PoS algorithm and, the PoS variations involving virtual mining.

Proof-of-Work.

The idea behind a PoW protocol is to make validation tasks difficult to perform, but trivial to verify. This idea was first presented as a solution to the email-spamming issue [134] and applied in a system called Hashcash [135]. The email sender should solve a cryptopuzzle finding the hash of a string, containing all the necessary information of the receiver, which has to meet a certain target. The usage of the Secure Hash Algorithm (SHA) [136], mapping data of arbitrary length to data of a fixed length in a non-invertible way, ensures a costly procedure to find a valid hash. B-money [123] suggested, in 1998, a PoW procedure where the computational effort can be easily quantified in terms of commodities baskets. At the same time, a PoW-based decentralized digital currency called Bit Gold was proposed [137] such that nodes should generate strings of bits using one-way functions with a cost expressed in number of compute cycles. The last Bitcoin's precursor, RPOW [138], incorporates the hashcash scheme creating Reusable PoW (RPoW) tokens. Bitcoin, as its precursors, uses a computational hard validation procedure to create rare and valuable goods. The real contribution brought by the system is the combination of decentralization, double-spending resistance, Sybil resistance and trustless node management with the “block-chain” architecture.

The PoW protocol consists in a race among nodes to be the winner and therefore gaining a reward of new minted tokens. The competition takes place among particular nodes, called *miners*, aiming at producing a valid PoW consisting in the hash value computation of a previous block header. In order to validate a block, the computed hash should meet a precise difficulty requirement. The nature of the problem relates the mining procedure to a lottery race where the validation process is completely aleatory and the probability of finding a valid hash is proportional to miners’ computing power. Once the winner is found it acts as a leader node attaching to the blockchain its selected block of transactions. Its epoch expires with a new valid block, thus a new winner of the mining race. Bitcoin consensus provides for the coincidence of both validator and leader roles in a single node. In general, PoW blockchains may separate the leader election (mining/transaction validation) form the transaction ordering procedure (i.e., Bitcoin-NG [23]).

Strong consistency would ensure a single chain of valid blocks published on the ledger. A PoW mechanism, however, guarantees consistency on a probabilistic form (forms of eventual consistency [20; 104; 139; 140]) since *forks* may occur. Whenever two blocks are validated approximately at the same time, or the network latency is delaying the transmission of a valid solution to the network, the result is the presence of two valid chains with the same block number. This inconsistent situation is solved with the validation of a new block through the *longest chain* rule : the chain with the most blocks is considered as the valid one, noting that the chain related to the greatest PoW effort may not be the longest chain [141]. The rule is proposed as a probabilistic solution to the Byzantine Generals problem [86]. Other variants of the longest chain rule were proposed in order to scale PoW blockchains : GHOST [142] proposed the *heaviest chain* rule that is confirming the block in the chain with the highest aggregate difficulty level, i.e., with the greatest computational load involved.

The economic incentives [143] resulting from the mining procedure induce miners to reduce the validation costs in order to maximize their earnings. Over the years the democratic idea pushed by Bitcoin of one-CPU-one-vote has left room for a centralizing trend in the validation process with a decreasing number of active solo miners and the formation of powerful coalitions of miners, *mining pool*, showing practical advantages but also motivating opportunistic pool-hopping behaviors [15]. Centralization in a permissionless environment results in increased vulnerability to double-spending attack. Decentralization is a characterizing feature for blockchain based cryptocurrency, one may argue that pool formation is nothing more than a converging trend to the original banking system [144]. An approach to face this monopoly trend is the inclusion of memory-access operations in the PoW computations accompanied by memory-bound functions. However, these schemes cannot make this centralization trend disappear since it requires specialized mining equipment and thus benefits from miners cooperation, as the original PoW (i.e., Litecoin [145; 146]).

Mining devices are constructed to compute hash values as fast as possible. The Bitcoin system was conceived for a CPU mining that was quickly replaced by a GPU (Graphic Processing Unit) mining. GPUs can perform hash computations in a more efficient way with respect to classical CPUs, therefore general Altcoins started adopting GPU mining at the end of 2010. This results in faster operations, due to operations parallelizing [147] and in energy savings [148]. When hardware based mining solutions took over the computing power dedicated in mining activities experienced, despite strong fluctuations, an exponential growth [149]. It worth nothing that alternative PoW-schemes try to compensate the incredible waste of energy with useful work at an academic level ; *Primecoin* [150] searches for prime numbers chains (Cunningham chain [151]), *NooShare* [152] executes Monte-Carlo simulations, *Shoker* [153] proposes matrix-product problems to solve while in [154] authors propose to replace PoW hashing function with alternative one-way functions satisfying additional properties.

Pseudo-random leader elections based on PoW schemes [155] are generally prone to *grinding attacks*. The practice consists in testing several candidate blocks improving in this way the possibility of being a leader in the following round. Hence the need of unbiased unpredictable random elections as those adopted in [156; 157]. The need of alternative PoX schemes (i) motivating the proof of "useful" efforts and (ii) improving performance [158] in terms of security, scalability and eco-friendliness is evident.

Proof-of-Stake and Virtual Mining Alternatives.

The PoS mechanism resumes the PoW one while passing from a real mining to a *virtual mining* (i.e., consumption-free mining). It replaces the PoW leader election based on mining, with an alternative approach depending on users' investments in the blockchain, i.e., their stake : the amount of virtual tokens held by a user ; in other words, the mining race costs are replaced by shares in the consensus. The probability of becoming a leader is proportional to one's stake ; once a leader is selected among stake-holders, it has the right of validating the preferred block. As for PoW, consensus finality is not met and the "*richest chain*" rule breaks deadlock points – the valid chain is the one with the highest total amount of stake involved. Hence PoS could avoid the centralization trends observed with Bitcoin. PoS-type algorithms differ in the (i) estimate of users' holding and, in the (ii) adopted incentive mechanisms.

Users' stake can be estimated as an amount of coins stored in an account. However, security and fairness issues [159] arise when considering this consensus configuration : leader election components are quite predictable, and a selection based solely on the amount of tokens held by users is unfair ("rich-get-richer"). Hence, alternative solutions were proposed to elect the leader taking into account its stake.

One of the first PoS variations consists in weighting a coin stake by its "age" (i.e., the time elapsing between the last movement of the coin). In *PeerCoin* [160] the *coin age* has the same role of the computational power for the classical PoW scheme. However, the real difference is to give all participants the chance to be elected, thus solving monopoly-like situations. Despite stake-based coins (e.g., PeerCoin and *Nextcoin* [161]) prevent centralization trends, their underlying protocols encourage amassing coins and stay inactive in the network – that exposes the network to Sybil and DoS attacks [83]. Thus, the ideas to punish coins accumulation trends (*proof-of-stake-velocity* [162]) and to assign the reward for the validated blocks only to the active users (*proof-of-activity* [155]). Active peers are the ones that solve a crypto-puzzle with a difficulty target depending on the users' stake, thus hash computing improves network security. Leading stake-holders, responsible for block validation, are therefore picked in a pseudo-random fashion.

In both *Ouroboros* [157] and *Snow White* [156] participants use pseudo-random function to predict the block-generator however, while the former takes into account only the stake distribution in the network, the latter additionally relies on a pre-image (nonce) calculation. More precisely, Snow White is an "hybrid" protocol cleverly mixing PoW (computing only one hash per round) and PoS (the hash should meet a target depending on user's stake). *Blackcoin* [163] and *Nova Coin* [164] are the first applications using this type of hybrid schemes (i.e. mixing different consensus mechanisms).

One of the latest variants of the PoS scheme was recently proposed by Ethereum. This is *Casper* [165] that is to be incorporated into the "Serenity" [166] version of the platform. Casper brings the PoS scheme closer to the traditional BFT model – more precisely, it combines the concepts of security deposits with voting in order to reach agreement. Peers have to make a security deposit in order to be elected as validating peers. The pseudo-random election takes into account the deposit entity made by the candidates and elect a set of validators. That is, Casper cannot be considered as an hybrid algorithm mixing PoS and BFT (see Section 1.4.2.3) since election and validation are not independent processes.

Concerning rewards distribution, PoS protocols originally distributed rewards among all peers regardless the elections results [157; 156] with the result of incentivizing the famous *nothing-at-stake* [167] attack. Today these naive implementations are overcame by valid alternatives : some [1; 165] asking validators to lock an amount of coins (*proof-of-deposit*), some [160] asking to destroy it (*proof-of-burn*), and some [168; 169; 170] asking to allocate a significant amount of memory/disk-space (*proof-of-capacity*) or to provide wireless network coverage (*proof-of-coverage*).

Efficient PoS alternatives based on virtual mining working for open-access blockchains with random leader election within untrusted nodes are the PoET (*proof-of-elapsed-time*) and the PoI (*proof-of-importance*) consensus schemes. The former adopts a trusted execution environment (TEE) in Intel SGX for the results verification [78] for guaranteeing both safety and randomness of the leader election. Peers make a request of *wait time* for processing the election procedure ; the winner of the lottery is the validator with the shortest waiting. Correctness of the election can be publicly verified within the TEE : leaders generate a proof testifying they had the shortest wait time and additionally, they prove that the block broadcast happened right after the waiting expiration. The platform NEM (New Economic Movement [171]) proposes a blockchain based on a peculiar block validation process (i.e., *harvesting*) and a PoI [41] consensus algorithm determining the user that create and append transactions block (i.e., *harvester*). NEM works with an underlying cryptocurrency (i.e., XEM) that characterizing the balance of each account on the network that is split in a *vested* and an *unvested* part. Eligible validating peers are evaluated according to the amount of vested XEM and the support their accounts give to the network (i.e., number of transaction partners and number and size of transactions in the last 30 days). Contrary to previous mechanisms, PoI does not incentivize peers to save their coins/resources increasing their voting power. Harvester candidates are incentivezed to be 'active' in the network.

PoS enables both public and private leader election thus, the consensus protocol is applicable by both blockchain with and without permissions. Restricted elections result in DoS resilience since leader in the epoch become known to the stake-holder community at first and then to th public. Moreover, permissions on block validation may be assigned in order to improve the efficiency of the system.

That is, stakeholders privately delegate a representative set of validating peers (*delegated proof-of-stake* DPoS [172]). The list of witnesses is shuffled at the end of each round in such a way that each validator can produce block according to a certain rate. Witnesses are paid out for each produced block.

1.4.2.2 BFT Algorithms

Traditional BFT protocols – resilient to both byzantine and crash failures – generally work under partial synchrony assumptions, bounded communication latency and a classical client-server architecture. Due to their nature (state machine replication protocols) properties of liveness and safety are guaranteed. Moreover, in BFT, both consensus *proposal* and consensus *decision* events are separated. The downside in these agreement protocol class is the communication complexity [173]. Hence, the necessity for closed-system adoption (i.e., permissioned blockchains).

The *Practical Byzantine Fault Tolerant* (PBFT) protocol [174] is a BFT variant that addresses the consensus problems for small systems, since agreement among n nodes is reached through the transmission of $O(n^2)$ messages ; it does so relying on a three phase round division where in each round a block is validated passing through a *pre-prepared*, *prepared* and *commit* steps. Each peer proposal access to the next phase only with the 2/3 network approval. Therefore, the algorithm requires at least $3f + 1$ honest replicas to tolerate f failing nodes. Recent PBFT variant *SIEVE* [175] introduce non-determinism in the chaincode execution handling transactions with occasionally different outputs. Moreover, an alternative PBFT-based consensus protocol recently proposed simplifies the traditional failure model for better efficiency levels. The idea behind *XFT* protocol [176] is to exploit the following assumption : *adversaries cannot control the majority of the nodes* $n > 2f$. In this way the crash fault tolerant protocol avoids considering byzantine failures.

With the arrival of consortium blockchains, the BFT protocol (popular in the financial sector) was amended to support open reading rights (public). *Stellar Consensus Protocol* (SCP [77]) is a BFT-variant based on permissions to choose a pool of known participants to trust. Participation to this pool (*quorum*) is open and global consensus is reached intersecting all the chosen quorums. In the same way, in *delegated BFT* protocols [177] only a class of representative peers comes to vote. The most popular BFT-open protocol adopting trusted subnetwork in the block validation process is *Ripple* [76]. It make use of *unique node lists* (UNLs) playing the same role as the Stellar quorums.

The main characteristic of the protocol is that agreement is reached when the 80% of the nodes vote for the same candidate block, this result in low adversary power tolerance. The recent BFT variant, *proof-of-authority* (PoA) [178], relies on a set of trusted nodes (authorities) with a rotating leader. PoA algorithms [179; 180] ensure better performance with respect to PBFT consensus since working with less message exchanges (i.e., 1-2 message rounds to commit a block).

Classical BFT scalability drawbacks, regarding the number of nodes participating in the consensus, have been solved with hybrid consensus protocols appropriately mixing PoX with BFT algorithms used in permissioned environments. This mix results in *committee* formation driving the consensus process replacing the original leader role. Hybrid models contemplate the usage of two different consensus procedures ; one to form the committee and another one to establish consensus among the nodes inside the community. Note that, however, by "hybrid" we do not mean any committee-based consensus procedures (e.g., Hyperledger utilizes PBFT) ; hybrid algorithms are the ones mixing two different consensus schemes. In order to differentiate those hybrid schemes running classical BFT protocol – to the ones that make use only of PoX procedures – we denote them as *hybrid BFT-based* algorithms.

Nowadays, it is possible to find blockchains not requiring global consensus where each node has its own hash chain containing only the transactions where a user is involved. *Cong* proposed in [181] a system where agreement is established on special blocks representing a set of transactions. These systems can reach full *horizontal scalability* (i.e., scalability in the number of nodes) at the expense of robustness.

1.4.2.3 Hybrid BFT-based Algorithms

Hybrid consensus mechanisms are born with the intent of preserving permissionless consensus but overcoming the trade-off between scalability and performance. Standard PoX consensus has to be improved by combining it with parts of BFT-based permissioned consensus mechanism. The idea of dividing the agreement process into different parts (see Section 1.3), initially proposed by private blockchains such as Hyperledger, is the key to built scalable permissionless protocols providing consensus finality. The assignment of tasks to the nodes is carried out by means of a committee-formation ; consensus is driven by a community of nodes that build blocks at a first stage and then come to vote for their validity.

At first, the committee is formed, which then will agree on the validation of a block. Membership of the committee is open to all nodes in the blockchain ; they acquire voting rights for the second phase through a PoX scheme. Existing hybrid algorithms involve PoW and PoS procedures to establish the leading nodes in the committee responsible for validating blocks. The idea of joining a committee through a PoW procedure is to assign voting power to each participant in proportion to their computational strength ; this is the case of *ByzCoin* [182] and *PeerCensus* [183] where Bitcoin meet strong-consistency. Committee formation through PoX schemes is a dynamic process ; participants receive a share of the committee through real or virtual mining. *Tendermint* [184] is the most popular protocol where Bitcoin PoW protocol is replaced with a PoS scheme that is, virtual mining. For Tendermint and other less known protocols [185; 186; 187] random committee selection is (can be) replaced by an assessment of the amount of tokens held by the blockchain nodes.

The right combination of PoX and BFT algorithms significantly improves the blockchain performance ; however, scalability and throughput are not positively affected with a huge single-committee. Therefore, blockchains may adopt a consensus procedure based on multiple committee, also known as *sharding* [37]. In this way transactions can be processed in parallel by different *shards* (i.e., committees) of few nodes since their size is inversely proportional to the achieved performance level.

1.4.3 Summary of Consensus Mechanisms and Their Evolution

The diagram in Fig.1.4 summarizes the evolution of the procedures to reach consensus in distributed systems, starting from the classic pre-blockchain algorithms - (i) Classic consensus - passing through the early blockchain consensus - (ii) Proof-of-X and (iii) Hybrid consensus - and, ending with the consortium solutions widely used today - (iv) Consortium BFT consensus and (v) Hybrid BFT-based. We have highlighted five main classes of consensus and characterized (where possible) the different variants. We consistently cite the main algorithms representing the consensus classes, encountered in the previous discussion.

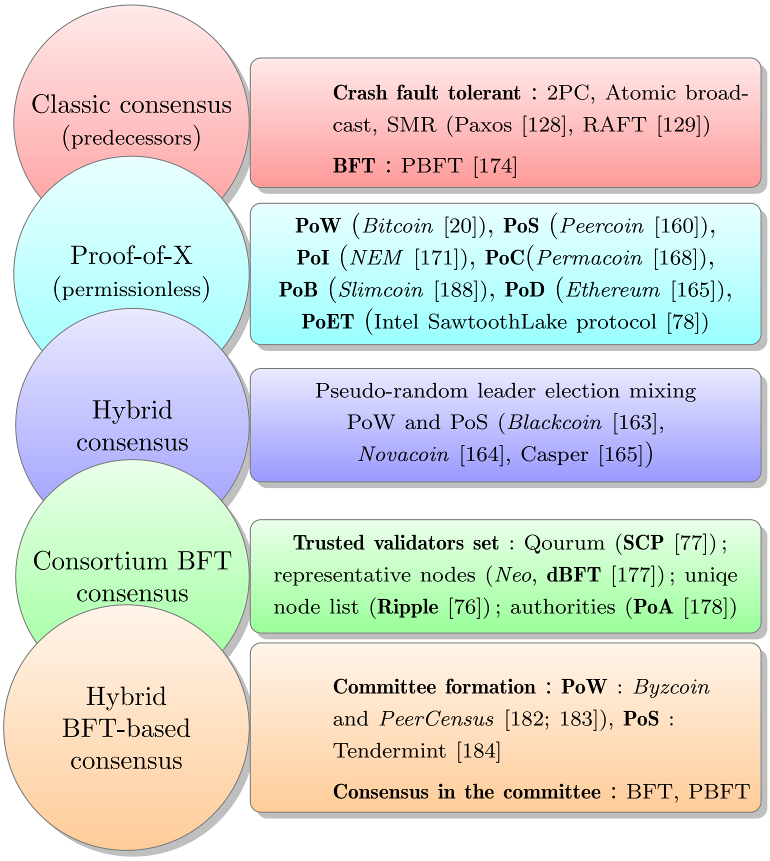

Classic consensus (predecessors)
Crash fault tolerant : 2PC, Atomic broadcast, SMR (Paxos [128], RAFT [129])
BFT : PBFT [174]
Proof-of-X (permissionless)
PoW (Bitcoin [20]), PoS (Peercoin [160]), PoI (NEM [171]), PoC(Permacoin [168]), PoB (Slimcoin [188]), PoD (Ethereum [165]), PoET (Intel SawtoothLake protocol [78])
Hybrid consensus
Pseudo-random leader election mixing PoW and PoS (Blackcoin [163], Novacoin [164], Casper [165])
Consortium BFT consensus
Trusted validators set : Qourum (SCP [77]) ; representative nodes (Neo, dBFT [177]) ; uniqe node list (Ripple [76]) ; authorities (PoA [178])
Hybrid BFT-based consensus
Committee formation : PoW : Byzcoin and PeerCensus [182; 183]), PoS : Tendermint [184]
Consensus in the committee : BFT, PBFT

FIGURE 1.8 – Evolutionary route of consensus protocols in five classes from pre-blockchain to post-blockchain protocols

1.4.4 Comparison between Blockchain Consensus Protocols

Previous sections presented the problem of reaching consensus in a distributed system. Traditional consensus protocols have opened the way to PoX-type mechanisms and then reconsidered in permissioned blockchains fro their performances. *Vukolic* [92] work is one of the first at addressing a comparative analysis on the different consensus procedures however, it focuses only on the PoW-based algorithm and traditional BFT scheme. Recent works [93; 36; 37; 35; 189] compare different agreement protocols in terms of (i) *node identity management*, (ii) *energy saving*, (iii) *tolerated power of adversary*, (iv) *transaction finality*, (v) *communication complexity*, (vi) *nodes scalability*, (vii) *throughput* and, (viii) *latency level.*

Table 1.4 and Table 1.5 summarize these comparative studies. The data shows the tendency to implement safer and high-performance (1000 tps) blockchain-based systems with low energy impact and low latency, that reach a final agreement with the guarantee that the validated blocks will not be discarded. It can be deduced that further work needs to be done regarding the message overhead between the consensus participants (n in Table 1.4) and Table 1.5).

TABLE 1.4 – Summary about consensus mechanisms : comparative analysis of PoX consensus algorithms

Property	**PoW**	**PoS**	**DPoS**	**PoET**	**PoI**
Participation mode	permissionless	both cases	both cases	both cases	both cases
Energy saving	no	partial	partial	partial	yes
Tolerated adversary	$< 25\%$ power	$< 51\%$ stake	$< 51\%$ peers	TEE	$< 50\%$ importance
Consensus finality	no	no	no	no	no
Message overhead	$O(1)$	$O(1)$	$O(1) - O(n)$	$O(1)$	$O(1)$
Node scalability	> 1000	> 1000	> 1000	> 1000	> 1000
Throughput (tps)	7-30	100-200	millions	1000	4000
Latency (s)	up to 600	up to 600	unknown	unknown	unknown

TABLE 1.5 – Summary about consensus mechanisms : comparative analysis of BFT and BFT-based consensus algorithms

Property	**PBFT & variants**	**Consortium BFT**	**Hybrid BFT-based**
Participation mode	permissioned	permissioned	both cases
Energy saving	yes	yes	yes
Tolerated adversary	$< 33.3\%$ replicas	variable (20%-33.3%)	$< 33.3\%$ replicas
Consensus finality	yes	yes	yes
Message overhead	$O(n^2)$	$O(n^2)$	$O(n) - O(n^2)$
Node scalability	< 100	$100 - 1000$	$100 - 1000$
Throughput (tps)	up to 110k	up to 10k	1000
Latency (s)	less than 1	less than 1	up to 20

1.5 Blockchain Implementation

Leveraging on the important background presented in the previous sections, this section is meant to start our blockchain vademecum [6], to give to the reader a tutorial about when to use blockchain, which solution to use, and how to use it, based on use-case requirements.

During the past few years, research societies along with industrial and governmental institutions intensively worked on DLT and blockchain, trying to understand better this paradigm and its place in today's market. This resulted in many publications and standardization activities as well. In the following, we provide the reader with a decision model to understand *When* to use the blockchain technology (Section 1.5.1) and *Which* type of blockchain suits a certain use case best (Section 1.5.2). The decision model is characterized by two decision paths (When and Which paths) that can be traversed either consecutively or independently ; the decision points can be both direct questions or trade-off points [7]. In the following, we use Fig. 5.19 as a support for the When and the Which questions in Sections 1.5.1 and 1.5.2.

General Purpose Reading List.

In developing this tutorial we made use of a broad spectrum of documents going beyond academic literature, and including books, white-papers, technical reports, blockchain forums, discussion papers, and online encyclopaedias. We concentrated on works showing real applications of blockchain in the industry going beyond the well-known digital payment systems proposed by cryptocurrencies. The main investigated areas were : (i) finance, (ii) security-and-privacy, (iii) public, (iv) Internet-of-Things (IoT), (*v*) smart business. We report such reference works in Table 1.6. In addition, our reading list includes works investigating when a blockchain can revolutionize a business [190; 191; 192], benefits and drawbacks of both permissioned and permissionless blockchains [92; 105; 193; 74; 43], and links with traditional solutions [194; 195; 196; 197; 198].

[6] 'Vademecum' is a term that may not be well-known by the reader. It derives from the latin expression 'Vade Mecum', literally meaning 'go with me'. It refers to a synthetic collection of information concerning a specific field or technique (blockchain in our case), having the goal to provide the reader with quick and concise responses on the different details of the specific field or technique.

[7] Trade-off points represent situations that involve a choice between two or more aspects, where the loss of value for one aspect constitutes an increase in value for the other one(s). In the proposed decision tree alternatives are (i) blockchain or traditional database features for Section 1.5.1 and, (ii) permissionless or permissioned blockchain features for Section 1.5.2.

TABLE 1.6 – Reading list on blockchain application domains

(i)	clearing, collateralization, real estate [199; 200; 201; 202; 203].
(ii)	personal data-management [204; 94].
(iii)	energy [205; 206; 207], health-care [208; 209; 210; 211; 212; 213; 214; 215; 216].
(iv)	storage, authentication, e-commerce [217; 218; 219; 220] communications & networking [221; 222; 223].
(v)	supply chain [224; 225; 226], transportation [227; 228].

1.5.1 When to Use Blockchain ?

This section focuses on the first general question of the vademecum : when to use blockchain as a technology ? Our use-case oriented answer to the When question is given passing through the following direct questions and trade-off points (see Fig. 5.19). The vademecum aims to provide an answer for any use-case questioning whether the blockchain represents a good business solution.

(1) Do you need to store and share a ledger state ?
We start from a situation where a ledger database is required i.e., data in transaction form needs to be stored and shared. Data constitute the ledger state, which is subject to updates that must be shared over the network. Whenever it is not needed to share a stored state, complex cryptographically-based architectures result unnecessary for simply letting stored data to be accessible. Therefore, in the presence of a negative answer blockchain is certainly not needed and traditional solutions are preferable.

(2) Are there multiple potential writers ?
The adoption of blockchains makes sense only when data need to be stored by multiple users and shared among them. Indeed, in a blockchain multiple users (not necessarily all network users) are supposed to have writing access and permission to participate in the procedure to establish consensus among parties. Blockchain lets business move from hierarchical client-server systems with locked writing rights to decentralized P2P interactions with multiple (if not all) nodes able to write to the distributed ledger.

(3) Who do you entrust with the ledger maintenance ?
Blockchain enables interactions among trustless actors circumventing any intervention by a central authority. The need for decentralized systems arises whenever network participants lose their trust on a (alternative or pre-existing) centralized system. However, the transition from a centralized to a

decentralized system is not necessarily radical ; blockchains can decentralize some functions while keeping others centralized. Blockchain has revolutionized the concept of 'trust', which is no more related to the identity of the actors in charge of the validation procedure, but it is related to the protocol architecture. Clients trust the technology that is forcing validators to follow the protocol punishing or making unfeasible any possible deviation. For such a key strategic question on the trust, we can spot three possible types of answers :

a) An external third party : the system maintenance is entrusted to an external entity which in case of failure could be switched. In such a case, designers should opt for a centralized architecture that is easy to deploy and maintain by the trusted third party.

b) A group of selected actors : nodes in charge of updating the ledger participate to the system. Their identity can be known or unknown, however, the methods for selecting these nodes and the targeted activities are important aspects. Indeed, the class of partially-centralized systems includes a spectrum of possibilities such as adopting private distributed ledger, creating *consensus committee* [37], and structuring the communication with external trusted systems [229]. Instead of providing open-access to anyone, blockchains can bind certain of their functionalities (read and write) arranging *permissions*. We may therefore have an escalation of permissions, from the single permission to read the transaction log to the ability of validating transactions. At first, permissioned blockchains select participants with network access controls ; their identity must be known. Then, permissions are given to implement any type of change to the data registry ; different trust levels can be associated with different nodes' roles (see Section 1.3). Moreover, whenever the validation of a transaction is linked to an external variable realization, one may choose whether to trust or not the actor designated to communicate with the outside.

   Regardless of any restrictions on the node roles, once decided to trust a restricted entourage for the validation process, one may wander which actor to entrust the verification of the operation correctness. Let us recall that block verification consists in a repeated check of both the chained blocks integrity and authenticity – carried out in most cases by the validators themselves – and the chained blocks validity. Blockchain transparency allows any network participant to verify whether a published block was validated according to the protocol since all network nodes have the same view on the log. On the other hand, verification checks are entrusted to a central authority whenever participants differ in the view they have of the ledger. Thus, the next question at this point is :

(3.b) Do you need the ledger to be publicly verifiable ?
Whenever a system requires public verifiability, one may keep restrictions on writing rights but at the same time leave the freedom to everyone to observe the system state – as for open-permissioned blockchains. For those cases in which verification checks may not be in the public domain, the choice between a private blockchain (full-permissioned) and a traditional solution is clearly linked to the nature of the verifier(s). Verifying peers coincide with the so-called validating peers in a private environment where transactions validation is performed by trusted parties. The choice now is between a *centralized verifier* – leading to the adoption of a traditional central database where the group of trusted nodes organize themselves in a central authority (with both reading a writing rights) representing however, a potential single point of failure – and a *distributed verifier* – consisting in several trusted validators known to the network operating in a P2P framework where all the participants in the system may connect to each other. The adoption of a blockchain (permissioned in this case) rather than a traditional solution is dominated by trade-offs regarding mainly the impact on the throughput, the costs, the presence of the basic blockchain features, the failure resistance level and the adaptability to different business cases.

Trade-off (3.b) performance, cost efficiency and adaptability VS blockchain features and failure resistance
Traditional centralized databases are widely used both for their simple architecture – easy to adapt to each use case and often affordable as the data is stored and maintained from a single central computing node – and, for the speed and ease in updating the data they manage – every change is managed by the central authority and immediately communicated to users [12]. In fact, the central authority can easily modify data with CRUD (Create, Read, Update, and Delete) commands. Thus, the technology strengths consist in high levels of performance (in terms of transaction processing rate), low costs in adopting the technology (in terms of design and management cost, as conventional softwares are cheaper than blockchain solutions) and high degree of adaptability in managing any type of data and its use.

Despite the countless advances made by blockchain technologies to reach higher levels of scalability, throughput and latency, blockchain will likely always be less performing than a centralized database. This is because processing any change in a distributed system – through transactions – requires additional efforts consisting in : (i) applying and verifying the digital signature, (ii) agreeing on a unique vision of the data ledger, (iii) replicating data across the network and, (iv) updating the ledger only with *write*-operations. In blockchain the idea is that the validating nodes independently process transactions and then at a second stage compare the obtained results with the rest of the network until they come to an agreement. However, blockchain offers, at the same time, the six important features presented in Section 1.2.5 (decentralization, immutability, confidentiality, integrity, authenticity and transparency), that are absent (in their entirety) in traditional databases. In addition, since blockchain is first and foremost a distributed ledger, it is robust against node failures [8]. Adopting or not blockchain is therefore a matter of which set of quality properties to privilege between (i) performance, cost efficiency and adaptability and, (ii) blockchain fundamental features and failure resistance.

c) The public community : Whenever trust cannot be laid on a set of network nodes, it is better to have confidence in a protocol (i.e., a set of rules) that guarantees the correct functioning of a system maintained by the public community. Permissionless blockchains enable untrusted parties to interact without relying on any *man-in-the-middle* (i.e., disintermediation). Transaction history is fully transparent to everyone. Validation and verification are carried out in a fully open and distributed fashion ; any network node can participate in the process possibly remaining pseudonymous.

1.5.2 Which Blockchain to Use ?

Thanks to the attractive blockchain properties (Section 1.2.5), the development community has worked hard to broaden its range of applicability. At this point, the vademecum suggests to apply blockchain also to multi-access shared ledger situation such that there is a circle of trust, and concessions in terms of performance, cost efficiency and adaptability can be acceptable.

Permissionless blockchains require users to direct their trust towards cryptography and related mathematics, while permissioned ones ask for confidence in few (or all) nodes of the network. Therefore, given that blockchain is the right technology after the When question, at this stage the first question

[8] However, it should be noted that for permissioned blockchains any centralized procedure (such as validation, verification or external communication) can be considered as a single point of failure.

the designer may wander is in which of the two categories falls its use-case. In addition, if directed to a permissioned blockchain one may choose whether or not to put restrictions on data ledger access.

The vademecum chart in Fig. 5.19 can now be read from the bottom to the top.

(4) Which is the blockchain primary adoption ?

Blockchain can be primarily adopted as (i) a *system of records* (SOR) and as a (ii) *platform*. Polarization toward the former or the latter application class is important to characterize the blockchain nature.

a) Blockchain as a system of records

SOR's principal goal is storing data and wisely processing it in order to re-present to users the history of data. Blockchain constitutes an innovative solution to track the history of information modifications that is characterized by interesting features, including its transparency. The question now is which blockchain solution between a permissionless and a permissioned one is best for a SOR. Firstly, one should realize if there are disclosure issues. Once understood the desired privacy level (between anonymity and confidentiality), the choice is a matter of trade-offs ; high performance comes at a cost.

(4.A) Is confidentiality[9] *required ?* Privacy and confidentiality within blockchains are controversial ; what permissionless blockchains can hide to the network is the users' identity only, conversely, every operation performed in the network is in the public domain. Hence, permissionless blockchains guarantee users some degree of anonymity (pseudonymity) without offering any confidentiality in transacting on the blockchain. On the other hand, private blockchains (with restrictions on both writing and reading operations - and where participants are known in the network) can ensure that 'what happens in the network remains in the network'. Therefore, if operations are not to be disclosed to the public, the most appropriate solution is a blockchain that is not accessible to everyone, i.e., a full-permissioned blockchain ; otherwise, the following trade-off allows discriminating among a permissionless blockchain and a permissioned one.

Trade-off (4.A) performance VS cost efficiency : In the absence of confidentiality constraints, one should concern about the importance of performance over cost efficiency. In order to achieve a processing rate of the order of thousands tps, the classical permissionless blockchain structure must

[9] We mean by the term 'confidentiality' the non-disclosure to the public of the operations performed by blockchain users.

be abandoned. Blocks of transactions should no longer be processed one at a time ; blockchain needs to adopt an architecture favoring the processing of multiple blocks in parallel. These result in a more complex technology structure with high design costs. Permissioned blockchains (both open-permissioned and full-permissioned) offer good performance due to their restricted nature where data validation, verification, replication and modification are faster with respect to a public environment. Thus, whenever priority is given to the throughput, the best choice is in favor of permissioned solutions (both full-permissioned and open-permissioned).

(4.A.i) Which is the performance level required ? If it is required to have performance comparable to that of a centralized system, a possible solution is to store data (i) *off-chain* or (ii) on-chain via smart contracts. Blockchain initial aim was to enable data-storing on-chain ; however the kind of data stored was the transaction history. In the Bitcoin blockchain external data was initially stored on the ledger through unofficial transaction manipulation (e.g., writing in a coinbase transaction or using a fake account address) discovered and disseminated by avid network users [230]. Due to the limited space provided by the OP-RETURN, second-generation blockchains proposed alternative solutions based on smart contracts and off-chain solutions. Data can be included in a smart contract at variable or event level directly on-chain (on a blockchain – no matter the nature – supporting smart contracts), however performance (up to thousands tps) is not still comparable with the one offered by traditional databases (e.g. Multichain early versions [231]). Off-chains solutions are the best in terms of performance ; raw-data are stored off-chain, while it is possible to handle meta-data or hashed-data on-chain as a complementary storage (e.g., Swarm [232] and Filecoin [233]). However, the ease of communication between the two technologies heavily depends on the type of blockchain and the corresponding off-chain solution chosen. The ideal off-chain storage is a private cloud attached to the corresponding blockchain, thus a full-permissioned structure (e.g., Microsoft Cryptlet Fabric [234]).

b) Blockchain as a platform

In general a blockchain-based system enables digital data-sharing, digital data-storing and virtual interactions among peers. The principal goal of a blockchain platform is to form P2P digital relationships favoring digital exchanges and business automatization.

(4.B) Which is the platform primary purpose ? The central question relies on the platform primary purpose between the following fundamental categories :

i) Asset digital exchange : Blockchain enables the sharing of any valuable data (i.e., asset) among parties without any geographical and timing constraint. Both the asset nature and the size of the data-flow impact on the choice of the blockchain nature and its architectural design.

(4.B.i) Which is the asset nature ? Assets could be *sensible data* that have to be managed restricting access to the record – full-permissioned blockchains. If no disclosure issue occurs, the quest of adopting or not permissions in writing rights merely depends on trade-offs : for better performance than that offered by Bitcoin-like blockchains one should pay the price of not guaranteeing full transparency (auditability) and equal rights of participation.

Trade-off (4.B.i) performance VS blockchain features : The choice whether to give priority to the basic blockchain features rather than to the performance is strictly linked to the nature of the exchanged assets in the network. To give the reader an idea, let us take the case of *tokens.* Blockchain became popular thanks to assets *tokenization* ; the aim is to create a trading system of items that cannot be duplicated. Cryptocurrencies propose alternative payment methods through their tokens that represent a currency, i.e., a generic payment instrument. Other types of tokens such as *security tokens* – representing a participation, in terms of dividends, voting rights, interest rates and/or percentage of the issuing entity's profits – and *utility tokens* – representing only the right to purchase goods and services of the issuing entity – were created on blockchain in order to digitally participate in a business having easy access to digital services-goods [235]. In the case of currencies, all blockchain properties (auditability in particular) are fundamental in the system, thus blockchain designers are forced to loose something in terms of performance since usually currencies are intended for the widest possible public. On the other hand, security and utility tokens are considered as an alternative investment method, therefore transparency is not essential in this case and one may adopt permissioned blockchains profiting from higher processing rate with respect to permissionless solutions.

ii) Business automatization : Blockchain platforms allow smart contract deployment and execution with the aim of letting any business automate its functionalities. After questioning the sensitivity of the automatically managed data (as in question A.1), it is important to consider the ability to support world changing applications. There is no perfect blockchain for every use-case. However, what a selection of participants is affecting the most are : (i) the non functional properties of *security* and *robustness* in terms of failures resistance and, (ii) all the features related to blockchain applicability – that is, the *flexibility* to adapt the designed blockchain protocol to different business cases. Therefore, the choice is a matter of trade-off ; more flexible architectures are usually less robust. *Trade-off (4.B.ii) flexibility VS robustness :* Permissionless blockchains suffer from limitations in data-storing, computations, scalability and performance which does not make it applicable to many business situations. On the other hand, permissioned blockchains result more flexible for configuration since governed and hosted by a single central committee of trusted nodes ; therefore, any type of change is made faster than in a fully open and trustless environment. A classic example is off-chain storage that results more intuitive in private networks that ease communications between the off-chain storage system and the blockchain [236]. Concerning security and robustness : is it better to adopt a permissionless blockchain architecture or to build a new structure on top of it. In fact, fully open-access distributed ledgers are quite robust against any type of failure as long as 50% of the system nodes are honest (see Section 1.4.1.3). In order to have robust but performing public blockchains, a possible solution is to use *side-chains* [237]. With side-chains one may move assets and functions from the principal blockchain (*main-chain*) to a second one. Thus, it is possible to have a private blockchain linked to a permissionless one. [230] gives a detailed report on the levels of performance and flexibility in permissioned, permissionless and open-permissioned blockchains. With regard to security and robustness in DLT we refer to the works of *Lin et al.* [40] and *Li et al.* [39].

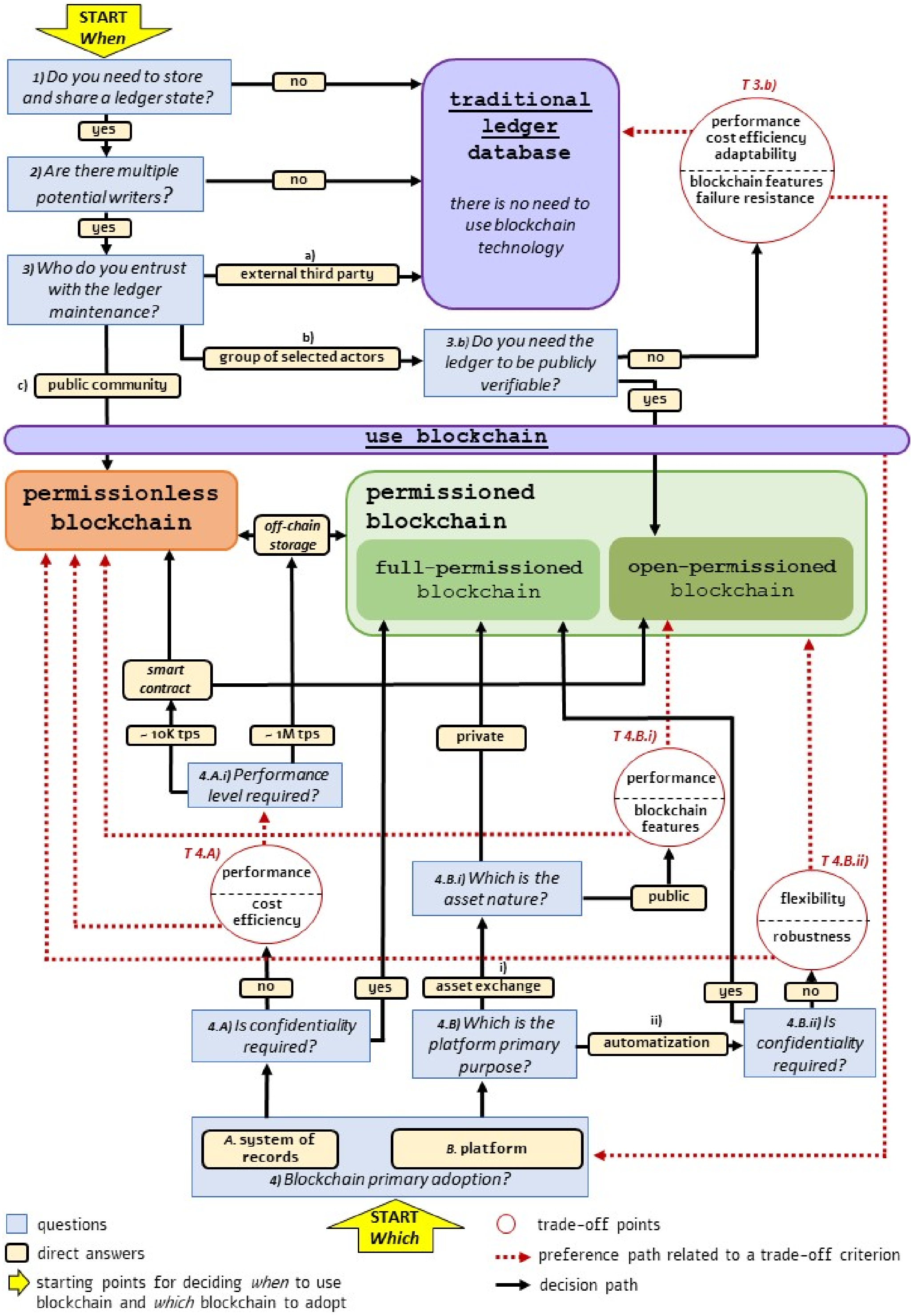

START
When
1) Do you need to store and share a ledger state?
no
yes
2) Are there multiple potential writers?
no
yes
3) Who do you entrust with the ledger maintenance?
a)
external third party
b)
group of selected actors
c)
public community
traditional ledger database
there is no need to use blockchain technology
T 3.b)
performance
cost efficiency
adaptability
blockchain features
failure resistance
3.b) Do you need the ledger to be publicly verifiable?
no
yes
use blockchain
permissionless blockchain
off-chain storage
permissioned blockchain
full-permissioned blockchain
open-permissioned blockchain
smart contract
~10K tps
~1M tps
4.A.i) Performance level required?
private
T 4.B.i)
performance
blockchain features
T 4.A)
performance
cost efficiency
4.B.i) Which is the asset nature?
public
T 4.B.ii)
flexibility
robustness
no
yes
i)
asset exchange
yes
no
4.A) Is confidentiality required?
4.B) Which is the platform primary purpose?
ii)
automatization
4.B.ii) Is confidentiality required?
A. system of records
B. platform
4) Blockchain primary adoption?
questions
direct answers
starting points for deciding when to use blockchain and which blockchain to adopt
START
Which
trade-off points
preference path related to a trade-off criterion
decision path

Figure 1.9 – When to use blockchain, and which type, instead of adopting a traditional database system. Red circles represents trade-off points between crucial aspects for the different blockchain use-case. The red arrows indicate the consequence of giving priority to one aspect rather than the other, while black arrows report answers to all the questions – coming with an order – of anyone interested in the blockchain technology. 'tps' : transactions per second.

1.6 Summary

For a new technology to realize its full potential, a lot of circumstances need to co-exist before network effects can be realized. Moreover, the actual advantages in using blockchain instead of any other traditional solution (such as centralized databases) need to be understood or at least showcase. Hence, the need for a *vademecum* guiding designers toward the right decision about when to adopt blockchain or not, which kind of blockchain better meets use-case requirements, and how to use it.

In this chapter we provided the community with such a vademecum, while giving a general presentation of blockchain (that goes beyond its usage in Bitcoin) and surveying a selection of the vast literature that emerged in the last few years. We draw the key requirements and their evolution when passing from permissionless to permissioned blockchains, presenting the differences between proposed and experimented consensus mechanisms, and describing existing blockchain platforms. All to provide any potential blockchain user with the information to join the blockchain network as a *transacting party* and/or a *validating node*.

Chapter 2

Game Theory and Rational Agents

Content

This chapter provides an introduction to Game Theory, a branch of Decision Theory that considers decision-makers as rational agents. The rationality of the players taking part in a game enables to predict their actions and thus the outcome of the game. Knowing a game (i.e., a problem that can be modeled as a game) and the strategies available to its players (i.e., the actions agents can perform), it is possible to predict its final outcome that can be more or less stable (i.e., equilibria or solution in the core). Game participants may play as individual or as a group leading to two different types of modeling. The theoretical content of this chapter derives from the following literature : [238; 239; 240; 241; 242]. Following chapters present real problems in the blockchain context that we model and analyze as games.

2.1 Introduction on Game Theory

Game theory is a branch of mathematics devoted to the study of optimal decision-making process in presence of multiple decision-makers. The latter are called *players* or equivalently *agents*. Game theory derives from *Decision Theory* and it can be also called *Interactive Decision Theory*. The difference between the two is that decision theory deals with a single agent's choices while game theory is characterized by having different agents (i.e., more than one) taking decisions. Moreover, game theory studies interactions among agents' choices ; decisions of a player depend on other agents' ones. The birth of game theory is largely attributed to the mathematician John von Neumann and the economist Oskar Morgenstern for their theory on economic and social behaviors based on the concepts of game and strategy published in the 1940s [243]. In 1950 Mathematician John Nash earned his PhD with a thesis on *non-cooperative games* [244] considered as the first significant extension of the work proposed by von Neumann and Morgenstern. Game theory has a wide range of applications in social sciences. Despite its numerous applications in recent years, game theory is still considered a young and developing science.

Several situations may be modeled as games leading to a game classification. Two main classes of games are : *non-cooperative games* and *cooperative games*. In this thesis, we deal with both classes of games i.e., games in which individual players compete among each other and games in which there exists the possibility of external enforcement to cooperate. These definitions materialize in several examples that are collected in [238].

In the following we present non-cooperative and cooperative games with the aim to show the reader the difference between the two classes of games. Although there exist problems that can be addressed with both non-cooperative and cooperative game theory making the distinction between the two quite informal, the problems we present throughout the thesis are modeled so that the non-cooperative and cooperative approaches differ from each other.

The following sections aim at familiarizing the reader with the concept of agents' rationality and game theoretical modeling in both non-cooperative and cooperative scenarios. Modeling techniques presented in this chapter are adopted in the following of this thesis to model blockchain users' behaviors. We focus then on a particular class of games adopted to model bankruptcy situations that may arise in the blockchain context as well. Properties of this class of games have important connotations when contextualized in the blockchain environment.

The chapter is organized as follows. Section 2.2 introduces the concept of agents' rationality. In Section 2.3 and Section 2.4 we present non-cooperative and cooperative games as well as their solution concepts. Bankruptcy games and their properties are presented in Section 2.5.

For further details on game theoretical modeling we invite the reader to consult [240].

2.2 Rationality of the Agents

Game theory helps understanding agents behaviors in different situation. It is based one fundamental assumption : ***rationality of the agents*** which does not necessarily mean being egoistic. Agents behaving egoistically try to get the best for themselves without considering the satisfaction of the others. On the other hand, rational agents aim to maximize their own *utility function* which not always coincides with an egoistic choice. Hence, rationality overcomes egoism by taking into account players' preferences and utility functions. The second assumption states that each player is able to completely analyze the problem knowing other agents' utility functions. Each player knows what other rational players do for maximizing their utility and behave accordingly. It is important to stress out that agents' payouts depends on others' people behaviors.

In general, any situation with two or more agents involving quantifiable consequences, can be modeled as a game and game theory can help determining the 'most likely' outcomes of the game. Let us define a few terms characterizing game theoretical modeling. A *game* consists of an interactive decision-making process among multiple decision-makers called *players*. Any set of actions available to a player in a game, depending on the game's structure, identifies a *strategy*. The *payoff* is the quantifiable (according to a utility function) payout a player receives according to the outcome of the game.

Let us present rationality in a more formal way. Each player of a game has some strategies to choose from and every combination of choices of each player results in a possible outcome of the game. The very first assumption characterizing game theory is the fact that **each player is able to order the outcomes of the game according to consistent preferences**. That is, players can observe all possible strategy combinations, they have preferences on the outcomes resulting from these combinations and they are able to order these preferences. These concepts can be translated in mathematical form with a function defined as follows.

Definition 2.1 (Utility function) *A utility function is a real valued function $u : X_i \to \mathbb{R}$ defined on the set of alternatives X_i of player i fulfilling the property that $u(A) \geq u(B)$ ($u(A) > u(B)$) if the alternative A is preferred (strictly preferred) to alternative B.*

It is important to notice that the utility function u depends from the behaviors of all players i.e., their strategy choices.

The second concept characterizing game theory is the fact that **players are fully able to analyze the problem (i.e., the game) and take the right decisions**. This fundamental assumption states that players are able to to make a full and deep analysis of the game, on other players' preferences and on the consequences of their choices. Players have then a global vision of the problem i.e., they know the involved players, the strategies they have , their preferences and the moment when they are called to make a decision.

2.3 Non-cooperative Games

Following the steps that gave birth to game theory, we start addressing non-cooperative game where players do not cooperate and make individual choices within a game.

2.3.1 Game Representation

Finite n-players games can be represented with a *tree* that provide information on the initial setting, all possible evaluations and all final outcomes of the game. This representation enables representing a set of players which act in sequence. Formally, the theoretical concept which models this situation is the *extensive form game* [245].

Definition 2.2 (extensive form game) *An extensive form game with perfect information is a tuple* $\Gamma = \langle N, T, P, (A_h)_{h \in V}, (u_i)_{i \in N} \rangle$, *where :*

- N *is the set of players.*
- $T = (V, E)$ *is a directed rooted tree.*
- $Z \subset V$ *is the set of terminal nodes.*
- $P : V \setminus Z \to N$ *is a function assigning to each non-end node a player in* N. *The function* P *identifies at which nodes a player acts.*
- $A_h = \{(x_h, x_i) \in E\}$ *for each node* $h \in V \setminus Z$ *is the set of edges going from node* h *to some other nodes and represents the set of actions at node* h *of the tree* T.
- $\Omega_i = \{s_i : V \setminus Z \to A_1 \times A_2 \times \dots A_h \times \cdots \times A_H, h : P(h) = i\}$ *is the set of pure strategies of player* i. *Every pure strategy of player* i *is a function that assigns an action* $a \in A_h$ *to every node* $h \in V \setminus Z$ *in which player* i *is involved (formally,* $h : P(h) = i$*).*
- $\mathcal{S}_i = \{\sigma_i : \Omega_i \to [0, 1], \sum_{s \in \Omega_i} \sigma_i(s) = 1\}$ *is the set of mixed strategies of player* i. *A mixed strategy is a probability distribution over the set of pure strategies of player* i.
- $u_i : Z \to \mathbb{R}$ *is the utility function for player* $i \in N$.

Fig. 2.1 represents a game in extensive form Γ with players $N = \{A, B\}$ and non-terminal nodes $V \setminus Z = \{a, b, c\}$. The structure and the notation of a game in extensive form is not practical for representing finite 2-players games. Every game in extensive form can be rewritten in a more compact way, called *normal form* representation [245], as shown in Fig. 2.2.

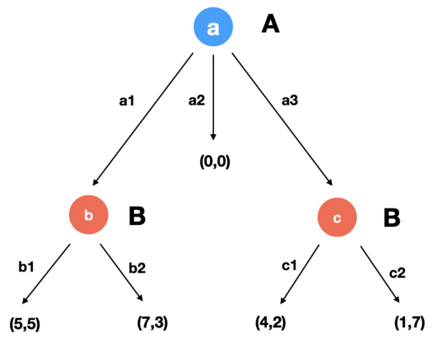


FIGURE 2.1 – Game Γ in extensive form.

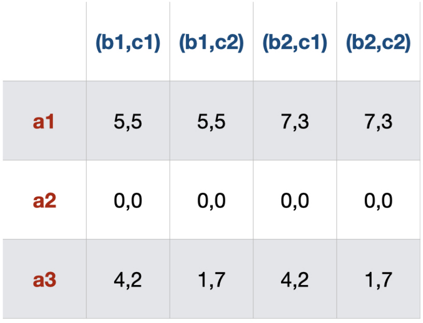
Fig. 2.1 represents a game in extensive form Γ with players $N = \{A, B\}$ and non-terminal nodes $V \setminus Z = \{a, b, c\}$. The structure and the notation of a game in extensive form is not practical for the purpose of the analysis. Every game in extensive form can be rewritten in a more compact way, called *normal form* representation [245], as shown in Fig. 2.2.

Definition 2.3 (normal form game) *A game in a normal form representation is identified by a tuple* $\Gamma = \langle N, \mathcal{S}, u\rangle$*, where* N *is a finite set of* n *players,* $\mathcal{S} = \mathcal{S}_1 \times \mathcal{S}_2 \times \cdots \times \mathcal{S}_n$ *where* $\mathcal{S}_i$ *is the set of strategies of player* i *and* $u : \mathcal{S} \to \mathbb{R}^n$ *is the utility function of the players.*

A set of strategies $\mathcal{S}_i$ is available to every player i in the game. Supposing that every player picks a strategy $\sigma_i \in \mathcal{S}_i$; then it is possible to compute the utility for a player i, $u_i(\sigma_1, \sigma_2, \ldots, \sigma_n)$, which is the i-th component of the utility function u. Since players are rational agents, their goal is to maximize their utility by choosing their own strategy. Usually there is no strategy that allows every player to maximize their utility, therefore we have to consider strategy profiles $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_i, \ldots, \sigma_n) \in \mathcal{S}$. Each player i chooses a strategy σ_i and the outcome $u(\sigma)$ may please every player so that they do not want to change their strategy. The following section introduces *solution concepts* of a game i.e., strategy profiles with different properties.

2.3.2 Solution and Equilibria

A solution concept is a strategy profile that described the dynamic and the result (i.e., the final outcome) of the game. Von Neumann and Morgenstern [243] further characterized the concept of rationality by providing an additional assumption ; the **elimination of dominated strategies**. This assumption provides a method to solve a game identifying its final outcome as well as its strategy



	(b1,c1)	(b1,c2)	(b2,c1)	(b2,c2)
a1	5,5	5,5	7,3	7,3
a2	0,0	0,0	0,0	0,0
a3	4,2	1,7	4,2	1,7

FIGURE 2.2 – Game Γ in normal form.

profile. Elimination of dominated strategies assumes that, independently from the choice the other players make, a player does not choose alternative A, if B is available to her allowing her to get more. By iterating the process of eliminating dominated strategies, a game in normal form can be solved. If a strictly dominant strategy exists for one player in a game, that player will play that strategy. If both players have a strictly dominant strategy, the game has only one unique equilibrium known as *Nash equilibrium.*

Definition 2.4 (Nash equilibrium) *A strategy profile* $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_i, \ldots, \sigma_n) \in \mathcal{S}$ *is a Nash equilibrium if for every player* i *and for every* $\tau_i \in S_i$ *we have that :*

$$u_i(\sigma_1, \sigma_2, \ldots, \sigma_i, \ldots, \sigma_n) \geq u_i(\sigma_1, \sigma_2, \ldots, \tau_i, \ldots, \sigma_n)$$

The definition of Nash equilibrium is based on the concept of *best response*, i.e., the strategy σ_i that maximizes the utility of a player i, given the strategies of the other players σ_{-i}. In a Nash equilibrium no player has an incentive to unilaterally change its strategy since utilities do not increase. Nash [246] proves that every game in normal form admits at least one Nash equilibrium. *Nash equilibria* are reasonable solution concepts since they represent a scenario in which nobody is tempted to unilaterally change her own strategy. However, the set of Nash equilibria is not always a singleton, it might happen indeed that there is more than one equilibrium. Here below some properties of Nash equilibria are introduced.

Definition 2.5 (strong Nash equilibrium [247]) *A Nash equilibrium* $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_i, \ldots, \sigma_n)$ $\in \mathcal{S}$ *is said to be strong if and only if for all* $C \subseteq N$ *and all* $\tau_C \in \mathcal{S}_C$*, there exists* $i \in C$ *such that* $u_i(\sigma_C, \sigma_{-C}) \geq u_i(\tau_C, \sigma_{-C})$.

In [247] the authors prove that the outcome of every strong Nash equilibrium is Pareto efficient i.e., no player can improve her outcome without reducing the outcome of another players. Strong Nash equilibria are easy to be identified, but they do not always exist.

Definition 2.6 (stable Nash equilibrium [248]) *A Nash equilibrium $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_i, \ldots, \sigma_n) \in \mathcal{S}$ is said to be stable if it belongs to the set S which is minimal with respect to the following property : for every $\epsilon > 0$ there exists $\delta > 0$ such that any upper-hemicontinuous compact convex valued correspondence pointwise within Hausdorff distance δ of the best response correspondence of Γ has a fixed point within ϵ of S.*

The concept of stable equilibria was introduced in [249] in order to exclude less meaningful Nash equilibria i.e., those equilibria that are less resilient against small changes. After [249], several other definitions of stability were introduced. We cite the definition provided in [248], which fulfills some useful properties. One of these states that there always exists a stable Nash equilibrium. Moreover, stable Nash equilibria survive after the iterated deletion of *weakly dominated strategies*, i.e., those strategies $\sigma_i \in \mathcal{S}_i$ that perform as well as or worse than another strategy $\sigma'_i \in \mathcal{S}_i$ no matter which strategy the other players choose (formally, we have that $u_i(\sigma_i, \tau_{-i}) \leq u_i(\sigma'_i, \tau_{-i})$ for all $\tau_{-i} \in \mathcal{S}_{-i}$). In the process of iterated deletion [239] weakly dominated strategies are excluded from the set of strategies available to players and the set of Nash equilibria is recomputed.

A Nash equilibrium of an extensive form game is a Nash equilibrium of the associated normal form game. A *subgame* of an extensive form game (with perfect information) on node $v \in V \setminus Z$ is another extensive form game (with perfect information) obtained as a part of the directed tree starting at the decision node v. A *subgame perfect equilibrium* is a strategy profile that induces a Nash equilibrium on any subgame. For more details see, for instance, the book [250].

Extensive form games can be represented in normal form and solved with the principle of elimination of dominated strategies, however an alternative method exists and is known as ***backward induction***. It consists of reasoning backwards in time, from the end of a problem (represented by the end node) till the beginning (the first/root node).It proceeds by examining the last decision step, identifying what action would be most optimal at that moment and iterating the procedure. Note that both solving principles, elimination of dominated strategies and backward induction, are used to solve games and find Nash equilibria in Chapter 4.

2.4 Cooperative Games

In order to characterize cooperative game theory it is crucial to identify a set of players and to define a characteristic function for games. We denote by N a finite set of players. Any subset S of N is said a coalition. The cardinalities of N and S are denoted by n and s, respectively while 2^N denotes the set of all possible coalitions.

Definition 2.7 (Characteristic function) *The characteristic function (or side payment) of a cooperative game is a function*

$$v : 2^N \to \mathbb{R}$$

such that $v(\emptyset) = 0$, where $\emptyset$ denotes the empty set.

This function v associates a real value to all the possible coalitions. More precisely, $v(S)$ represents the value that each coalition S can get for itself, once it is formed. The condition on the empty set represents a sort of normalization condition.

Definition 2.8 (Cooperative game) *A cooperative game is defined by the pair (N, v) where N is the set of players and v is the characteristic function that assigns a value $v(S)$ to each coalition $S \subset N$.*

The value $v(S)$ represents in general something that can be shared among agents in the coalition. Whenever amount $v(S)$ can be freely divided among the members of S, in any way we have that the game (N, v) is a Transferable Utility (TU) game.

The question now is : *how is this value shared ?*. In our work, the following chapters are devoted to find the answer to the question : *how is this bitcoin reward shared among pools' participants ?*

2.4.1 Solution, Imputation and Core

Given the definition of cooperative game, it is now the time to introduce the idea of solution. Given a game (N, v) a solution is a vector $(x_1, ..., x_n)$ where x_i represents the amount assigned to the player i. Moreover given a game (N, v) a set of solution vectors is a solution concept.

As we can see, there are several solution concepts for cooperative games. This can be explained with the fact that cooperative model is complex and can fit several situations. Therefore, it is not possible to define a solution concept providing a reasonable outcome for all the situations that a game can model. That is the reason why some requirements have to be asked. There are three minimal conditions that are common to all the solution concepts :

1. $x_i \geq v(i) \quad \forall i$
2. $\sum_{i=1}^{n} x_i \leq v(N)$ (feasibility)
3. $\sum_{i=1}^{n} x_i \geq v(N)$ (efficiency).

The first condition has a simple meaning : the output of the game must give to each player not less than what he/she can get by its own. The second condition establishes that the amount distributed among players cannot be more than the available one, while the third inequality is an efficiency condition, since the whole amount $v(N)$ has to be distributed. All the solutions satisfying these important constraints are called *imputations*.

Definition 2.9 (Imputation) *Given a cooperative game (N, v), we call imputation any vector x fulfilling condition (1) and condition (2) presented above.*

We can denote this set of reasonable solutions by $\mathcal{I}(v)$. Let us now consider a subsets of $\mathcal{I}(v)$.

Definition 2.10 (The Core) *Let $v : 2^N \to \mathbb{R}$ be a TU game. The core of the game, denoted by $\mathcal{C}(v)$, is the set :*

$$\mathcal{C}(v) = \left\{ x \in \mathbb{R}^n : \sum_{i \in N} x_i = v(N) \wedge \sum_{i \in S} x_i \geq v(S), \forall S \subset N \right\}.$$

The idea behind the definition of the core is to enforce condition (1) which guarantees that players do not object since they do not get less than what they can get alone. Imputations are feasible solutions rejected neither by the single players nor by the grand coalition N. Imputations in the core are not rejected by any coalition. Therefore, $\mathcal{C}(v)$ is an imputation set without all the imputations that are rejected by at least one coalition. The core is a meaningful concept since it provides a set of efficient, feasible and *stable* solutions. Each coalition with associated payoffs in the core receives at least what it deserves, therefore no subset of players have any incentive in leaving the grand coalition. In Definition 5.19 it is the following inequality

$$\sum_{i \in S} x_i \geq v(S), \forall S \subset N$$

which represents a stability concept guaranteeing that agents with offered payoffs in the core stay in the grand coalition and cooperate together.

2.5 Bankruptcy Games

A bankruptcy situation arises whenever there are some agents claiming a certain amount of a divisible estate, and the sum of the claims is larger than the estate. Let $N = \{1, \ldots, n\}$ be a set of agents. Formally, a *bankruptcy situation* on the set N consists of a pair $(\mathbf{c}, E) \in \mathbb{R}^N \times \mathbb{R}$ with $c_i \geq 0$ for all $i \in N$ and $0 < E < \sum_{i \in N} c_i = C$. The vector $\mathbf{c}$ represents agents' demands (each agent $i \in N$ claims a quantity c_i) and E is the estate that has to be divided among agents (and it is not enough to satisfy the total demand C).

Denoting by $\mathbb{B}^N$ the class of all bankruptcy problems $(\mathbf{c}, E) \in \mathbb{R}^N \times \mathbb{R}$ with $0 < E < \sum_{i \in N} c_i$ and N as a set of agents. A *solution* (also called *allocation rule*) for bankruptcy situations on N consists of a map $f : \mathbb{B}^N \to \mathbb{R}^N$ assigning to each bankruptcy situation in $\mathbb{B}^N$ an *allocation vector* in $\mathbb{R}^N$, which specifies the amount $f_i(\mathbf{c}, E) \in \mathbb{R}$ of estates E that each player $i \in N$ receives in situation $(\mathbf{c}, E)$. A solution must satisfy a minimal set of "natural" requirements :

— $f_i(\mathbf{c}, E) \geq 0 \quad \forall i \in N$ (Individual rationality), saying that every agent must receive a non-negative amount of the estate E ;

— $f_i(\mathbf{c}, E) \leq c_i \quad \forall i \in N$ (Demands boundedness), stating that no agents receives strictly more than what she/he claims ;

— $\sum\limits_{i \in N} f_i(\mathbf{c}, E) = E$ (Efficiency), requiring that the entire estate must be allocated among the claiming agents.

We now introduce three well-studied solutions for bankruptcy situations (see, for instance, [242; 241; 251]). The first one is the *Proportional rule.*

Definition 2.11 (Proportional rule (P)) *For each bankruptcy situation $(\mathbf{c}, E) \in \mathbb{B}^N$, the proportional rule yields the allocation vector $P(\mathbf{c}, E) = \pi \mathbf{c}$, where π is such that $\sum_{i \in N} \pi c_i = E$.*

Example 2.5.1 *Consider a bankruptcy problem with three claimants $N = \{1, 2, 3\}$ and such that $(\mathbf{c}, E) = ((4, 5, 3), 10)$. First, compute the parameter $\pi = \frac{E}{C} = \frac{10}{12} = \frac{5}{6}$, then get the proportional allocation multiplying π by the vector of demands $\mathbf{c}$:*

$$P(\mathbf{c}, E) = \frac{5}{6}(4, 5, 3).$$

The second solution from the literature that we consider is the *Constrained Equal Awards rule.* This allocating rule ignores differences among claimants and rule out cases where agents receiving more than what they claim thanks to the boundedness constraint.

Definition 2.12 (Constrained equal awards rule (CEA)) *For each bankruptcy situation* $(\mathbf{c}, E) \in \mathbb{B}^N$*, the constrained equal awards rule is defined as* $CEA_i(\mathbf{c}, E) = \min\{c_i, \lambda\}$ *where the parameter* λ *is such that* $\sum_{i \in N} \min\{c_i, \lambda\} = E$*.*

The CEA rule consists of giving to every agent the same amount until the agent's demand is not completely satisfied and the estate is not finished.

Example 2.5.2 *Consider the bankruptcy situation given in Example 2.5.1. The first step assigns to all players* $x = (3, 3, 3)$*. The game to solve is now :* $(\mathbf{c}', E') = ((1, 2, 0), 1)$*. Since the third agent is satisfied* E' *have to be allocated within the first two players. Here* $\lambda = \frac{7}{2}$ *hence,*

$$CEA(\mathbf{c}, E) = \left(\frac{7}{2}, \frac{7}{2}, 3\right).$$

The last division rule we consider, i.e. the *Constrained Equal Losses rule*, is an alternative to the CEA rule which focuses on the losses in which claimants incur. More precisely, instead of equating awards this rule equates losses.

Definition 2.13 (Constrained equal losses rule (CEL)) *For each bankruptcy situation* $(\mathbf{c}, E) \in \mathbb{B}^N$*, the constrained equal losses rule is defined as* $CEL_i(\mathbf{c}, E) = \max\{c_i - \lambda', 0\}$ *where the parameter* λ' *is such that* $\sum_{i \in N} \max\{c_i - \lambda', 0\} = E$*.*

Example 2.5.3 *Consider again the bankruptcy situation given in Example 2.5.1. At first, a loss of 1 unit is assigned to each player :* $x = (3, 4, 2)$*. Then, it is easy to check that* $\lambda' = \frac{1}{3}$ *and therefore, we have* $(4, 5, 3) - \frac{1}{3} = (\frac{10}{3}, \frac{13}{3}, \frac{7}{3})$*. So,*

$$CEL(\mathbf{c}, E) = \left(\frac{10}{3}, \frac{13}{3}, \frac{7}{3}\right).$$

2.5.1 Bankruptcy Games : Game theoretical Division Rules

Bankruptcy problems are not necessarily bankruptcy games, they can be solved without using game theory, i.e. by applying a natural allocating rule such as the proportional one. Moreover, division rules are not necessarily game theoretical division rules. However, these types of problems where an estate is divided among claimants suggest a cooperative behavior. Therefore, it is natural to try to model bankruptcy problems as a cooperative game and analyze the relative solutions. The first step is defining a characteristic function.

It is clear that the grand coalition must form and that $v(N) = E$, but it is not clear which value should be associated to single players and to intermediate coalitions. From the literature there exist a "pessimistic" approach to evaluate a coalition ; the pessimistic characteristic function assigns to a coalition a value which is what S gets once the other $N \setminus S$ players have already taken what they claim. The coalition gets the part of the estate that is left after all other players are fully rewarded.

$$v_p(S) = \max\Big(0, E - \sum_{i \in N \setminus S} c_i\Big)$$

Bankruptcy problems can be represented as cooperative games with pessimistic characteristic functions. At this point, it is natural to ask : *what about division rules associated to this type of games ?* Authors in [251] proposed a necessary and sufficient condition for a division rule to be a game theoretical division rule :

Theorem 2.1 (Curiel 1987) *An allocating rule $f(\mathbf{c}, E)$ for bankruptcy problems is a game theoretical division rule if and only if $f(\mathbf{c}, E) = f(\mathbf{c}^T, E)$, $c_i^T = \min\{c_i, E\}$ for all $i \in N$.*

This condition derives from the fact that cooperative games corresponding to the bankruptcy problems $(\mathbf{c}, E)$ and $(\mathbf{c}^T, E)$ are the same. According to theorem both the constrained equal losses and the proportional rules are not a game theoretical rules. For the CEL rule it is not so immediate as for the proportional one. Let us show this fact with a simple example :

Example 2.5.4 *Let $(\mathbf{c}, E) = ((2, 11), 10)$ and so $(\mathbf{c}^T, E) = ((2, 10), 10)$.*
We have two different outcomes for the problems :

$$CEL(\mathbf{c}, E) = (0.5, 9.5), \quad CEL(\mathbf{c}^T, E) = (1, 9).$$

Concerning the constrained equal awards rule, it is immediate to see that it is a game theoretical division rule by simply using the rule's definition : $CEA_i(\mathbf{c}^T, E) = \min\{c_i, E, \lambda\} = \min\{c_i, \lambda\} = CEA_i(\mathbf{c}, E)$ since $\lambda \in [0, c_n] : \lambda \leq E$.

Another interesting consideration on the solutions of bankruptcy problems is the fact that the core of the associated pessimistic games coincides with the the set of admissible solutions.

Proposition 2.1 *Given a bankruptcy problem* $(\mathbf{c}, E) \in \mathbb{B}^N$ *and its associated pessimistic cooperative game* $v_{p(\mathcal{C},E)}$*, calling* x *the outcome of the problem/game we have that*

$$x \in \mathcal{C}(v_p) \Longleftrightarrow \sum_{i \in N} x_i = E \quad \textit{and} \quad 0 \leq x_i \leq c_i, \forall i \in N$$

Proof. *Concerning the only if part, the first condition is efficiency. For the second one we get* $\forall i \in N, x_i \geq v_p(i) \geq 0$ *and* $E - x_i = \sum_{j \in N \setminus \{i\}} x_j \geq v_p(N \setminus \{i\}) \geq E - c_i$ *which implies* $x_i \leq c_i$*.*
In order to prove the if part let us notice that efficiency condition is satisfied.

This theorem states that every game theoretical division rule of bankruptcy problems provides allocations which are in the core of the associated cooperative games. Therefore given a bankruptcy game $(\mathbf{c}, E)$, the constrained equal awards rule provides an outcome contained in the set $\mathcal{C}(v_{(\mathbf{c},E)})$ as $\forall S \subseteq N$ we have two cases : $v_p(S) = 0 \leq \sum_{i \in S} x_i$ and $v_p(S) = E - \sum_{i \in N \setminus S} c_i \leq E - \sum_{i \in N \setminus S} x_i \leq \sum_{i \in S} x_i$.

2.5.2 Bankruptcy Rules' Properties

Following the presentation in [242], we now focus on some properties from the literature that are satisfied from the three previously introduced solutions, and that will be relevant for the discussion about Bitcoin systems in the next section. The first property consists of a basic equity requirement stating that players with identical claims are identical for the allocation rule. More precisely, agents with the same demands are treated identically in the sense that they receive the same fraction of the estate.

Property 2.1 (Equal treatment of equals) *For all* $(\mathbf{c}, E) \in \mathbb{B}^N$ *and all* $i, j \in N$ *we have that* $c_i = c_j$ *implies*

$$f_i(\mathbf{c}, E) = f_j(\mathbf{c}, E).$$

The second property requires that an allocation rules is invariant to scale change, so it does not depend on the units in which problems' demands and estate are expressed.

Property 2.2 (Scale invariance) *For all $(\mathbf{c}, E) \in \mathbb{B}^N$ and all $\gamma > 0$ we have :*

$$f(\gamma\mathbf{c}, \gamma E) = \gamma f(\mathbf{c}, E).$$

The next property refers to a situation where the population of agents may change. This property considers the case in which after providing an allocation vector according to a solution applied to the original situation $(\mathbf{c}, E) \in \mathbb{B}^N$, a group of agents $S \subset N$ forms and re-apply the solution on the reduced situation $(\mathbf{c}_S, \sum_{i \in S} f_i(\mathbf{c}, E)) \in \mathbb{B}^S$, where $\mathbf{c}_S = (c_i)_{i \in S}$ and the new estate is $\sum_{i \in S} f_i(\mathbf{c}, E)$. This property states that a solution applied to any reduced problem should provide to agents in S the same output as in the original situation.

Property 2.3 (Consistency) *For all $S \subset N$, all $(\mathbf{c}, E, N) \in \mathbb{B}$ and all $i \in S$ we have :*

$$f_i(\mathbf{c}, E, N) = f_i(\mathbf{c}_S, \sum_{i \in S} f_i(\mathbf{c}, E, N), S).$$

The following two properties characterise the allocation priorities of an allocation rule. Exemption establishes that when claims are smaller than the equal division the rule has to first satisfy them and therefore cut larger claims.

Property 2.4 (Exemption) *For all $(\mathbf{c}, E) \in \mathbb{B}^N$ if $c_i \leq E/n$ then $f_i(\mathbf{c}, E) = c_i$.*

Exclusion gives an opposite message, since it excludes from the problem those agents with very low claims.

Property 2.5 (Exclusion) *For all $(\mathbf{c}, E) \in \mathbb{B}^N$ if $c_i \leq L/n$ where $L = \sum_{i=1}^n c_i - E$ then $f_i(\mathbf{c}, E) = 0$.*

The next property deals with the possibility that a group of agents may aggregate their demands and appear as a single claimant, or that a single agent may split her demand to represent several claimants. If these behaviors are not beneficial for the agents, we say that a solution satisfies no advantageous merging or splitting.

Property 2.6 (No advantageous merging or splitting) *Let $(\mathbf{c}, E) \in \mathbb{B}^N$ and $(\mathbf{c}', E') \in \mathbb{B}^{N'}$ be such that $N' \subset N$, $E = E'$ and suppose there is an agent $i \in N'$ such that $c'_i = c_i + \sum_{j \in N \setminus N'} c_j$ and $c'_j = c_j$ for each $j \in N' \setminus \{i\}$, then*

$$f_i(\mathbf{c}', E') = f_i(\mathbf{c}, E) + \sum_{j \in N \setminus N'} f_j(\mathbf{c}, E).$$

It is possible to split this property in two distinct properties by considering the two different behaviors of merging and splitting.

Property 2.7 (No advantageous merging) *Let* $(\mathbf{c}, E) \in \mathbb{B}^N$ *and* $(\mathbf{c}', E') \in \mathbb{B}^{N'}$ *be such that* $N' \subset N$, $E = E'$ *and suppose there is and agent* $i \in N'$ *such that* $c'_i = c_i + \sum_{j \in N \setminus N'} c_j$ *and* $c'_j = c_j$ *for each* $j \in N' \setminus \{i\}$, *then*

$$f_i(\mathbf{c}', E') \leq f_i(\mathbf{c}, E) + \sum_{j \in N \setminus N'} f_j(\mathbf{c}, E).$$

Property 2.8 (No advantageous splitting) *Let* $(\mathbf{c}, E) \in \mathbb{B}^N$ *and* $(\mathbf{c}', E') \in \mathbb{B}^{N'}$ *be such that* $N' \subset N$, $E = E'$ *and suppose there is and agent* $i \in N'$ *such that* $c'_i = c_i + \sum_{j \in N \setminus N'} c_j$ *and* $c'_j = c_j$ *for each* $j \in N' \setminus \{i\}$, *then*

$$f_i(\mathbf{c}', E') \geq f_i(\mathbf{c}, E) + \sum_{j \in N \setminus N'} f_j(\mathbf{c}, E).$$

According to Property 2.7 no group of agents has an incentive to aggregate their demands and to be treated as a single agent whose claim is the sum of the individual demands of their participants. Conversely, Property 2.8 says that no agent has an incentive to divide his/her demand to represent several claimants whose demands add up to her/his original claim. Solutions introduced in this section satisfy alternative sets of properties, as reported in Table 2.1 (for more details on these results see, [242]).

Properties	CEA	CEL	P
Equal treatment of equals	Yes	Yes	Yes
Scale invariance	Yes	Yes	Yes
Consistency	Yes	Yes	Yes
Exemption	Yes	No	No
Exclusion	No	Yes	No
No advantageous merging or splitting	No	No	Yes
No advantageous merging	Yes	No	Yes
No advantageous splitting	No	Yes	Yes

TABLE 2.1 – Three bankruptcy solutions and some of their properties.

Summary. This chapter presents the two different ways to model situations where agents act rationally (i.e., as rational players) that play as an individual (i.e., normal and extensive form games) or as a group (i.e., cooperative games). Then, a particular class of cooperative game, bankruptcy games, is presented. These games model bankruptcy situations i.e., situations where a resource need to be allocated among a set of claiming agents. In the blockchain context, bankruptcy situations arise when validating nodes (modeled as rational users) need to be remunerated for their validating tasks via new minted crypto-currencies.

Chapter 3

Rational Validators and Rewarding Strategies

Content

This chapter models and analyzes the behaviors of a particular type of validating nodes operating on a specific DLT i.e., Bitcoin miners. Miners who operate validating tasks in the Bitcoin blockchain are modeled as rational agents who may or may not behave maliciously (i.e., perform an attack) in certain situations. The situation we present in this chapter is the process of remunerating miners for their validating work. Two types of malicious behaviors are analyzed ; *pool-hopping* and *block-withholding.* The first part of the chapter is devoted to describing and assessing the pool-hopping phenomenon while in the second part block-withholding miners are modeled as players of a bankruptcy game. The content of the chapter originates from [15] and [16].

3.1 Introduction to Rewarding Miners

As presented in Chapter 1 (Section 1.3), blockchain transactions are collected in blocks, validated and published on the distributed ledger. Nakamoto [20] proposed a Proof-of-Work system based on Back's Hashcash algorithm [135] that validates blocks and chains them one to another. Let us recollect that the Proof-of-Work system requires finding an input of a predefined one-way function (*e.g.,* hash function) generating an output that meets the difficulty target. More precisely, the goal for the block validators (*miners*) is to find a numerical value (*nonce*) that added to an input data string and "hashed" gives an output which is lower than the predefined threshold. A miner who finds a *full solution* (i.e., a nonce meeting the difficulty target) broadcasts it across the network.

Miners compete to be the first to find a full solution in order to publish the block and gain a reward, denoted with B, consisting in new minted crypto-currencies. The Bitcoin monetary policy establishes the reward amount one receives once it validates a block (6.25 BTC, valid until the next halving scheduled for 2024 when the block mining reward is decreasing to 3.125 BTC) – noting that a block can contain more than 2000 transactions, while respecting the block size limit of 1 MB. Mining is a competitive crypto puzzle (a *mining race*) that participants try to solve as fast as possible. The difficulty D of the crypto puzzle limits the rate at which new transaction blocks are generated by the blockchain (*e.g.,* it takes approximately 10 minutes to find a full solution in the Bitcoin network). This difficulty value is adjusted periodically in order to meet the established validation rate. At the time of writing, in order to validate a block in the Bitcoin blockchain, miners needs to generate (on average) a number $D = 20.1\text{T}$ of hashes [1].

Mining is a procedure through which miners can gain a substantial amount of money. Nowadays, due to the high difficulty values, solo miners (i.e., miners who work alone with a personal device) find a full solution with a time variance range of billions of years. Small miners survive in this new industry by joining mining pools. A mining pool is a cooperative approach in which multiple miners share their efforts (i.e., their computational power) in order to validate blocks and gain rewards. Once a full solution is found, pool's reward is split among the miners. In this way small miners, instead of waiting for years to be rewarded, gain a fraction of the reward on a regular basis.

Miners' reward is based on their contribution in finding a full solution. In order to give proof of their

[1] The value has been recorded on October 31^{st}, 2021.

work, miners submit to the pool partial solutions, i.e., nonces that do not meet the original threshold, but a higher one. The solutions of this easier crypto puzzle are considered "near to valid" solutions and called *shares*. For those blockchains that adopt a SHA-256 function, every *hash value* (i.e., output of the hash function) is a full solution with probability $\frac{1}{2^{32}D}$, and each hash has a probability of $\frac{1}{2^{32}}$ to be a share. Hence, a share is a full solution with probability $p := \frac{1}{D}$.

Miners are rewarded according to the number of shares that they provide. Whenever a share is also a full-solution a block is validated and the pool gains a reward that is split among pool participants according to the number of shares that they have reported. Mining pools are managed by a pool manager that establishes the way in which miners should be rewarded. Each pool adopts its own rewarding system.

There exist several rewarding approaches that can be more or less attractive to miners (see [252]). These rewarding methods have to meet certain requirements in order to guarantee the proper functioning of the ecosystem. Since the aim of forming pools is to share efforts and rewards thus, participants should be rewarded in a fair manner with respect to their contribution to the pool, i.e., proportionally to the hashing power invested in the pool. Original simple rewarding methods met this fairness criterion at the expense of vulnerability to some miners strategic behaviors.

Mining pool attacks.

An attack to a mining pool refers to any miner's behavior which differs from the default practice (the honest one) and that jeopardizes the collective welfare of the pool. Rosenfeld provided in [252] an overview of the possible malicious behaviors regarding pools whose profitability depends on their own rewarding mechanism. Miners attack a mining pool in order to obtain a higher reward i.e, miners behave as rational agents whose decisions are driven by the expected gain.

Miners may attack their pool at the time of reporting their Proof-of-Work. More precisely they can (i) delay in reporting a share (i.e., *block-withholding*) and/or (ii) report a share elsewhere (i.e., *pool-hopping*). The former is a practice consisting in delaying in reporting shares and full solutions to a mining pool. This practice implies delaying a block validation and the consequent possession of the reward, that in some cases may be profitable for attackers. Pool-hopping is an attack where miners "hop" from a pool to another one according to pools' attractiveness.

Related works and contributions.

The problem for a pool manager is to establish how to redistribute the rewards among pool participants in order to prevent malicious behaviors (as the ones listed above). In other words, the pool manager must choose an "appropriate" rewarding mechanism preventing (possibly, all) different types of attacks. Concerning the block-withholding practice, *Schrivers et al.* [253] make use of non-cooperative game theory to propose a rewarding mechanism (denoted as *incentive compatible*) that prevents this attack. This specific rewarding system is robust against malicious actions operated inside a pool, however it does not behave as well in an inter-pool environment since it cannot prevent pool-hopping. In this case, Rosenfeld [252] shows that malicious miners can gain at the expenses of the honest ones, who receive a lower reward than the expected one. In [254] the authors use cooperative games to prove that pool-hopping is not preventable, thus mining pools are not stable coalitions.

The first part of this chapter, Section 3.2, presents the methodology designed to analyze the pool-hopping phenomenon, focusing on the detection of pool-hoppers. By analyzing those Bitcoin transactions that pools create for rewarding its participants, it is possible to determine time epochs where miners worked. Thus, we can identify those miners who have worked intermittently for pools adopting a rewarding system which pays out for each validated block. This evaluation leads us qualifying the miners who have hopped along with their hopping behavior and financial performance.

In the second part of the chapter, Section 3.3, starting from the model in [253], we propose an alternative incentive compatible rewarding mechanism discouraging the pool-hopping practice. We model and analyze with game theory the rewarding problem by reinterpreting the reward function in [253] as an outcome of a bankruptcy game. Then, we construct, analyze and test a new rewarding mechanism adoptable by pools to remunerate contributing miners.

3.2 Pool-hopping and Rewarding Strategies

Bitcoin public nature provides us with both *generation transactions* [45] and transactions used by pools to remunerate their participants. The first are those transactions by means of which new bitcoins are generated and issued in the system. The second are the so called "*rewarding transactions*" which are transferring the funds received through the generation process to pool participants in a chain of ownership, i.e., new minted bitcoins are transferred from pools' addresses (*input*) to miners' ones (*output*) in the form of Unspent Transaction Output, UTXO (see Section 1.6). Proper analysis of the public ledger can reveal a lot of information on the network users. Deanonymization techniques applied on a selected set of users do contribute in enhancing the quality of the extracted information. In our analysis, the Bitcoin actors we are interested in are the ones involved in the rewarding transactions, i.e., pools and miners, remunerated according to pools policies ; the first step of our work is determining the related *user network* [255].

Rewarding methods with direct correspondences between block validation and miners' payout do help in hoppers detection. Through a targeted analysis of the transactions ledger, it is possible to place a miner – intent on working for a pool – within a time interval denoted as *epoch*. This time period is nothing more than an estimate of a miner's working time for a given pool. Hoppers detection consists in comparing miners' epochs in different pools. All the actors in a pool have full knowledge of the internal mining activities carried out over time, which is not the case for any user external to pool. Once pool-hoppers have been identified among all the miners interacting with multiple pools, it is possible to describe the hopping evolution over time.

3.2.1 Rewarding Methods

Rewarding methods [256] generally pay out according to a division into *rounds*, where a round is the time elapsed between two different block validations performed by the same pool. Thus, pool distributes the block reward among miners at round end, in proportion to their contribution within the round. Some methods replace such a round-based remuneration opting for the attribution of a *score* to each share. Score-based methods register the time of a share submission, assign a value to the share according to a scoring function and remunerate the submitter correspondingly. Thus, rewards calculation is no more linked to the concept of round but it is based on the scoring systems in use.

However, payouts are still executed on a round basis.

Besides rewarding fairness, score-based methods also offer pool-hopping resistance. Meni Rosenfeld analyzes in [256] several score-based rewarding systems. We briefly present in the following the two most popular methods of this type in use in 2017 [2] :

— *Pay-Per-Last-N-Shares (PPLNS) method [256].* It represents an allocation rule rewarding miners for their recently submitted shares – eventually computed at different rounds. The block reward B, instead of being distributed only among actual round participants, can also be assigned to those miners who have submitted shares in a previous round. Then, for short actual rounds, the rewarding calculation may consider also shares submitted in previous rounds. Note that, with the PPLNS system it can happen that some submitted shares may not be considered for the reward calculation (i.e., shares in long actual rounds). N varies with the difficulty parameter D ; it can be up to twice D (rounded down to integer), as an upper-bound threshold. More accurate implementations tend to set $N < 2D$; e.g., the author in [257] proposes to score shares as the inverse of D, and determines N as a given portion of the score, hence $N \leq D$.

— *Slush method.* Slush pool [258] has been the first mining pool developing a rewarding method aimed at combating pool-hopping. Block rewards are divided in proportion to the score miners obtained with their reported share(s) in the rounds. Slush's scoring system dynamically calculates a score for a specified share according to its submission time. More precisely, the score at time t_0 for a share s submitted at time t_s is given by :

$$score(s, t_0) = \exp\left(\frac{t_s - t_0}{\Lambda}\right) \tag{3.1}$$

where Λ is a constant value representing the speed at which the score decreases over time. An exponentially decreasing scoring is well suited due to its invariance to time shift.

3.2.2 Pool-Hopping

Pool-hopping is a strategic behavior influenced by the attractiveness of mining in terms of expected payoffs [259]. In other words, pool-hopping consists in mining for a pool only when its attractiveness is high and leaving it when it is not (i.e., directing computing power towards another pool).

[2] The first year of the thesis and the year this contribution has been submitted to be published.

Meni Rosenfeld shows in [256] that pools using proportional systems encourage this practice in the sense that pool-hopping results more profitable than mining continuously. Since for proportional methods each share is paid out according to the total number of submitted shares in the round, the longer is the round the less is the reward per share. Each share payoff corresponds to $\frac{B}{S}$, where S is the total number of shares submitted to the pool during a round. Thus, submitting shares to a pool as long as the rounds are short and then hopping elsewhere results in a higher gain than the fair share [253]. What makes this practice, known as *early mining*, feasible is the miners' knowledge of the round start time and the number of shares taken into account for the reward calculation. PPLNS and Slush methods propose scoring functions that do not encourage early mining practice, thus the pools are preventing hopping behaviors on round length basis.

However, as the Slush score is a function of the time elapsed since a share submission, pools' attractiveness is eventually affected by the hash-rate fluctuations over time. Pools' hashing power directly influences the number of shares found and the round length. In addition, both PPLNS and Slush systems' 'hoppability' is affected by difficulty (D) and reward (B) adjustments. Therefore, advanced pool-hopping strategies are possible for both Slush and basic PPLNS implementations. Nevertheless, the two methods present different levels of hoppability as described in [257].

3.2.3 User Sub-network Inference Process

Bitcoin transactions move funds from one or more 'inputs' to one or more 'outputs' containing all the references of previous and next owners. Users receive bitcoins in the form of unspent transaction output (UTXO) in their *addresses*, i.e., strings identifying sources and destinations for bitcoin payments. In order to be spent, funds contained in the output of a transaction must turn into the input of a new one. Inputs and outputs must match in the sense that the entire value of the previous transaction(s) has to be transferred. Thus, whenever a user aims at sending an amount lower than the UTXO value, a second output containing a *changing address* is added. That is, a bitcoin address held by the sender appears as an output with the purpose of transferring funds back.

3.2.3.1 Coinbase, rewarding and transferring transactions

Each transactions block contains a generation transaction remunerating the pool or the miner who validates it. These type of transactions does not specify any previous UTXO to spend, its input field

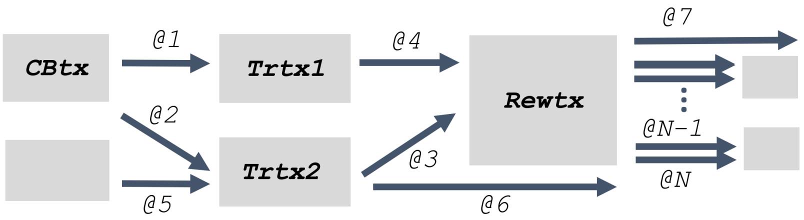


Figure 3.1 – Transaction sub-network. Each arrow is labeled with sender's address. CBtx, Trtx, Rewtx are respectively coinbase, transferring and rewarding transactions.

(denoted as *coinbase*) contains arbitrary data ; generation transactions, also known as *coinbase transactions*, provide funds that cannot be spent for at least 100 confirmed blocks. Due to this requirement, Bitcoin system rewards only for valid blocks in the *longest chain* (i.e., the longest blockchain branch). While mining pools are financed by coinbase transactions, miners are remunerated through ordinary transactions we denote as *rewarding transactions*. Ideally, they should transfer the coinbase UTXO, however this is not always the case. Some pools use additional transactions – we refer to as *transferring transactions* – that move funds to specified addresses within the pool, which reward miners later on.

All these transactions are publicly readable from the blockchain ledger and can be easily gathered through Bitcoin-Core [260]. This information leads to construct the **transaction sub-network** [255] representing the flow of bitcoins between a specific subset of transactions – of coinbase, rewarding and transferring types – over time. Fig. 3.1 example shows pool-miners interactions where coinbase transactions are inputs of transferring transactions whose outputs consist in rewarding ones. Arrows represent the funds flow.

The topology of Fig. 3.1 example synthesizes a generic view of the Bitcoin network, where one may detect the relationships between various input and output addresses. For coinbase, rewarding and transferring transactions, the addresses involved certainly belong to pools and miners however, it is not known to which user they belong exactly since each of them can have multiple addresses. Moreover, it is considered a good practice for confidentiality to generate and use new addresses for each operation on the network. As hoppers detection requires the analysis of the interactions among Bitcoin specific users, it is necessary to map each bitcoin actor with its associated addresses.

3.2.3.2 Address-User Mapping Procedure

The address-user mapping problem can be solved using two basic heuristics introduced in [261], which derive from the expert knowledge of Bitcoin protocols and common practices. The mapping

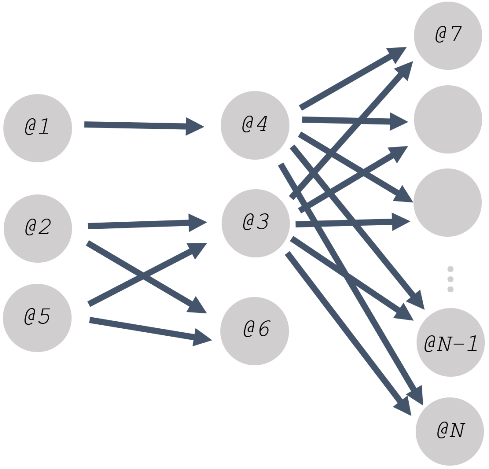


Figure 3.2 – Address sub-network related to Fig. 3.1 example.

procedure we use is based on the concatenated application of the two heuristics. The first heuristic, addressing transaction inputs only, exploits the concept of *multi-input transactions* : it assumes that all the inputs in a transaction belong to the same bitcoin user. This heuristic results reasonable in its simplicity since (i) different users likely very rarely collaborate in a single transaction, and (ii) very often it happens that users move funds from their multiple addresses to one or multiple deposit accounts. The second heuristic deals with changing addresses : it associates input addresses with output ones when they result to be completely new (i.e., they never appeared in the blockchain before).

Therefore, applying the mapping procedure one can derive the user sub-network from the address sub-network. The relationship between addresses can be easily inferred from the transactions sub-network ; Fig. 3.2 shows the address sub-network corresponding to the Fig. 3.1 example. Once the address sub-network is derived, addresses are associated with their users following the two heuristics ; Fig. 3.3 shows an instance of user sub-network for the given example.

It is not possible to obtain a perfectly true **user sub-network**, since heuristics are not perfect and addresses do not contain any information about their owners. In order to maximize the precision, due to the limited number of coinbase, transferring and rewarding transactions with respect to the overall volume of transactions, we have applied the mapping procedure to the entire transaction network (including ordinary transactions as well). Doing so, we can provide an accurate representation of miners and pools interactions, since we are associating with these specific users the highest possible number of employed addresses.

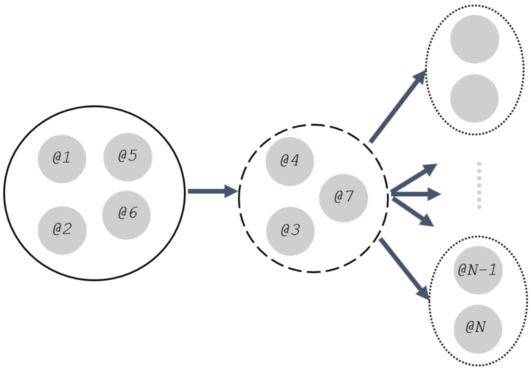


FIGURE 3.3 – User sub-network related to Fig. 3.1 example.

The application of such deanonymization techniques for the mapping allows us building an interaction network between pools and miners. Pool-hopping consists in mining for two or more pools during a round, thus hoppers submitting shares to multiple pools are remunerated accordingly by multiple pools. Hoppers detection requires the analysis of those miners who are connected to several pools.

3.2.4 Hoppers Detection

The previous section provides us with those network pre-processing steps for detecting pool-hoppers. From the constructed user sub-network it is possible to select those miners who are remunerated from multiple pools. In order to determine whether a miner who submits shares to more than one pool is really implementing a hopping strategy, it is necessary to focus on its rewarding transactions. Our analysis addresses those pools adopting rewarding methods which are paying out miners on a round basis and assumes that :

— Miners' pay-outs are entirely originated by coinbase transactions [3] and,

— Each reward is originated by a single coinbase transaction [4].

3.2.4.1 Simplistic Case

For pools using per-round rewarding systems, rewarding transactions should be ideally linked to the corresponding rounds they are paying for. This link could be found in their inputs, namely in the coinbase transactions providing pools with the rewards that have to be redistributed. It would then be possible, by analyzing the rewarding transactions, to place a miner in a certain round. In Fig. 3.4

[3] This work does not take into account rewards mining pool can gain from non-mining activities.

[4] Funds deriving from different coinbase transactions may be adopted to pay out a single miner contribution.

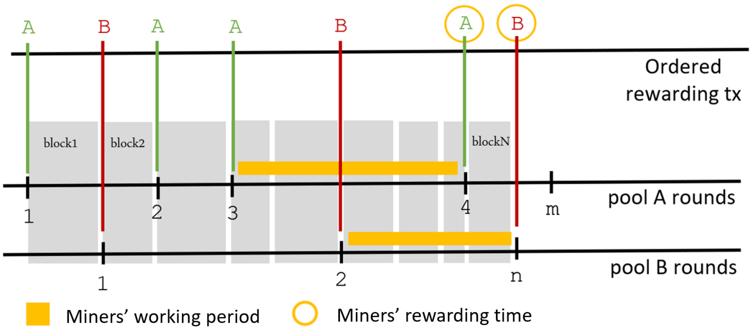


Figure 3.4 – Simplistic miners' positioning. Blocks are represented with gray boundaries. Vertical lines associate each round end with the corresponding rewarding transaction.

we show how to identify miners' working period : a miner rewarded at time A is associated with the corresponding round of pool A that is marked in yellow. Then, considering multiple pools and their rounds, it would be enough to check if the same miner is present in two overlapping rounds of different pools. The procedure results quite trivial once pools' rewarding transactions are ordered according to their corresponding rounds, that is, in accordance with the corresponding coinbase transactions.

In a 2-pool case, if the same miner is present in two consecutive (ordered) rewarding transactions belonging one to a pool and one to another one, we can declare it as a hopper. There may be doubtful situations whenever the same miner is found in two rewarding transactions of different pools which are not directly consecutive (i.e., after ordering we could have two transactions created by pool A, one created by pool B and the same miner present in the first and third transaction). In this cases it is mandatory to analyze the rounds length to establish if the miner adopted or not a hooping strategy.

3.2.4.2 Realistic Case

In reality, pools do not associate exactly a paying transaction to each round. Miners receive their payouts once they cross a predefined *sending threshold* (Slush pool adopts a threshold of 0.001 BTC [258]). This means that in reality pools pay for miners' work on multiple rounds with a single rewarding transaction. In Fig. 3.5 pool A is rewarding a miner for its work in two rounds.

Rewarding transactions no longer use the reward provided by a single coinbase but they have several generation transactions as inputs. In this way, it is no longer possible to place a miner in its working round.

Miners paid out with the rewarding transactions can be placed within a time interval that we

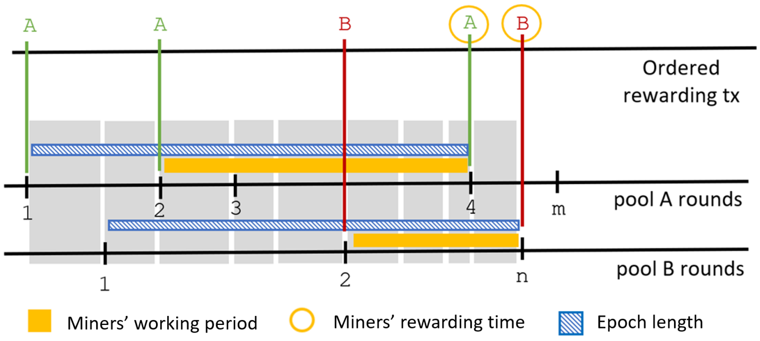


FIGURE 3.5 – Realistic miners' positioning.

denote as "*epoch*". This time period can coincide with one or more rounds. We set the beginning of an epoch with the oldest coinbase used as an input in the rewarding transaction. The epoch ends with the round allowing miners to reach the sending threshold, namely the round closing about 100 blocks before the one in which the rewarding transaction takes place. As shown in Fig. 3.5 the epoch end coincides with the last round end. The epoch start depends on the coinbase transactions used in the rewarding transaction. Pools may pay out miners with funds generated during the rounds they work for or they may use older coinbase transaction as in Fig. 3.5. Note that we assume payments are executed as soon as funds are available (after 100 confirmations), which results in neglecting the processing time for the rewarding transactions.

By positioning miners in epochs, we cannot identify the exact rounds they work for. They may have worked continuously in all the rounds within the epoch or they could have worked intermittently. The idea is always to check if a miner is present in two overlapping time periods, however it can be defined as a hopper with a certain probability depending on epochs' length. As for the ideal case, transaction ordering should simplify the detection procedure. Rewarding transactions can be sorted by membership blocks, that is, ordering procedure refers to the coinbase transactions corresponding to epoch ends. For the 2-pool case in which a miner is found in two consecutive rewarding transactions with different creators, we can conclude with any doubt that the miner is a pool-hopper. This is due to epoch definition, since it ends with that round in which miners submitted the number of shares necessary to reach the threshold value. Hence, there is the certainty of miners' presence in the epochs' final rounds.

By considering only consecutive transactions, other cases where hopping is possible are overlooked. First of all, this approach detect only hoppers in epochs' final rounds without considering the other rounds within them. Moreover, as in the simplistic case, we can have hoppers paid out in two rewarding transactions that are not adjacent. Epochs length could help us in detecting miners who are likely to hop, thus in this cases we are talking about *potential hoppers*. Potential hopping is formalized in the following paragraphs but not considered in the measurement results as left for future works.

How can we establish the hopping probability for a potential hopper ? Evidently it depends on the number of overlapping rounds that the epochs in question contains and on their lengths.

Pool operator's point of view. The case in which a user can easily define the hopping probability for a potential hopper is the one in which he is a pool manager. Let us suppose that a pool operator aims at detecting among the miners who are working for him those who are hopping to a second pool adopting a score-based rewarding method paying out on a round basis (i.e. ; Slush or PPLNS). This manager can precisely position his miners in his pool's round history. Moreover, through a proper analysis on the transaction network he can derive a miner sub-network and subsequently can place this entities within epochs. Whenever the operator finds one of his workers in an epoch that overlaps with his pool's rounds, he can compute the hopping probability for his miner. Given a miner m who works for the pool manager in the rounds $r_I^{p_1}$ where $I \subset [1, \infty)$ and who is present in a time epoch $E \subset [1, \infty)$ of a second pool with length $l = \sum_{i \in E} r_i^{p_2}$ his hopping probability would be the following :

$$\mathbb{P}(m \text{ is an hopper}) = \frac{\sum_{i \in E \cap I} r_i^{p_2}}{l}.$$

External user's point of view. Hopping probability in this case may be defined for two pools using PPLNS or Slush method. Potential hopper is any miner found in two overlapping epochs. However, those potential hoppers presenting in two consecutive rewarding transactions created by two different pools can be defined as real hoppers. For those potential hoppers who can be found in not directly consecutive transactions the hopping probability can be defined as follows :

$$\mathbb{P}(m \text{ is an hopper}) = \frac{\sum_{i \in E_1 \cap E_2} r_i^{p_2}}{\sum_{i \in E_1} r_i^{p_1}},$$

where E_1, E_2 denotes pools' overlapping epochs ordered according to the oldest coinbase. Thus, the probability for a miner to hop depends on the number of overlapping rounds with respect to the oldest epoch's length. In the trivial case with one overlap between a round ($|E_2| = 1$) and a younger epoch, we can classify it as a pool-hopping behavior.

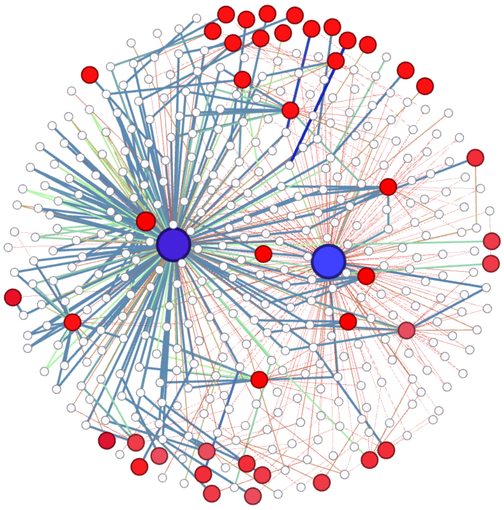


FIGURE 3.6 – Pool-miners graph representation. The left blue node is Kano pool and the right blue node is Slush pool. Detected pool-hoppers are highlighted in red. Edge thickness and color depend on the frequency of users interactions.

3.2.5 Measurement Results

We analyze the user sub-network involving miners interacting with the two most popular mining pools adopting score-based rewarding methods : Slush pool [258] and Kano pool [262]. Since these pools opt for a per-round rewarding system, we can apply the hopping detection technique showed in Section 3.2.4. We aim at capturing *real-hoppers* (i.e., miners who have definitely hopped) and the result of their behavior. Thanks to the user sub-network, we can describe the phenomenon evolution over a predetermined time window.

The analyzed period is April 6-20, 2016, chosen for its massive transactions flow (over 5m transactions) such that Kano pool recorded its maximum hashing power [263] ; with a substantial number of miners attracted in joining the pool it has been a remarkable period for hopping opportunities.

First, we detect all the miners, interacting with the two pools, that can potentially adopt a hopping strategy. Afterward, according to our hypothesis (Section 3.2.4), we capture the real-hoppers among them. Moreover, we perform a daily analysis on the number of hoppers and the number of 'hops' made, both globally and individually. Fig. 5.21 captures the interactions of the miners with Kano and Slush pools (represented by blue nodes). Overall, 43 pool-hoppers were detected within a population of 69 miners active with the two pools (26 miners are not hopping within the time period, thus they are mining continuously for one of the two pools), and an approximate global population of 150.000

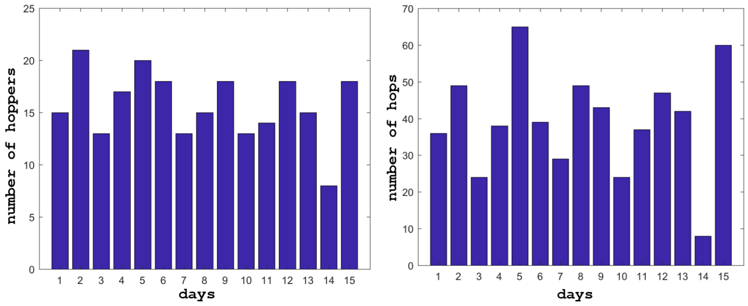


FIGURE 3.7 – Daily number of hoppers and hops.

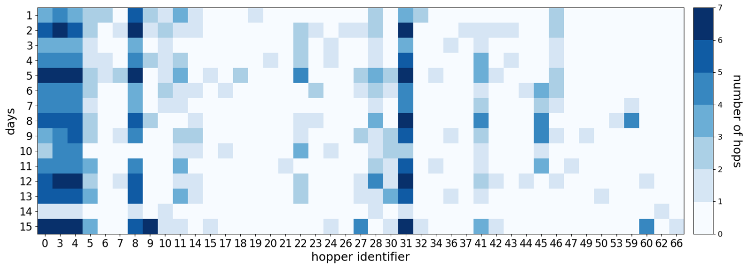


FIGURE 3.8 – Heat map : daily hopping intensity per hopper.

miners. Fig. 3.7 shows the trend of the number of pool-hoppers and their hops during the considered 15 days.

Fig. 3.8 further characterizes the phenomenon on a daily basis, showing for each of the identified hoppers the number of hops per day. We can roughly spot 3 different types of hopper : (i) *frequent hoppers*, miners who opt for a hopping strategy more than 10 times (e.g., #0, #3, #4), (ii) *occasional hoppers*, performing a limited number of hops, less than 10 times (e.g., #6, #7, #14), and (iii) *changing-pool hoppers*, miners who hop just once in the time window (e.g., #19, #20, #21).

Changing-pool hoppers may have simply changed the pool to work for, since they do not present an intermittent behavior. Frequent and occasional hoppers represent together the 46.4% of the hopping miners, while 11 miners perform only a single hop.

In addition we quantify the hopping frequency for those miners who are changing pool more than 5 times in the observed period. Fig. 3.9 presents the distribution of the *inter-hopping* times (i.e., the time elapsing between two successive hops performed by the same hopper), with boxplots (reporting a quartile box, minimum and maximum) indicating over the number of hops. Approximately half of the miners shows a median time less than 12 hours, with small variance, while the other half shows a higher median with an inter-quartile range of many hours. There is no evidence of the adoption of a

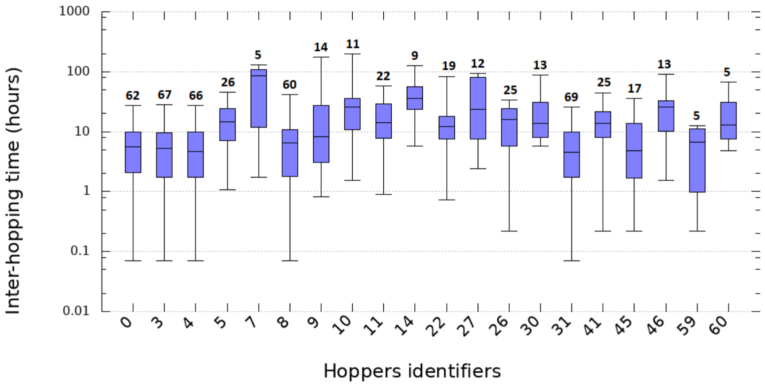


FIGURE 3.9 – Inter-hopping time boxplot distributions (logscale).

strict hard-coded frequency, hence the hopping decision appears to be event-based as expected.

Moreover, we are interested in quantifying the advantage in hopping. The distribution of hoppers' revenues over the time window is presented in Fig. 5.22 with boxplots ; the hoppers' order is the same than the one in Fig. 3.8 ; for some hoppers, only few transactions exist and in that case instead of the boxplot only few points are plotted. The last two boxplots on the right-side of Fig. 5.22 cover all hoppers together and all static miners (that mine continuously without hopping) in the time period ; we find a median transaction reward 3 times higher for hopping miners than for static ones. This highlights a very important gain justifying pool-hopping ; note that miners' computational power cannot be inferred from the ledger, and in practice for pool-hoppers it may significantly differ from that of the 'typical' miner.

Finally, we report in Fig. 3.11 the correlation between the number of hops and the Bitcoin (USD) price evolution over the 15 days ; we compute the average Bitcoin price and the number of hops performed within a time window of two hours. No evident correlation between the two distributions emerges, i.e., we find here a confirmation that pool-hopping strategies do not appear to be hard-coded in miners behavior as a function of the Bitcoin actual value.

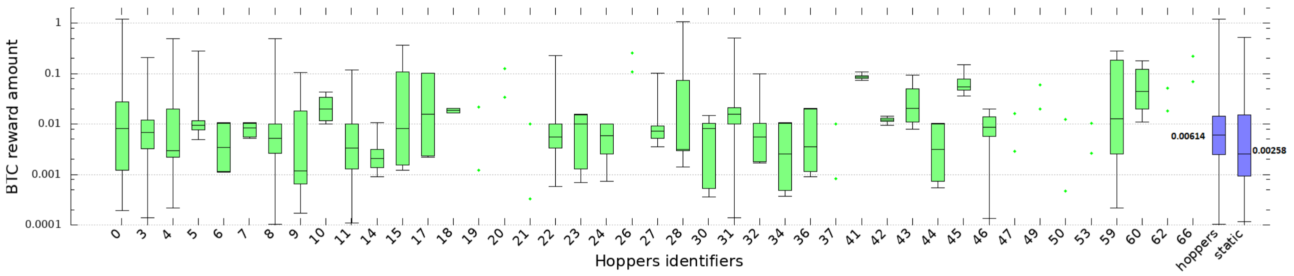


FIGURE 3.10 – Boxplot distributions of rewarding transactions BTC values (logscale).

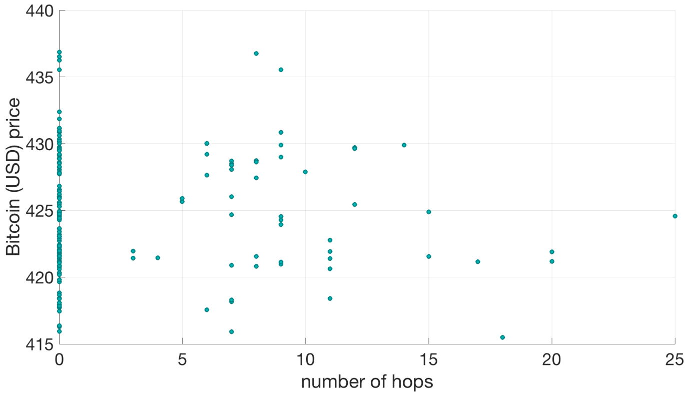


FIGURE 3.11 – Correlation plot : BTC price vs number of hops.

Experimental findings. Our analysis determines that the hopping phenomenon is relatively limited in a time window which could have led to a high hopping propensity (only 0.02% of the miners did perform pool-hopping between the two selected pools).

Nonetheless, it is worth noting that our approach is able to spot hoppers assuming rewards happen in a per-round fashion, which is not always the case in practice ; so, the number of hoppers we detect as to be considered a lower bound, and further studies are needed to get closer to an even more realistic hopper detection.

Independently of this latter aspect, the key finding of our empirical study is that we show hoppers' rewards significantly exceed those of static miners. Besides these facts, we have also determined that pool-hoppers have disparate behaviors, likely not correlated, and not dependent on the actual value of the crypto-currency. This suggests that there may not be a shared methodology to implement a hopping strategy, and that pool-hoppers do proceed on a do-it-in-your-own, manual, fashion. Similar conclusions have been stated by works that have removed the assumptions we made in the hopper identification phase [264; 265].

3.3 Rewarding Rational Miners : Bankruptcy Situations

Following the logic introduced in [253], we propose an alternative rewarding mechanism discouraging both pool-hopping and block-withholding behaviors. This section abandons score-based rewarding methods (presented in the first part of this chapter) and analyzes round-based remuneration systems. By reinterpreting the reward function proposed in [253] as an outcome of a bankruptcy situation, we construct, analyze and test a new rewarding mechanism adoptable by pools to remunerate miners.

The following section presents the basic model for mining pool. Please note that definitions about bankruptcy situations are also introduced in Chapter 2. In Section 3.3.2, we introduce a reward function from the literature, we compare it with a new one (based on a modified version of the CEL rule for bankruptcy situation) and we show that the two are equivalent with respect to incentives in reporting shares or full solutions. Then, in Section 3.3.3, we compare these two methods from a multi-pool perspective by showing (also with the aid of simulations) that the CEL-based reward function performs better than the one from the literature in discouraging miners to hop from a pool to another.

3.3.1 The Model

Let $N = \{1, \dots, n\}$ be a finite set of miners. Time is split into *rounds*, i.e., the period it takes any of the miner in the pool to find a full solution. During a round miners participate in the mining race and report their shares (and the full solution) to the pool manager. Once the full solution is submitted, the pool manager broadcasts the information to the network and receives the block reward B. Then, the pool manager redistributes the block reward B among the miners according to a pre-defined reward function. The round is then concluded and a new one starts. For the sake of simplicity we set $B = 1$.

The situation is represented by the vector $\mathbf{s} = (s_1, s_2, ..., s_n) \in \mathbb{N}^N$, defined as *history transcript*, that contains the number of shares s_i reported by each miner $i \in N$ in a round. Letting $S = \sum_{i \in N} s_i$ be the total number of reported shares, the reward function $R : \mathbb{N}^N \to [0, 1]^n$, according to [253], is a function assigning to each history transcript $\mathbf{s}$ an allocation of the reward $(R_1(\mathbf{s}), \dots, R_n(\mathbf{s}))$, where R_i denotes the fraction of reward gained by the single miner $i \in N$ and $\sum_{i \in N} R_i = B = 1$.

Following the approach in [253], under the assumption of rationality, miners want to maximize their individual revenues over time. Let K be the numbers of rounds that have been completed at time t and let $\mathbf{s}_j$ be the transcript history for any round $j \in K$. Given a reward function R, a miner

$i \in N$ will adopt a strategy (i.e., the number of reported shares at each round j) aimed at maximizing her total reward given by

$$\lim_{t \to +\infty} \sum_{j \in K} R_i(\mathbf{s}_j),$$

where a strategy affects both the number of completed rounds and the number of reported shares. In [253], a reward function R is said to be *incentive compatible* if each miner's best response strategy is to immediately report to the pool a share and a full solution. Assuming that (i) one single pool represents the total mining power (normalized to 1) of the network and that (ii) each miner $i \in N$ has a fraction α_i of the hashing power, then the probability for a miner i to find a full solution is α_i. Under this assumption, Schrijvers et al. [253] show that a miner $i \in N$ has an incentive to immediately report her shares if and only if the reward function R is monotonically increasing (i.e., $R_i(\mathbf{s} + \mathbf{e}^i) > R_i(\mathbf{s})$ for all history transcripts $\mathbf{s}$, where $\mathbf{e}^i = (e^i_1, \ldots, e^i_n) \in \{0,1\}^N$ is a vector such that $e^i_j = 0$ for each $j \in N \setminus \{i\}$ and $e^i_i = 1$.). Moreover, they show that a miner $i \in N$, finding a full solution at time t, has an incentive to immediately report it if and only if the following condition holds :

$$\sum_{j=1}^{n} \alpha_j \cdot (R_i(\mathbf{s}^t + \mathbf{e}^j) - R_i(\mathbf{s}^t)) \leq \frac{\mathbb{E}_{\mathbf{s}}[R_i(\mathbf{s})]}{D} \tag{3.2}$$

for all vectors of mining powers $(\alpha_i)_{i=1}^n$ and all history transcripts $\mathbf{s}^t$, where $\mathbb{E}_{\mathbf{s}}[R_i(\mathbf{s})]$ is the expected reward for miner i over all possible history transcripts. Condition (3.2) results from the comparison of the withholding strategy – i.e., $\sum_{\mathbf{s}:S=1} \mathbb{P}(\text{find } \mathbf{s}) \cdot (R_i(\mathbf{s}^t + \mathbf{s}))$ – with the honest one – i.e., $R_i(\mathbf{s}^t) + \frac{\mathbb{E}_{\mathbf{s}}[R_i(\mathbf{s})]}{\mathbb{E}_{\mathbf{s}}[S]}$ – knowing the fact that the total number of submitted shares in a round, $S = \sum_{i \in N} s_i$, follows a geometrical distribution of parameter $p = \frac{1}{D}$ (i.e., the probability for a share to be a full solution), where D is the difficulty of the crypto puzzle. Therefore, the value of the parameter D corresponds to the average number of submitted shares in a round.

Let us now report some game-theoretical basic definitions presented in Chapter 2. A *bankruptcy situation* arises whenever there are some agents claiming a certain amount of a divisible estate, and the sum of the claims is larger than the estate. Formally, a *bankruptcy situation* on the set N consists of a pair $(\mathbf{c}, E) \in \mathbb{R}^N \times \mathbb{R}$ with $c_i \geq 0$ $\forall i \in N$ and $0 < E < \sum_{i \in N} c_i = C$. The vector $\mathbf{c}$ represents agents' demands (each agent $i \in N$ claims a quantity c_i) and E is the estate that has to be divided among them (and it is not sufficient to satisfy the total demand C).

We denote by $\mathbb{B}^N$ the class of all bankruptcy situations $(\mathbf{c}, E) \in \mathbb{R}^N \times \mathbb{R}$ with $0 < E < \sum_{i \in N} c_i$ A *solution* (also called *allocation rule* or *allocation method*) for bankruptcy situations on N is a map

$f : \mathbb{B}^N \to \mathbb{R}^N$ assigning to each bankruptcy situation in $\mathbb{B}^N$ an *allocation vector* in $\mathbb{R}^N$, which specifies the amount $f_i(\mathbf{c}, E) \in \mathbb{R}$ of the estate E that each agent $i \in N$ receives in situation $(\mathbf{c}, E)$.

A well-known allocation rule in the literature is the *Constrained Equal Losses* (CEL) rule, which is defined in Def. 2.13 (see, for instance, [242; 241] for more details on bankruptcy situations and the CEL rule).

3.3.2 Incentive Compatible Reward Functions

Schrijvers et al. [253] introduce a reward mechanism that fulfills the property of incentive compatibility using the identity of the full solution discoverer w. Given a vector $\mathbf{e}^w = (e_1^w, \dots, e_n^w) \in \{0,1\}^N$ such that $e_i^w = 0$ for each $i \in N \setminus \{w\}$ and $e_w^w = 1$, the incentive compatible reward function R is the following :

$$R_i(\mathbf{s}; w) = \begin{cases} \frac{s_i}{D} + e_i^w\Big(1 - \frac{S}{D}\Big), & \text{if } S < D \\ \frac{s_i}{S}, & \text{if } S \geq D \end{cases} \qquad \forall i \in N, \tag{3.3}$$

where s_i is the number of shares reported by miner i, S is the total number of reported shares in a round and D is the crypto puzzle difficulty.

This function rewards miner i proportionally to the submitted shares in the case $S \geq D$. On the other hand, in the case $S < D$, each miner receives a fixed reward-per-share equal to $\frac{1}{D}$ and the discoverer w of the full solution receives, in addition, all the remaining amount $1 - \frac{S}{D}$. So, in both cases, $\sum_{i \in N} R_i(\mathbf{s}; w) = B = 1$. We can see that the reward function R is the combination of two distinct allocation methods. In a *short round*, i.e., when the total amount of reported shares is smaller than the difficulty D of the original problem, the reward function allocates a fixed amount-per-share to all agents equal to $\frac{1}{D}$, but the agent w who finds a solution is rewarded with an extra prize. Instead, in a *long round*, i.e., when the total amount of reported shares exceeds the difficulty of the problem, the reward function allocates the reward proportionally to the individual shares.

Remunerating miners in a per-share fashion, for long rounds, would lead pool going bankrupt since the reward B results insufficient to pay out all the reported shares. For long rounds, the rewarding mechanism proposed in [253] is nothing more than a solution to a bankruptcy situation. Therefore, it is possible to create new reward functions by simply substituting in long rounds (i.e., in bankruptcy situations) different bankruptcy solutions.

Hence we can construct two new rewarding functions by substituting to the proportional division the CEA and the CEL rules defined in Def 2.13 and Def 2.12 respectively.

$$R_i^{cea}(\mathbf{s}; w) = \begin{cases} \frac{s_i}{D} + e_i^w \Big(1 - \frac{\mathbf{S}}{D}\Big), & \text{if } \mathbf{S} < D \\ \min\Big(\frac{s_i}{D}, \lambda\Big) \quad \text{s.t.} \quad \lambda : \sum_i \min\Big(\frac{s_i}{D}, \lambda\Big) = 1, & \text{if } \mathbf{S} \geq D \end{cases} \qquad \forall i \in N$$

$$R_i^{cel}(\mathbf{s}; w) = \begin{cases} \frac{s_i}{D} + e_i^w \Big(1 - \frac{\mathbf{S}}{D}\Big), & \text{if } \mathbf{S} < D \\ \max\Big(\frac{s_i}{D} - \lambda', 0\Big) \quad \text{s.t.} \quad \lambda' : \sum_i \max\Big(\frac{s_i}{D} - \lambda', 0\Big) = 1, & \text{if } \mathbf{S} \geq D \end{cases} \qquad \forall i \in N$$

In Section 2.5.2 we have listed some common properties for solutions of bankruptcy situations. Some of these properties are satisfied by proportional, CEA and CEL solutions (see Table 2.1), and therefore they are obviously also satisfied by R, R^{cea} and R^{cel}, respectively, on the long run (i.e., $\mathbf{S} \geq D$). In the following, we discuss the interpretation of these properties in the Bitcoin context.

*Equal treatment of equals (2.1).*This property ensures impartiality : agents with the same number of shares have to be rewarded in the same way. In the Bitcoin framework impartiality of the pool manager is mandatory.

Scale invariance (2.2). This property ensures the independence of the reward function from the currency in which problem's variables are defined.

Consistency (2.3). Consistency expresses the independence of a reward function with respect to population restrictions. In the Bitcoin framework, this property implies that no group of miners has an incentive to re-apply a solution in the reduced problem and there is no advantage for miners to renegotiate the allocation in a sub-group once a reward function is accepted by the pool's participants.

No advantageous merging or splitting (2.6). This property rules out the possibility for a group of miners to aggregate their shares in order to appear as a single miner, or conversely, the possibility for a single miner to split its share and appearing as several miners. In the Bitcoin framework, this last behavior identifies with the Sybil attack presented in Chapter 1. These miners can create multiple accounts with different "username" and e-mail addresses and pretend to be different entities. The proportional rule is the only rule which satisfies no advantageous merging or splitting. Therefore, it guarantees pools' robustness with respect to the Sybil attack. However, in order to prevent this attack it is enough to have an allocation rule satisfying only *no advantageous splitting.* As reported in Table 2.1, we have that the CEA-based reward function R^{cea} satisfies *no advantageous merging (2.7)* on long runs, whereas the CEL-based reward function R^{cel} satisfies *no advantageous splitting (2.8)* on long runs.

Hence, we can create a new rewarding mechanism based on the CEL rule defined in Definition 2.13 that is robust against the Sybil attack and that preserves incentive compatibility.

Definition 3.1 *Given the identity of the full solution discoverer w, for all $i \in N$ the CEL-based reward function $\widehat{R}$ is defined as follows :*

$$\widehat{R}_i(\mathbf{s}; w) = \begin{cases} \frac{s_i}{D} + e_i^w\Big(1 - \frac{S}{D}\Big), & \textit{if } S < D \\ \frac{e_i^w}{D} + \max\Big(\frac{s_i}{D} - \lambda, 0\Big),\ \lambda : \sum_i \max\Big(\frac{s_i}{D} - \lambda, 0\Big) = 1 - \frac{1}{D}, & \textit{if } S \geq D \end{cases},$$

where $\mathbf{e}^w = (e_1^w, \ldots, e_n^w) \in \{0,1\}^N$ is a vector such that $e_w^w = 1$ and $e_i^w = 0\ \forall i \in N \setminus \{w\}$, s_i is the number of shares reported by miner i, $S = \sum_{i \in N} s_i$ is the total number of reported shares in a round and D is the crypto-puzzle difficulty.

We assign to agent w, who finds the solution during a long round, an extra prize of $\frac{1}{D}$ to add to the allocation established by the classical CEL rule for the bankruptcy situation $(\mathbf{c}, E) = (\frac{1}{D} \cdot \mathbf{s}, 1 - \frac{e_i^w}{D})$, with the estate reduced by $\frac{1}{D}$. More precisely, in long rounds $\widehat{R}_i(\mathbf{s}; w) = \frac{e_i^w}{D} + CEL_i(\frac{1}{D} \cdot \mathbf{s}, 1 - \frac{e_i^w}{D})$. In other words, it means that 1 is added to the count of the shares s_i reported by the full solution discoverer w. If the value $\frac{s_i}{D} - \lambda$ is negative, by default, the agent w is receiving $\frac{1}{D}$. This incentive is sufficient to make the reward function incentive compatible.

Before proving this statement, let us compare the allocations provided by the classical CEL rule and $\widehat{R}$ through an example with $n = 3$, $D = 10$ and $E = 1$.

Example 3.3.1 *Given the following bankruptcy situation :* $\mathbf{s} = (2, 7, 8)$*, miner* 1 *finds the full solution* ($w = 1$) *and* $(\mathbf{c}, E) = ((0.2, 0.7, 0.8), 1)$*. By Definition 2.13, it is easy to check that* $\lambda = 0.25$*, hence :*

$$CEL(\mathbf{c}, E) = (0, 0.45, 0.55).$$

Now, consider the new CEL-based rule $\widehat{R}$, a prize of $\frac{1}{D} = 0.1$ is allocated to miner 1*, and a new bankruptcy situation $(\mathbf{c}, E')$ arises where the estate is reduced by* 0.1 *;* $(\mathbf{c}, E') = ((0.2, 0.7, 0.8), 0.9)$*. By Definition 5.21, now $\lambda = 0.3$ and we have that :*

$$\widehat{R}((2,7,8);1) = (0.1, 0, 0) + CEL((0.2, 0.7, 0.8), 0.9) = (0.1, 0, 0) + (0, 0.4, 0.5) = (0.1, 0.4, 0.5).$$

In order to prove that the CEL-based rule is incentive compatible we need to present some preliminary results. More precisely, to express Condition (3.2) for this new reward function we need to focus on the parameter λ of the definition. This parameter depends on miners' demands and it changes value from round to round. It is important to analyze how the parameter varies if an additional share is found. Let us denote by :

(i) λ_1 the value of the parameter λ when miner i finds the full solution and immediately reports it to the pool and,

(ii) λ_2 the value of the parameter after delaying in reporting the full solution by one additional share. By convention, if miner i finds the additional share the parameter is denoted as λ_2^1, while if any other miner finds it we have λ_2^2.

By analyzing the different values of the parameter λ it is possible to derive the following result :

Proposition 3.1 *Let us consider* $CEL_i(\mathbf{c}, E) = \max(c_i - \lambda_1, 0)$ *and* $CEL_i(\mathbf{c}+\mathbf{e}_j, E) = \max(c_i^{'} - \lambda_2, 0)$. *For each* $(\mathbf{c}, E) \in \mathbb{B}^N, i, j \in N$ *we have that* $\lambda_1 \leq \lambda_2$.

Proof. *Let us report the efficiency condition for the two allocations :*

$$\max(c_j - \lambda_1, 0) + \sum_{i \in N \setminus \{j\}} \max(c_i - \lambda_1, 0) = \, = \max(c_j + 1 - \lambda_2, 0) + \sum_{i \in N \setminus \{j\}} \max(c_i - \lambda_2, 0).$$

If $c_j \leq \lambda_1$, *efficiency condition implies that* $\sum_{i \in N \setminus \{j\}} \max(c_i - \lambda_1, 0) \geq \sum_{i \in N \setminus \{j\}} \max(c_i - \lambda_2, 0)$. *Hence,* $\lambda_1 \leq \lambda_2$. *For* $c_j > \lambda_1$ *let us assume, by contradiction, that* $\lambda_1 > \lambda_2$. *The assumption implies that* $\sum_{i \in N \setminus \{j\}} \max(c_i - \lambda_1, 0) \leq \sum_{i \in N \setminus \{j\}} \max(c_i - \lambda_2, 0)$. *However,* $\max(c_j - \lambda_1, 0) = c_j - \lambda_1 < c_j - \lambda_2 < c_j + 1 - \lambda_2 = \max(c_j + 1 - \lambda_2, 0)$ *and this leads to contradiction.* ■

Corollary 3.0.1 *Given the situation of Proposition 3.1 we have that :* $\lambda_2 - \frac{1}{D} \leq \lambda_1 \leq \lambda_2$.

Now, we are ready to prove the incentive compatibility of the new reward function based on the CEL rule.

Proposition 3.2 *The CEL-based reward function* $\widehat{R}$ *of Definition 5.21 satisfies the property of incentive compatibility.*

***Proof**. Let us write down Condition* (3.2) *for* $\widehat{R}$ *:*

$$\alpha_i\left(\frac{1}{D}+\max\left(\frac{s_i+1}{D}-\lambda_2^1,0\right)-\frac{1}{D}-\max\left(\frac{s_i}{D}-\lambda_1,0\right)\right)$$
$$+(1-\alpha_i)\left(\max\left(\frac{s_i}{D}-\lambda_2^2,0\right)-\frac{1}{D}-\max\left(\frac{s_i}{D}-\lambda_1,0\right)\right)\leq\frac{\mathbb{E}_s[\widehat{R}_i(\mathbf{s};w)]}{D}.$$

Since the average reward $\mathbb{E}_s[\widehat{R}_i(\mathbf{s};w)]$ *is positive, the right hand side is positive. Therefore, the condition is fulfilled if the left hand side is not positive.*
Due to Proposition 3.1 and Corollary 3.0.1 we have that : $\frac{s_i}{D}-\lambda_2^2\leq\frac{s_i}{D}-\lambda_1\leq\frac{s_i+1}{D}-\lambda_2^1$.
If all the terms in the form $\max(\cdot,0)$ *are positive, then the condition is fulfilled :*

$$\alpha_i\left(\frac{1}{D}-\lambda_2^1+\lambda_1\right)+(1-\alpha_i)\left(-\lambda_2^2-\frac{1}{D}+\lambda_1\right)\leq\max(\alpha_i,1-\alpha_i)(-\lambda_2^1-\lambda_2^2+2\lambda_1)\leq 0.$$

If $\frac{s_i}{D}-\lambda_2^2\leq 0\leq\frac{s_i}{D}-\lambda_1$ *we get :*

$$\alpha_i\left(\frac{1}{D}-\lambda_2^1+\lambda_1\right)+(1-\alpha_i)\left(-\frac{s_i}{D}-\frac{1}{D}+\lambda_1\right)\leq\max(\alpha_i,1-\alpha_i)(-\lambda_2^1-\frac{s_i}{D}+2\lambda_1)\leq 0.$$

If $\frac{s_i}{D}-\lambda_1\leq 0\leq\frac{s_i+1}{D}-\lambda_2^1$ *we get :*

$$\alpha_i\left(\frac{s_i+1}{D}-\lambda_2^1\right)+(1-\alpha_i)\left(-\frac{1}{D}\right)\leq\max(\alpha_i,1-\alpha_i)\left(\frac{s_i}{D}-\lambda_2^1\right)\leq\max(\alpha_i,1-\alpha_i)\left(\frac{s_i}{D}-\lambda_1\right)\leq 0.$$

In the end, if all the terms in the form $\max(\cdot,0)$ *are equal to* 0*, then the condition is fulfilled, since the left hand side is negative.* ■

3.3.3 A Multi-Pool Analysis

Pool-hopping consists of a practice in which miners leave a pool to join another one that is considered more attractive in terms of remuneration. More precisely, during a round a miner performing pool-hopping (i.e., a *hopper*) stops submitting shares to the pool she was working with at the beginning of the round and starts submitting shares to a different one. A hopper leaves, during a mining race, a pool entering (or already in) a long round for a pool that is currently in a short round. The hopping miner receives an increasing reward from the brand new pool (in short round) and a decreasing reward from the pool left (facing a bankruptcy situation where the resource B is insufficient to remunerate the working miners).

In a multi-pool framework, the total mining power of the network is represented by different mining pools each with its own computational power. Differently from Section 3.3.1, each miner $i\in N$

is characterized by α_i that now represents a fraction of the pool hashing rate. Indeed, in the single-pool framework, we denote with α_i the fraction of the total hashing power.

Hopping affects the actual rewards of a pool. If a miner performs pool-hopping the pool loses computational power and so on average the full solution is found later, i.e., the rounds become longer.

In our multi-pool analysis, we assume that pool-hopping is performed at the very beginning of a long round and that miners hop between two pools adopting the same rewarding mechanism. Every mean denoted as $\mathbb{E}[\cdot]$ is considered conditioned to the fact that the miner is in a long round : $\mathbb{E}_{\mathbf{s}}[\cdot|S > D]$. From now on, we mark with an asterisk (*) every variable defining the reward of miners once pool-hopping is performed.

3.3.3.1 Hopping Analysis on Schrijver's Rewarding Function

When miner i is remunerated with reward function R her incentive to perform pool-hopping can be measured as the difference between (i) the average reward when hopping $\mathbb{E}[R_i^*]$ and (ii) the average reward $\mathbb{E}[R_i]$ when working for the pool :

$$\delta_{hop} := \mathbb{E}[R_i^*] - \mathbb{E}[R_i].$$

Proposition 3.3 *The reward function R proposed by Schrijvers et. al. always gives miners a positive incentive $\delta_{hop} > 0$ to perform pool-hopping.*

Proof.*As shown in [253], the average reward of an honest miner $i \in N$, i.e., not hopping, is :*

$$\mathbb{E}[R_i] = \alpha_i.$$

A hopper (hopping at time t) receives an increasing reward from the new pool in a short round and a decreasing one from the pool left. The sum of the two represents the total reward. On average at the end of a short round ($S = D$) a miner has found $\alpha_i \cdot D$ shares. The round finishes after $D + t$ shares are found, with $t \in [0, +\infty)$, hence the reward for the miner who performs pool-hopping is the following :

$$\mathbb{E}[R_i^*] = \sum_{t=0}^{\infty} \left(\frac{\alpha_i t}{D} + \frac{\alpha_i D}{D+t} \right) p'(1-p')^t > \frac{\alpha_i}{1-\alpha_i} > \alpha_i,$$

where $p' = \frac{1-\alpha_i}{D}$ is the probability that a share found by an honest miner is a full solution, t is the time taken by an honest miner (working for the old pool) to find a new share and R_i^* is the reward obtained *by a miner who hops from a pool rewarding with R to another pool using the same reward function. Hence, the incentive to perform pool-hopping is always positive :*

$$\delta_{hop} = \mathbb{E}[R_i^*] - \mathbb{E}[R_i] > \alpha_i - \alpha_i = 0.$$

■

A second result deriving from Proposition 5.6 is the fact that, on average, the hopping miners gain more than their hashing ratio α_i. This has been empirically verified on the Bitcoin network in the contribution presented in Section 3.2. The average reward for a hopper, between pools adopting R, can be analytically computed according to the following result.

Proposition 3.4 *The average reward of miner $i \in N$ hopping between two different pools remunerating miners according to the reward function R is the following :*

$$\mathbb{E}[R_i^*] = \frac{\alpha_i}{1-\alpha_i} + \alpha_i^*,$$

where $\alpha_i^* = \alpha_i(1-\alpha_i)e^{1-\alpha_i}(-Ei(\alpha_i - 1))$ with $Ei(x) = \int_{-\infty}^{x} \frac{e^t}{t}$ denoting the exponential integral.

Proof. *On average, a miner with computational power α_i who performs pool-hopping receives :*

$$\mathbb{E}[R_i^*] = \sum_{t=0}^{\infty} \frac{\alpha_i t}{D} p'(1-p')^t + \sum_{t=0}^{\infty} \frac{\alpha_i D}{D+t} p'(1-p')^t.$$

The first term represents the reward received by the new pool in short round that can be easily computed as follows :

$$\frac{\alpha_i}{D} \sum_{t=0}^{\infty} t \cdot p'(1-p')^t = \frac{\alpha_i}{D} \cdot \frac{D}{1-\alpha_i} = \frac{\alpha_i}{1-\alpha_i}.$$

The second term (denoted as α_i^*) corresponds to the reward assigned by the pool left by the hopping miner. In order to compute this term we need to consider an approximation for $D \to \infty$:

$$\alpha_i^* = \sum_{t=0}^{\infty} \frac{\alpha_i D}{D+t} \frac{1-\alpha_i}{D} \left(1 - \frac{1-\alpha_i}{D}\right)^t \approx \alpha_i(1-\alpha_i)e^{1-\alpha_i} \sum_{t=D}^{\infty} \frac{1}{t} e^{-\frac{1-\alpha_i}{D}t}.$$

The computations can be solved by defining :

$$f(x) := \lim_{D\to\infty} f_D(x) = \lim_{D\to\infty} \sum_{t=D}^{\infty} \frac{1}{t} e^{-\frac{tx}{D}}.$$

Using Lebesgue's theorem and given the constraint $\lim_{x\to\infty} f(x) = 0$ *we get :*

$$f(x) = -Ei(-x).$$

Hence :

$$\alpha_i^* \approx \alpha_i(1-\alpha_i)e^{1-\alpha_i}f(1-\alpha_i) = \alpha_i(1-\alpha_i)e^{1-\alpha_i}(-Ei(\alpha_i - 1)).$$

■

Thanks to the result provided by Proposition 3.4 we note that the shares submitted in the pool left by the hopping miner (α_i^*) represent an important part of her average reward. More precisely, for values of $\alpha_i < 0.39$, α_i^* is more than the 50% of the average reward a miner would have got by not leaving the pool (i.e., her computational power α_i). For instance, if a miner has $\alpha_i = 0.2$ as computational power, she will get $\alpha_i^* \approx 0.11$.

3.3.3.2 Hopping Analysis on CEL-based Rewarding Function

Following similar arguments, we can analyze the incentive to perform pool-hopping when adopting the CEL-based rule $\hat{R}$. We can, then, compare the results obtained for the reward function R with the ones provided by $\widehat{R}$. We denote as β_i the average reward of function $\widehat{R}$ (corresponding to α_i for function R) that can be computed as follows since the probability for a miner i to find a full solution and to receive the extra prize $\frac{1}{D}$ is α_i :

$$\beta_i = \mathbb{E}[\widehat{R}_i] = \frac{\alpha_i}{D} + \mathbb{E}\left[\max\left(\frac{s_i}{D} - \lambda, 0\right)\right] \quad \lambda : \sum_i \max\left(\frac{s_i}{D} - \lambda, 0\right) = 1 - \frac{1}{D}.$$

Like in Section 3.3.3.1, let us define the incentive to perform pool-hopping $\widehat{\delta}_{hop} := \mathbb{E}[\widehat{R}_i^*] - \mathbb{E}[\widehat{R}_i]$ and let us compute the average reward received by a hopper :

$$\mathbb{E}[\widehat{R}_i^*] = \sum_{t=0}^{\infty}\left(\frac{\alpha_i t}{D} + \max\left(\frac{\alpha_i D}{D} - \lambda, 0\right)\right)p'(1-p')^t = \frac{\alpha_i}{1-\alpha_i} + \mathbb{E}\left[\max(\alpha_i - \lambda, 0)\right],$$

where $p' = \frac{1-\alpha_i}{D}$, t is the time taken by an honest miner to find a new share and $\widehat{R}_i^*$ is the reward obtained by a hopping miner.

Analogously to α_i^* for function R, we denote by β_i^* the reward given by the pool the hopper left :

$$\beta_i^* = \mathbb{E}\left[\max(\alpha_i - \lambda, 0)\right].$$

Hence we have that :

$$\widehat{\delta}_{\text{hop}} = \mathbb{E}[\widehat{R}_i^*] - \mathbb{E}[\widehat{R}_i] = \frac{\alpha_i}{1-\alpha_i} + \beta_i^* - \beta_i.$$

3.3.3.3 Comparison of the Rewarding Functions in a Multi-Pool Framework

We have, now, the metrics to compare the performance of the reward functions R and $\widehat{R}$ in hopping situations. Both rewarding systems present an incentive to hop in long rounds, however the miner rewarded with the CEL-based reward function are less incentivized. It is possible to compare the incentives $\delta_{\text{hop}}, \hat{\delta}_{\text{hop}}$ given by the two functions $R, \widehat{R}$ through the variables introduced in Section 3.3.3.2 since :

$$\hat{\delta}_{\text{hop}} \leq \delta_{\text{hop}} \Leftrightarrow \beta_i - \beta_i^* \geq \alpha_i - \alpha_i^*.$$

In order to show that the hopping incentive for the CEL-based reward function is lower with respect to the incentive given by R it is sufficient to prove that $\beta_i - \beta_i^* \geq \alpha_i - \alpha_i^*$.

Proposition 3.5 *Let N be the ordered set of miners : $\alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_n$, let us define $\alpha_{>i} := \sum_{j>i} \alpha_j$, as the global computational powers of the miners who are more powerful than α_i. Then, $\beta_i - \beta_i^* \geq \alpha_i - \alpha_i^*$ if $(1-\alpha_i)(\alpha_{>i} - \alpha_i)e^{-\alpha_i + (\alpha_{>i} - \alpha_i)^{-1}}(-Ei(\alpha_i - 1)) \geq 1$ where $Ei(\cdot)$ is the exponential integral function.*

Proof. *Given the definitions of β_i and β_i^* :*

$$\beta_i = \frac{\alpha_i}{D} + \sum_{t=0}^{\infty} \max\left(\alpha_i + \frac{\alpha_i t}{D} - \lambda, 0\right) p(1-p)^t \quad \text{and} \quad \beta_i^* = \sum_{t=0}^{\infty} \max\left(\alpha_i - \lambda, 0\right) p'(1-p')^t,$$

let us recall that $p = \frac{1}{D}$ is the probability for a share to be a full solution and that $p' = \frac{1-\alpha_i}{D}$ represents the probability for a share reported by an honest miner to be a full solution.

For $t \to \infty$ (i.e., for very long rounds) the function $\max(\cdot, 0)$ either tends to 0 or to 1 since in long rounds eventually the most powerful miner is receiving all the reward ($\max(\cdot, 0) \to 1$) and the other miners are receiving none of it ($\max(\cdot, 0) \to 0$). The limit value 0 is reached for $t = \gamma_i \cdot D$, where $\gamma_i \in \mathbb{R} \cup \{+\infty\}$ is defined as follows ; $\gamma_i := \text{argmin}_\gamma \{\frac{\alpha_i t}{D} - \lambda < 0, \forall t \geq \gamma D\}$. Indeed, $\gamma_i \cdot D$ represents the number of shares after which miner i is not rewarded.
Hence, it is possible to rewrite β_i and β_i^* in this form :

$$\beta_i = \frac{\alpha_i}{D} + \sum_{t=0}^{\gamma_i D} \left(\alpha_i + \frac{\alpha_i t}{D} - \lambda\right) p(1-p)^t \quad \text{and} \quad \beta_i^* = \sum_{t=0}^{\gamma_i D} (\alpha_i - \lambda)\, p'(1-p')^t.$$

The value of γ_i might change if the miner is performing pool-hopping, but for the sake of simplicity we approximate by considering the same γ_i in both cases.

Assuming that $\sum_t \lambda \cdot p(1-p)^t \approx \sum_t \lambda \cdot p'(1-p')^t$ we can approximate the difference between β_i

and β_i^* as follows :

$$\beta_i - \beta_i^* \approx \frac{\alpha_i}{D} + \sum_{t=0}^{\gamma_i D} \frac{\alpha_i t}{D} p(1-p)^t.$$

Due to the value of the difficulty D, we can consider the limit for $D \to \infty$, then :

$$\beta_i - \beta_i^* \approx \frac{\alpha_i}{D} + \sum_{k=0}^{\gamma_i D} \frac{\alpha_i k}{D}\frac{1}{D} e^{-k/D} = \frac{\alpha_i}{D} + \alpha_i \frac{1}{D^2} \sum_{k=0}^{\gamma_i D} k e^{-k/D} \to \alpha_i(1 - e^{-\gamma_i}(1+\gamma_i)).$$

Let us now compute explicitly γ_i. If $i = N$ (i.e., $\alpha_i = \operatorname{argmax}_j\{\alpha_j\}$) then $\gamma_i = \infty$. Otherwise at time $t = \gamma_i \cdot D$ the miners who receive a positive reward are all the $j \in N : \alpha_j > \alpha_i$, i.e., all the ones having larger computational power than i. According to the CEL rule definition we get the following balance equation :

$$\sum_{j>i} \left(\alpha_j \left(1 + \frac{t}{D} \right) - \lambda \right) = 1 - \frac{1}{D} \approx 1.$$

Since the time $t = \gamma_i \cdot D$ is the moment when the value $\max(\alpha_i(1+\frac{t}{D}) - \lambda, 0)$ turns from positive to null, we can say that $\alpha_i(1 + \frac{t}{D}) - \lambda \approx 0$. Therefore we have that :

$$\sum_{j>i} \left(\alpha_j \left(1 + \frac{t}{D} \right) - \alpha_i \left(1 + \frac{t}{D} \right) \right) = 1.$$

Replacing the value of t with $\gamma_i \cdot D$ we get $(\alpha_{>i} - (N-i)\alpha_i)(1+\gamma_i) = 1$, then :

$$\gamma_i = ((\alpha_{>i} - (N-i)\alpha_i)^{-1} - 1).$$

Now we can find a lower bound for γ_i (since $N - i \geq 1$) and so for $\beta_i - \beta_i^*$:

$$\gamma_i \geq \bar{\gamma}_i := ((\alpha_{>i} - \alpha_i)^{-1} - 1) \Longrightarrow \beta_i - \beta_i^* \geq \alpha_i(1 - e^{-\bar{\gamma}_i}(1 + \bar{\gamma}_i)).$$

The sufficient condition for $\beta_i - \beta_i^* \geq \alpha_i - \alpha_i^*$ is :

$$\alpha_i(1 - e^{-\bar{\gamma}_i}(1 + \bar{\gamma}_i) \geq \alpha_i - \alpha_i^*.$$

We get the statement of the proposition by replacing $\bar{\gamma}_i$ and α_i^* with their explicit formulas. ■

Thanks to Proposition 3.5, given miner i's hashing ratio (i.e., α_i) and the power of the miners who are stronger than i (i.e., $\alpha_{>i}$), we can check whether $\widehat{R}$ is giving a lower hopping incentive than the one given by R (i.e., check whether $\hat{\delta}_{\text{hop}} \leq \delta_{\text{hop}}$) by simply applying the sufficient condition introduced above that we denote as $f(\alpha_i, \alpha_{>i})$:

$$f(\alpha_i, \alpha_{>i}) := (1-\alpha_i)(\alpha_{>i} - \alpha_i)e^{-\alpha_i + (\alpha_{>i} - \alpha_i)^{-1}}(-Ei(\alpha_i - 1)) \geq 1.$$

Let us analyze the hopping performance of R and $\widehat{R}$ in the following example.

Example 3.3.2 *Given 5 miners ordered according to their hash rates :* $\alpha_1 = 0.10$ $\alpha_2 = 0.15$, $\alpha_3 = 0.20$, $\alpha_4 = 0.25$, $\alpha_5 = 0.30$, *using the condition provided by Proposition 3.5 we get :*

- $f(\alpha_1, \alpha_{>1}) = f(0.10, 0.90) = 0.59 < 1$, *miner* 1 *has a greater incentive to perform pool-hopping if rewarded with* $\widehat{R}$ *rather than with* R *;*
- $f(\alpha_2, \alpha_{>2}) = f(0.15, 0.75) = 0.66 < 1$, *miner* 2 *has a greater incentive to perform pool-hopping if rewarded with* $\widehat{R}$ *rather than with* R *;*
- $f(\alpha_3, \alpha_{>3}) = f(0.20, 0.55) = 1.24 > 1$, *miner* 3 *has a greater incentive to hop if rewarded with* R *rather than with* $\widehat{R}$ *;*
- $f(\alpha_4, \alpha_{>4}) = f(0.25, 0.30) > 10^6 > 1$, *miner* 4 *has a greater incentive to hop if rewarded with* R *rather than with* $\widehat{R}$ *;*
- $f(\alpha_5, \alpha_{>5}) = f(0.30, 0) \to \infty > 1$, *miner* 5 *has a really low incentive to perform pool-hopping if rewarded with* $\widehat{R}$*.*

We can see that miners representing the 75% *of the pool's computational power have a lower incentive to perform pool-hopping when the CEL-based rewarding mechanism is adopted.*

By analyzing function $f(\cdot, \alpha_{>i})$ – i.e., fixing $\alpha_{>i}$ – we can identify the cases in which $\hat{\delta}_{\text{hop}} \leq \delta_{\text{hop}}$ (where $\widehat{R}$ performs better than R). For instance, $f(\cdot, \alpha_{>i}) > 1$ for every $\alpha_{>i} < 0.4$, means that with $\widehat{R}$ not only the miners representing the most powerful 40% of the pool have a lower incentive to perform pool-hopping, but also the miner i who just follows in the ranking.

To compare the two reward functions, it is necessary to estimate the percentage of miners who have a lower incentive to perform pool-hopping. In Example 3.3.2 this percentage is computed as follows : $p(\alpha = \{0.1, 0.15, 0.2, 0.25, 0.3\}) = 75\%$. Formally we would like to estimate :

$$p(\alpha) := \sum_i \alpha_i \cdot \mathbf{1}_{\{\beta_i - \beta_i^* \geq \alpha_i - \alpha_i^*\}}.$$

We know that $p(\alpha) > 40\%$ thanks to the analysis of function f. In order to get a better idea of the range of the value of function p we perform a simulation.

Simulation. Due to the unpredictability of α, we assume that it comes from a random distribution.

More precisely, given $X_i \sim U[0,1]$, α_i is defined as follows : $\alpha_i := \frac{X_i}{\sum_j X_j}$.
We run a simulation with 100 different samples of α for n miners, with $n \in \{3, 10, 20, 30, 50\}$, and estimate the CDF of $p_n(\alpha)$ for every n. We compute explicitly β_i and β_i^*, without using the approximation above introduced.

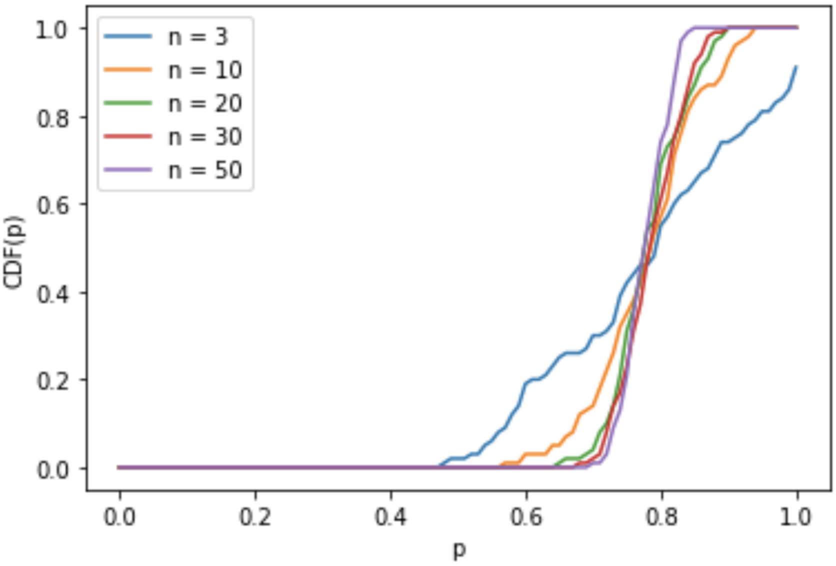


FIGURE 3.12 – CDF of every $p_n(\alpha)$, with $n \in \{3, 10, 20, 30, 50\}$.

The functions $p_n(\alpha)$ have *almost always* values over 0.5 (i.e., in just two cases out of 100 with $n = 3$, $p_3(\alpha)$ achieves value between 0.47 and 0.5).
This means that in most of the cases the majority of the miners have a lower incentive to perform pool-hopping with $\widehat{R}$ rather than R.

Conclusions. The contribution analyzes the robustness of two different rewarding mechanisms in both intra-pool and inter-pool environments. The reward function introduces in [253] is incentive compatible but not hopping proof. By reinterpreting R, in long rounds, as an allocation rule for a bankruptcy situation, we created a new rewarding function $\widehat{R}$ inspired to the well-known Constrained Equal Loss (CEL) rule. We show that this CEL-based rule is incentive compatible as R but it provides to most of the miners a lower incentive to perform pool-hopping in long rounds. In conclusion, if a pool wants to tackle this issue, the proposed rewarding function $\widehat{R}$ is the one to be recommended.

3.4 Summary

Summary. This chapter analyzes a particular type of blockchain users, i.e., validating nodes of the Bitcoin blockchain, who act in a rational manner driven in their choices by the mere intent of gaining a higher reward. The first contribution presents evidence of Bitcoin miners' hopping propensity emphasizing the rationality of miners behaviors. In the second contribution we model the rewarding problem as a bankruptcy game with the scope of identifying a reward function that can prevent both block-withholding and pool-hopping behaviors. In reality, Bitcoin and more in general blockchain systems are subject to countless attacks on a daily basis related to their design constructs, the peer-to-peer architecture and their application [266]. Rationality cannot always explain these behaviors; some blockchain users attack the system with the simple intent of compromising and/or destroying it [267], even if this comes at a cost. Hence, the presence of these Byzantine validators, who behave in a malicious manner no matter what, needs to be considered in assessing the robustness of blockchain systems (see Chapter 5).

Chapter 4

Blockchain Interoperability between Rational Blockchain Users

Content

This chapter is devoted to the analysis of (i) distributed cross-chain swap problem in the blockchain context and (ii) the behaviors of *transacting parties* aiming at exchanging assets through cross-chain swaps. Cross-chain swaps enable interactions between multiple blockchain i.e., blockchain interoperability. We start by presenting a mathematical framework allowing to characterize blockchain swap protocols. Game theory is then used to model transacting parties (who swap crypto-assets) as rational agents and characterize the equilibria of existing cross-chain protocols.

4.1 Introduction to Blockchain Interoperability and Cross-Chain Problems

The modern economy is moving to a new era where economical transactions use crypto-currencies instead of fiat money. Crypto-currencies (i.e., Bitcoin, Ethereum, etc.) are based on the use of blockchains, which are basically transactional systems governed by decentralized protocols. Blockchains pave the way for a new approach to organize and sustainably maintain long-term transactions as well as high-level services. It is interesting to note that the number of blockchains that currently hold the head of newspapers has gone from one in 2008 (the famous Bitcoin blockchain [20]) to a few tens in 2018 such as Hyperledger [51], Ethereum [58], Zcash [73], Corda [97], Ripple [76], Tendermint [1] etc. Each of these systems has its own modus operandi, its own governance and even its own way of agreeing on a common history. Each system has its own advantages that make it attractive for various applications and geopolitical contexts. We are witnessing the creation of several ecosystems, each with its own currency and governance.

Similar to modern international economical exchanges which are based on different government-issued currencies, inter-blockchain exchanges must be based on common rules resilient to attacks, failures or malicious behaviors affecting the network. There are currently several operational systems for achieving interoperability between different blockchains such as Kybernetwork [268], Aion [269], Cosmos [270] or Polkadot [271]. These systems can be classified into two categories according to their decentralization level : systems that use a trusted third-party to validate transactions or systems that realize it directly between blockchains without the need of a trusted third-party. In order to execute an exchange or a *swap* (i.e., a set of transactions between parties), transacting agents (i.e., blockchain users) are provided with a protocol to stick to. A protocol in this case consists of a specific sequence of instructions agents should perform to preserve the ACID properties [272] of the individual transactions or exchanges ; that is, *Atomicity* : the all-or-nothing occurs and each participant must know which state he or she is in ; *Consistency* : each successful transaction by definition commits only valid results ; *Isolation* : transactions run independently ; *Durability* : transactions cannot be abrogated after commitment.

Atomic cross-chain swaps fall into the class of the so called TAST (i.e., Token Atomic Swap Technology) [273] research ideas aiming at making blockchains interoperable. Differently from decentralized exchanges – initiatives recently emerged to remove the need of trust on traditional exchange services (e.g., the format proposed by the Ethereum DEX protocols [274; 275] atomically swapping ERC-20 tokens on the Ethereum blockchain) – atomic swaps have not limits in operating cross-chain. The very first atomic swap solution has been proposed for Bitcoin by *Nolan* [5] making use of hash-time locked contracts enabling conditional assets transfers. Nowadays few platforms actually support cross-chain exchanges that at this stage are still slow and inefficient. *Decred* [276] implements Nolan's logic on UTXO-based premissionless blockchains such as Bitcoin Cash [277; 278; 279], Litecoin [280], Qtum [281], etc. *BartherDEX* [282], part of the Komodo project [283], represents a cross-chain solution that matches orders and defines the swap protocol. *Blockchain.io* [284] implements atomic cross-chain swaps by combining centralized components (order matching) with decentralized ones (trade settlement and execution). Therefore, research now focuses on *hybrid* swap protocols, replacing decentralized commitment/locking schemes (hash-locks) with centralized ones, resulting more attractive and efficient. *AC3TW* and *AC3WN* [7] protocols propose atomic cross-chain swaps respectively with centralized and "decentralized" trusted authorities (i.e., an external agent and an external blockchain) acting as witnesses. According to *Arwen* protocol [285] crypto-assets are swapped through centralized exchange services that in no way acquire the assets custody. *XClaim* [286] overcomes blockchain data-structures incompatibility by swapping cryptocurrency-backed assets.

It should be noted that different swap protocols differ essentially in the involved parties. The set of swap participants can be composed only of the asset owners (e.g., as in [6]) or by owners accompanied by a trusted third party (e.g., as in the AC3TW protocol [7]).

To the best of our knowledge, there is (i) no formal analysis on the structure and the properties blockchain swap protocols satisfy and, (ii) no game theoretical modeling of participants strategic interactions (i.e., to follow or not to follow the prescribed protocol). More precisely, *Nolan* [5] presents the Bitcoin swap protocol in a functional manner. In [6] the author provides a partial game theoretical analysis (specific to the protocol) presenting no structural characterization of the possible equilibria of the system (see Section 4.4.1). Authors in [7] analyze the atomicity violations characterizing the protocol in [5]. However, no game theoretical result is provided.

According to recent studies [287; 288] proposed swap protocols are not properly analyzed neither from the structural point of view, responsible for their atomicity, nor from the strategical point of view making them satisfying *liveness* and *safety* properties (i.e., the protocol terminates in a valid state). More precisely, it is important to analyze the behaviors of the swap participants who can be considered as rational agents actively participating in the exchange by following or not the prescribed protocol according to their own objective function.

We propose, in Section 4.2, a generic game theoretical framework that formalizes the swap problem and characterizes blockchain swap protocols by clearly separating the contracts publishing phase and their commitment phase. Furthermore, we prove in Section 4.3 that (i) following a swap protocol characterized by an *effective* decision function (when players have the power to accept or decline the desired assets) is a subgame perfect equilibrium in dominant strategies and, that (ii) following a swap protocol characterized by concurrent publishing and *snap* (immediate) commitment is a Nash equilibrium. Our generic framework allows us to characterize, in Section 4.4, equilibria of two representative recent protocols presented in [5] and [7], respectively. In the case of the protocol proposed in [5] and generalized in [6], following the protocol is the unique subgame perfect equilibrium (in dominant strategies), while in the case of the protocol proposed in [7], following the protocol is a Nash equilibrium.

4.2 Swap Problems and Swap Protocols

In this section we propose a formal definition of the *swap* problem and a formalization of the corresponding *blockchain swap protocol* that can be atomic or not. The latter consists of two different phases (i.e., publishing and commitment of transfers). In a general *swap problem*, swapping parties aim at exchanging assets among themselves. A *swap protocol* defines the set of asset transfers and the order in which they should be executed. Given a set of assets and the corresponding owners, a *swap* consists of an asset ownership exchange within the set of owners.

Definition 4.1 (swap problem) *A swap problem is defined as a tuple* $\langle \mathcal{A}, \mathcal{O}, b_0, b_*, (u_i)_{i\in\mathcal{O}} \rangle$ *where :*

- $\mathcal{A} = \{1, \ldots, m\}$ *is the set of assets to be swapped ;*
- $\mathcal{O} = \{1, \ldots, n\}$ *is the set of owners or agents participating in the exchange. We consider in the following that* $n \leq m$ *since each owner owns at least one asset ;*
- $b_0, b_* : \mathcal{A} \to \mathcal{O}$ *are the original and the desired ownership maps – both surjective – such that* $\forall a \in \mathcal{A},\ b_0(a) \neq b_*(a)$ *and ;*
- u_i *is the payoff function for owner* $i \in \mathcal{O}$ *over bundles of assets in* $2^{\mathcal{A}}$ *such that* $u_i(b_0^{-1}(i)) < u_i(b_*^{-1}(i))$ *(i.e., each owner i strictly prefers the desired bundle of assets to her original bundle) and* $\forall S, T \in 2^{\mathcal{A}} : S \subseteq T,\ u_i(T) \geq u_i(S)$ *(i.e., a larger bundle is strictly preferred to a smaller one), for each* $i \in \mathcal{O}$*.*

The representation with two different surjective functions b_0, b_* describes swaps as ownership exchanges problems. Agents participate in a swap with the asset(s) they aim to exchange for others, preferring the desired new asset(s) to finding themselves as in the initial configuration. Moreover, we assume owners' payoff function increases monotonically with the size of the asset bundle.

The transition from an initial configuration to a post-swap configuration is defined by the corresponding *swap protocol* consisting in a sequence of operations to be executed in a certain order. In centralized swap protocols a central role in the swap is played by a *trusted third party.* That is, asset ownership is transferred first to the trusted third party that in turn transfers assets back to the new owners. On the other hand, decentralized swap protocols contemplate ownership transfers within asset owners only ; the latter agrees on a particular swap configuration (i.e., the assets to exchange) without trusting each other.

In order to formally define a swap protocol we need to introduce first, the structure of a *decentralized exchange protocol*, consisting of a sequence of asset(s) transfers to be committed.

Definition 4.2 (decentralized exchange protocol) *Let* $\sigma = \{(A^k, O^k, X^k) : |A^k| \geq |O^k|\}_k$, $k \in \{1, \ldots, t\}$, $t \in \mathbb{N} : t \leq m$ *be a sequence of exchanges where,*

- $A^k \subseteq \mathcal{A}$ *specifies the subset of assets involved in the exchange at step* k *;*
- $O^k \subseteq \mathcal{O}$ *specifies the subset of owners involved in the exchange at step* k *;*
- $X^k : A^k \to O^k$ *(surjective) specifies the owner* $X^k(a) \in O^k$ *of any asset* $a \in A^K$ *at step* k *;*

A sequence σ *defines a decentralized exchange protocol that engenders a sequence of maps* $b_1^\sigma, b_2^\sigma, \ldots, b_t^\sigma :$ $\mathcal{A} \to \mathcal{O}$ *such that for all* $k \in \{1, 2, \ldots, t\}$ *:*

— $b_k^\sigma(z) = b_{k-1}^\sigma(z),\ \forall z \in \mathcal{A} \setminus A^k$ *;*

— $b_k^\sigma(z) = X^k(z),\ \forall z \in A^k$*,*

where we set $b_0^\sigma = b_0$*.*

So, the triple (A^k, O^k, X^k) specifies that the asset set $A^k \subseteq \mathcal{A}$ is transferred at step k to owners $O^k \subseteq \mathcal{O}$ according to X^k. Note that, b_k^σ and b_{k-1}^σ differ only for the ownership of assets A^k belonging to $b_{k-1}^\sigma(A^k)$ at step $k-1$ and to O^k at step k.

Example 4.2.1 *Let us consider the swap problem* $\langle \mathcal{A}, \mathcal{O}, b_0, b_*, (u_i)_{i \in \mathcal{O}} \rangle$ *such that the set of assets is* $\mathcal{A} = \{a, b, c, d, e\}$*, the set of owners is* $\mathcal{O} = \{1, 2, 3\}$*, the original ownership map is* $b_0 = (1, 1, 2, 3, 3)$ *(meaning that the asset ownership is :* 1 *for assets* a *and* b*,* 2 *for* c *and* 3 *for* d *and* e*) and the desired ownership map is* $b_* = (2, 3, 1, 2, 1)$*.*

One may consider a sequence of exchanges involving first owners 1 *and* 2*, then* 1 *and* 3 *and finally* 2 *and* 3*. Precisely,*

$$\sigma = (\{a, c\}, \{1, 2\}, \{X^1(a) = 2, X^1(c) = 1\}),$$

$$(\{b, e\}, \{1, 3\}, \{X^2(b) = 3, X^2(e) = 1\}),$$

$$(\{d\}, \{2\}, \{X^3(d) = 2\}).$$

Sequence σ *engenders a sequence of ownership maps such that* $b_0 = (1, 1, 2, 3, 3)$*,* $b_1 = (2, 1, 1, 3, 3)$*,* $b_2 = (2, 3, 1, 3, 1)$*,* $b_3 = (2, 3, 1, 2, 1) = b_*$*.*

The sequence σ defines, for each protocol step k, (i) the assets A^k whose property is to be transferred, (ii) the new assets' owners O^k and, (iii) the function X^k assigning the precise owner to each asset. The protocol is decentralized as ownership transfers take place among the owners themselves. At each step k an asset can change owner, the final configuration at time t provides the final asset owner.

In the simple case where agents agree on exchanging assets through a single ownership transfer at each step k, the corresponding *single-swap protocol* is represented by the sequence $\sigma = \{(a^k, o^k) : o^k \in \mathcal{O}, a^k \in \mathcal{A}\}_{k\in\{1,\ldots,t\}}, t \in \mathbb{N} : t \leq m$.

Protocol σ describes a general asset exchange agreement, and not necessarily a swap where the transfer of an asset ownership exists only if associated with the transfer of another one. More precisely, swap participants are interested in the final (at time t) and not temporary (intermediate at step $k \neq t$) acquisition of one or more assets owned by other agents. In a swap protocol, each asset changes owner only once.

Definition 4.3 (decentralized swap protocol) *A decentralized swap protocol is defined as a decentralized exchange protocol where the set $\{A^k : k = 1, \ldots, t, t \in \mathbb{N} : t \leq m\}$ is a partition of the asset set $\mathcal{A}$.*

Example 4.2.1 provides a decentralized swap protocol σ since $\{\{a, c\}, \{b, e\}, \{d\}\}$ is a partition of $\mathcal{A}$. From the formalization introduced in Definition 4.2, it is possible to derive a graphic representation, by means of a digraph $\mathcal{D} = (V, E)$, as the one proposed by *Herlihy* in [6]. More precisely, vertexes are asset owners $V = \mathcal{O}$ and edges $E \ni e = (e_i, e_o) : (e_i, e_o) \in V^2 \wedge e_i \neq e_o$ can be derived by the original ownership map b_0 (providing e_i) and the desired one b_* (providing e_o). Fig. 4.1 presents the digraph of Example 4.2.1.

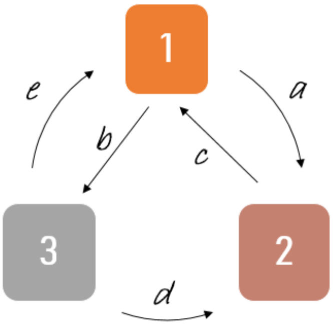


FIGURE 4.1 – Digraph swap protocol representation of the sequence $\sigma = (\{a, c\}, \{1, 2\}, \{X^1(a) = 2, X^1(c) = 1\}), (\{b, e\}, \{1, 3\}, \{X^2(b) = 3, X^2(e) = 1\}), (\{d\}, \{2\}, \{X^3(d) = 2\})$.

4.2.1 Atomic Swap Protocol

Whenever swapping parties do not trust each other it is in their interest to ensure that no participant can take advantage from the swap agreed on. The protocol must be constructed in such a way that the swap is performed in its entirety or no asset transfer is committed (i.e., all-or-nothing). In the case of failures during the protocol execution, every swap participant must be able to regain possession of the original owned assets.

Definition 4.4 (efficient decentralized swap protocol) *A decentralized swap protocol σ is said to be efficient if the engendered sequence $b_1^\sigma, b_2^\sigma, \ldots, b_t^\sigma$ is such that $b_t^\sigma = b_*$.*

Definition 4.5 (atomic decentralized swap protocol) *A decentralized swap protocol σ is atomic whenever it is efficient (in the sense of Definition 4.4) or $b_t^\sigma = b_0$.*

In order to prevent participants to externally exchange the assets involved in the swap, the protocol requires assets to be *locked* in specific transactions (i.e., they cannot be the object of other transfers). Once locked, the transfer commitment allows every participant to redeem the new swapped asset(s). Moreover, due to the atomicity requirement :

(i) any commitment should be conditioned on the correct asset locking i.e., transfers are committed only when all assets of the swap are correctly locked ;

(ii) consequently to failures in the assets locking, the initial situation must be restored ;

(iii) once an asset transfer is committed all the other transfers have to be committed, too.

When considering blockchains – decentralized trustless environments – swap protocols need to be *atomic* according to Definition 4.5. Conditional transfers in blockchain systems are implemented with *distributed contracts* i.e., scripts executed on blockchain nodes that can enforce and regulate relationships among network actors. A widely adopted commitment scheme is the crypto-primitive *hash-lock* [289] i.e., asset ownership is locked with a hash value h that is the outcome of a one-way function H with a secret s as input (i.e., $h = H(s)$) and can be unlocked only when s is revealed. Transfers are conditioned on the hash constraint guaranteeing that assets to be swapped have been properly locked. Whenever transfers are not committed, due to atomicity, the initial ownership configuration needs to be restored ; this can be implemented with *time-locks* [290] i.e., a contract primitive restricting the asset transfer until a specified future time. Hash-locks and time-locks properly combined enable conditional asset transfers necessary to satisfy requirements (i) and (ii) of a swap protocol. Hence, when considering a blockchain swap, an asset transfer consists of an atomic distributed contract that, according to certain protocols [5; 6], coincides with a *distributed hash-time locked contract* (HTLC).

The latter needs to be correctly implemented, published and validated on the network (i.e., part of the valid transaction history) before being committed. Concerning requirement (iii), a mechanism that forces committing all transfers is needed in order to avoid possible atomicity violations (see Section 4.4.1.3). However, having a forcing scheme comes at a price, i.e., loss of decentralization (see Section 4.4.2).

4.2.2 Atomic Blockchain Swap Protocol

In this work we consider blockchain swap protocols characterized by two distinct phases : a first phase in which transfers (i.e., hash-locked contracts) are published and a second one where they are committed. To achieve atomicity, the order in which operations in the two phases are executed matters (see Section 4.4). The commitment phase has to be conditioned by the execution of the publishing phase ; all the asset transfers have to be published before being committed. Therefore, a blockchain swap protocol is characterized by a publishing protocol σ_P followed by a commitment protocol σ_T. While the first protocol (σ_P) is a simple sequence of transfers, the second one (σ_T) is a decentralized swap protocol according to Definition 4.3.

In order to analyze the participants' strategic behaviors the protocol needs to specify the schedules in which agents perform operations. Therefore, given a sequence of operation σ, a set of asset owners $\mathcal{O}$ and a centralized trusted authority τ, we define a *decision function* $F : \{1, \ldots, t\} \to \mathcal{O} \cup \{\tau\}$ as a map that specifies the agent(s) $F(k)$ responsible for the publication (or commitment) of the transfer (A^k, O^k). Note that the agent(s) called to decide on the transfer can be either asset owners (i.e., $F(k) \in \mathcal{O}$) or an external trusted actor (i.e., $F(k) = \tau$).

Definition 4.6 (decentralized blockchain swap protocol) *A decentralized blockchain swap protocol or simply a blockchain swap protocol is defined by the pair* (σ_P, σ_T) *where*

- $\sigma_P = \{(A^j, O^j)\}_{j \in \{1,\ldots,t_P\}}$, $t_P \in \mathbb{N} : t_P \leq m$, $A^j \subseteq \mathcal{A}$ *and* $O^j \subseteq \mathcal{O}$ *is a sequence such that* $\forall j \in \{1, \ldots, t_P\}$, $O^j = \{o \in \mathcal{O} : o \in b_*(A^j) \vee o \in b_0(A^j)\}$ *and,*
- $\sigma_T = \{(A^k, O^k, X^k)\}_{k \in \{1,\ldots,t_T\}}$ *is a swap protocol engendering the sequence of maps* $b_1^{\sigma_T}, \ldots, b_{t_T}^{\sigma_T}$:

$\mathcal{A} \to \mathcal{O}$ according to Definition 4.3.

We associate to each sequence σ_P, σ_T the corresponding decision function F_P, F_T defined above.

Let us note that the publishing sequence σ_P is constructed in such a way that transfers that can be published are of type $(A^j, b_*(A^j))$ or $(A^j, b_0(A^j))$. That is, blockchain transactions can transfer the asset ownership to desired owners and original owners only.

Definition 4.7 (atomic blockchain swap protocol) *An atomic blockchain swap protocol consists of a pair (σ_P, σ_T) where the engendered ownership map at time t of the commitment protocol coincides with the desired one ; $b^{\sigma_T}_{t_T} = b_*$ or with initial one ; $b^{\sigma_T}_{t_T} = b_0$.*

4.2.2.1 Phases Separation

We have defined a blockchain swap protocol as a publishing sequence followed by a commitment emphasizing the precedence of the first phase over the second. However, it is necessary to condition the execution of the commitment protocol to the publication of all the contracts in order to have an atomic blockchain swap protocol. The following definition formalizes such a *commitment requirement.*

Definition 4.8 (commitment requirement) *Given a blockchain swap protocol (σ_P, σ_T) whenever there exists an asset transfer (using the swap structure previously given) that is not correctly published, then no asset transfer is committed. Formally, if, in σ_P, $\exists \bar{j} \in \{1, \ldots, t_P\} : O^{\bar{j}} \cap b_0(A^{\bar{j}}) \neq \emptyset$ then, in σ_T, $b^{\sigma_T}_k = b_0 \; \forall k \in \{1, \ldots, t_T\}$.*

Every blockchain swap protocol has to meet the commitment requirement defined above in order to be atomic. Hence, Definition 4.8 is a necessary condition (but not sufficient, see Section 4.4.1.3) for atomicity. Indeed, whenever an asset transfer is not correctly published the time-lock acts by not modifying the original asset ownership, i.e., transferring the asset back to the original owner (see [291; 292] for more details).

Focusing separately on the commitment protocol, it is possible to derive initial properties on blockchain swaps.

4.2.2.2 Commitment Protocol

As stated in the following proposition, whenever an asset transfer is not committed, the swap participant supposed to acquire the asset(s) ends up with less assets than expected while the original asset owner finds himself with an extra asset. Next proposition does not depend on the exact specification maps X^k, so we omit them in the sequence σ_T for the sake of simplicity.

Proposition 4.1 *Given a commitment sequence* $\sigma_T = \{(A^k, O^k, X^k)\}_{k\in\{1,\dots,t_T\}}, t_T \in \mathbb{N} : t_T \leq m$ *then, replacing* O^k *by* $b^{\sigma_T}_{k-1}(A^k)$ *in* σ_T *i.e., considering a new sequence* $\sigma^k_T = (A^1, O^1), \dots, (A^{k-1}, O^{k-1}), (A^k, b^{\sigma_T}_{k-1}(A^k)), (A^{k+1}, O^{k+1}), \dots, (A^{t_T}, O^{t_T})$*, for some* $k \in \{1, \dots, t_T\}$*, implies that :*

(i) $(b^{\sigma^k_T}_{t_T})^{-1}(O^k) \subseteq (b^{\sigma_T}_{t_T})^{-1}(O^k)$ *and,*

(ii) $(b^{\sigma^k_T}_{t_T})^{-1}(b^{\sigma_T}_{k-1}(A^k)) \supseteq (b^{\sigma_T}_{t_T})^{-1}(b^{\sigma_T}_{k-1}(A^k))$.

Proof. *The claims follow from the fact that* A^k *is not given to* O^k *in* σ^k_T *but it is given back to the original owners.* ■

At time t, assets A^k do not necessarily belong to O^k thus, O^k could find themselves with no asset i.e., $(b^{\sigma^k_T}_t)^{-1}(O^k) = \emptyset$. Moreover, the original asset owners find themselves with both the original assets and the swapped ones. Let us denote by σ^K_T, with $K \subseteq \{1, \dots, t_T\}$ the sequence of pairs obtained by replacing (A^k, O^k) in σ_T by $(A^k, b^{\sigma_T}_{k-1}(A^k))$ for any $k \in K$ (i.e. by the owner(s) of A^k at step $k-1$).

4.2.2.3 Sequential Phases and Beyond

The formalization provided by Definition 5.23, enables capturing :

1- *Single-asset and multi-assets swaps* depending on the cardinality of the asset set A^k at step k in the commitment protocol.

2- Swap protocols with *sequential publishing and commitment*, both single-asset and multiple-assets, where ownership transfers are published and committed according to a precise temporal order. Every crypto-asset transfer has to be executed before or after another one. In Section 4.4.1 we show that the sequentiality of the publishing phase combined with the leader role of a swap participant guarantees that the commitment ownership is verified.

3- Swap protocols with *concurrent publishing* where transfers are no longer published according to a given time order, but may be concurrently created and propagated to the blockchain network.

In fact, it is not essential that the various transfers are published one at a time but only that there is a clear distinction between the publication and the commitment phase. Therefore, the transfers publication can take place in a concurrential way and this is captured by the sequence σ_P where at step $j : j \in \{1, \ldots, t_P\}$ multiple assets transfers (A^j, O^j) are published.

4- Swap protocols with *snap commitment*, where assets transfers are all committed at the same time.

Proposition 4.2 *A blockchain swap protocol with a snap commitment scheme satisfying the commitment requirement as in Definition 4.8 is atomic.*

Proof. *By contradiction, if $b^{\sigma_T}_{t_T} \neq b_0 \wedge b^{\sigma_T}_{t_T} \neq b_*$ then, since no problem in the publishing occurs, there should be a situation where some assets transfers are triggered and some others are not. However, this contradicts the snap commitment.* ■

4.2.2.4 Decision Function

Let us focus on the decision function F previously defined. A blockchain swap can be completely driven by the asset owners only, in that case the function outcomes are all elements of $\mathcal{O}$. On the other hand, we can have the intervention of an external actor τ entrusted for committing assets transfers. More precisely, the external trusted actor cannot publish blockchain contracts in behalf of asset owners since blockchain swap protocols contemplate ownership transfers among owners only (as in Definiton 5.23). Considering the publishing phase we can notice that every asset owner is in charge of publishing the signed contract transferring the asset ownership. In case of multi-assets transfers a *multisig* scheme [293] can be adopted.

Definition 4.9 (ownership requirement) *In a blockchain swap protocol, the decision function F_P (corresponding to sequence σ_P) is such that $F_P(j) = b_0(A^j) \subseteq \mathcal{O}$ for any $j \in \{1, \ldots, t_P\}, t_P \in \mathbb{N} : t_P \leq m$.*

Focusing on the commitment phase we can derive the following definition stating that whenever the decision function is effective, agents in O^k have the power to accept or decline (i.e., redeem or not) the acquisition of the asset(s) in A^k.

Definition 4.10 *A decision function F_T is effective on σ_T if and only if $F_T(k) = O^k$ for any $k \in \{1, \ldots, t_T\}, t_T \in \mathbb{N} : t_T \leq m$.*

4.3 Game Theoretical Analysis of Cross-Chain Swaps

Thanks to the formalization presented in the previous section we can use game theory to analyze strategic behaviors of the swap participants (modeled as rational agents) in a blockchain swap protocol. In this sections we associate a specific blockchain swap protocol (σ_P, σ_T) (e.g., sequential, concurrent publishing and snap commitment) with the corresponding game (in strategic or extensive form). We prove that (i) following a swap protocol characterized by an *effective* decision function is a subgame perfect equilibrium in dominant strategies and, that (ii) following a swap protocol characterized by concurrent publishing and *snap* (immediate) commitment is a Nash equilibrium.

Considering decentralized swap protocols, the participants, intervening in both commitment and publishing phase, are more or less incentivized to stick or not to the protocol regulating the swap. The outcomes of a single agent, varying according to personal strategies, result depending from other agents strategic behaviors, too. What is common knowledge among the swap participants is the swap protocol structure that is represented by the pair (σ_P, σ_T).

As discussed in Section 4.2.2, blockchain swap protocols in both single and multi-assets case, can be characterized by sequential publishing and commitment, concurrent publishing and snap commitment. For those protocols characterized by concurrent moves (i.e., concurrent publishing) we can adopt a representation with *games in normal form* presented in Section 2.3.1 of Chapter 2.

Definition 4.11 *Let $\langle \mathcal{A}, \mathcal{O}, b_0, b_*, (u_i)_{i\in\mathcal{O}} \rangle$ be a swap problem, let (σ_P, σ_T) be a blockchain swap protocol and let $F_P : \{1, \ldots, t_P\} \to \mathcal{O}$ and $F_T : \{1, \ldots, t_T\} \to \mathcal{O} \cup \{\tau\}$ be decision functions. We associate the game in normal form $\langle N, (S_i)_{i\in N}, (u_i)_{i\in N} \rangle$ such that :*

- *$N = \mathcal{O}$ is the set of players ;*
- *$(S_i)_{i\in N} = \{action\ 1, action\ 0\}$ is the set of pure strategies for player i consisting in the pair (*follow *the protocol,* not follow *the protocol) labelled respectively as action 1 and action 0 ;*
- *$(u_i)_{i\in N} : u_i : \prod_{j\in N} S_i \to \mathbb{R}$ is the payoff function for asset owners $N = \mathcal{O}$ evaluating the outcomes of type $b_{t_T}^{\sigma_T}$ that is, for every player i we have $u_i\big((b_{t_T}^{\sigma_T})^{-1}(i)\big)$.*

We model protocols with sequential phases with *extensive form games* (see Section 2.3.1) in order to represent the sequencing of swap participants' moves and the fact that at each decision point asset owners know the moves history so far. Indeed, that blockchains involved in a swap are of public nature with open read access i.e., swap participants have a complete vision of other agents choices.

Definition 4.12 *Let $\langle \mathcal{A}, \mathcal{O}, b_0, b_*, (u_i)_{i\in\mathcal{O}}\rangle$ be a swap problem, let (σ_P, σ_T) be a blockchain swap protocol and a let $F_P : \{1, \dots, t_P\} \to \mathcal{O}$, $F_T : \{1, \dots, t_T\} \to \mathcal{O} \cup \{\tau\}$ be decision functions. We associate the extensive game form $\Gamma^\sigma = \langle N, T, P, (A_h)_{h\in V}, (u_i)_{i\in N}\rangle$ such that :*

- *$N = \mathcal{O}$ is the set of players ;*
- *T is a (binary) directed tree such that each directed path from the root $v_0 \in V$ to an end node $v \in Z$ is formed by precisely $t + 1$ nodes and t arcs (V is the set of the nodes of the tree T and $Z \subset V$ is the set of leaf nodes) ;*
- *$P(v) = F_{P,T}(l(v))$, for each $v \in V \setminus Z$, is the publisher/activator of the asset transfer(s) at step $l(v)$ in the protocol (σ_P, σ_T), where $l(v)$ is the number of arcs between v_0 and v on the unique path from v_0 to v (we assume that $l(v_0) = 0$) ;*
- *A_h for all $h \in V$ is formed by two outgoing arcs in h ; one arc in A_h is labeled with* action 1 *(i.e.,* follow *the protocol) and the other one,* action 0 *(i.e.,* not follow *the protocol) for any $h \in V \setminus Z$. So, a unique path from v_0 to an end node $z \in Z$ identifies a binary vector $p^z \in \{0, 1\}^t$ such that p^z_k is the label of the arc starting from node v with $l(v) = k$ on the path from v_0 to z.*
- *Any end node $z \in Z$ is associated with a unique outcome corresponding to the map $b^{\sigma^K_T}_{t_T}$ where $K \subseteq \{1, \dots, t_T\}$ is such that $p^z_k = 0$ for any $k \in K$, and $p^z_k = 1$ for any $\{1, \dots, t_T\} \setminus K$. So, for any $i \in \mathcal{O}$, the outcome $b^{\sigma^K_T}_t$ is evaluated by i with the payoff function $u_i\big((b^{\sigma^K_T}_t)^{-1}(i)\big)$.*

Proposition 4.3 *Let $\Gamma^\sigma = \langle N, T, P, (A_h)_{h\in V}, (u_i)_{i\in N}\rangle$ be the extensive form game associated with the swap problem $\langle \mathcal{A}, \mathcal{O}, b_0, b_*, (u_i)_{i\in\mathcal{O}}\rangle$ and the blockchain swap protocol with sequential phases (σ_P, σ_T) and let $F_T : \{1, \dots, t_T\} \to \mathcal{O} \cup \{\tau\}$ be a decision function. If F_T is effective on σ_T, then the strategy profile $(\hat{s}_1, \dots, \hat{s}_n)$ that specifies action 1 (follow the protocol) at any node is the unique subgame perfect equilibrium (in dominant strategies).*

Proof. *For each node v, by the fact that F_T is effective, we have that $P(v) = F_T(l(v)) = O^{l(v)}$. So, at each decision node $v \in V \setminus Z$, if player $P(v)$ specifies action 0 (not follow the protocol) at node v, then by the first claim of Proposition 4.1, player $P(v)$ ends up with a set of assets that is contained in the one that player $P(v)$ would obtain if she/he specifies action 1 at node v. So, the utility of player $P(v)$ is larger if it chooses, at each decision node, action 1 than action 0. It follows that at each node v action 1 strictly dominates action 0 for player $P(v)$.* ■

Note that if (σ_P, σ_T) is efficient, then in Proposition 4.3 the unique subgame perfect equilibrium in dominant strategies corresponds to the desired outcome b_*. Proposition 4.3 shows that, whenever players in the game have to decide whether to accept or decline an asset acquisition, they are incentivized to follow the protocol by accepting the desired asset. Nonetheless, a stronger result can be proved.

Proposition 4.4 *Let $\Gamma^\sigma = \langle N, T, P, (A_h)_{h\in V}, (u_i)_{i\in N}\rangle$ be the extensive form game associated with the swap problem $\langle \mathcal{A}, \mathcal{O}, b_0, b_*, (u_i)_{i\in\mathcal{O}}\rangle$ and the blockchain swap protocol with sequential phases (σ_P, σ_T) and let $F_T : \{1, \dots, t_T\} \to \mathcal{O} \cup \{\tau\}$ be a decision function. The decision function F_T is effective on the blockchain swap protocol (σ_P, σ_T), if and only if the strategy profile $(\hat{s}_1, \dots, \hat{s}_n)$ that specifies action 1 (follow the protocol) at any node is the unique subgame perfect equilibrium (in dominant strategies).*

Proof. *We have to prove the "if" since the "only if" directly follows from Proposition 4.3. By contradiction, if F_T is not effective we have the following cases : (i) $P(v) = F_T(l(v)) = b_0^{\sigma_T}(A^{l(v)})$, the original owner of the assets decides whether or not to follow the protocol or, (ii) $P(v) = F_T(l(v)) = O^k \neq b_0^{\sigma_T}(A^{l(v)})$ such that $k \in \{1, \dots, t_T\} : k \neq l(v)$, the activators are any player in the game but the original asset owners and the asset receivers.*

(i) Whenever the owners of the assets in the initial configuration have to decide between action 0 (not follow the protocol) and action 1 (follow the protocol), by the second claim of Proposition 4.1 they have no incentive to follow the protocol σ_T. That is, not following the protocol the asset set obtained at time t_T is greater then the one obtained by following the protocol.

(ii) Whenever the activators of transfers (A^k, O^k) are neither O^k nor $b_0^{\sigma_T}(A^k)$, the situation is more complex. Assuming that players activate only exchanges of other players, it always exist an activator $o^l(v) : l(v) \in \{1, \dots, t\}$ that has to decide whether to follow the protocol σ_T or to deviate by originating the sequence σ_T^K (defined in Section 4.2.2). However, the activator would be indifferent to the two strategies since $u_{o^{l(v)}}((b_t^{\sigma_T})^{-1}(o^{l(v)})) = u_{o^{l(v)}}((b_t^{\sigma_T^K})^{-1}(o^{l(v)}))$ due to the fact that the cardinality of the asset set remains unchanged.

Therefore, the strategy profile specifying action 1 (follow the protocol) at each stage cannot be a perfect equilibrium in dominant strategies if F_T is not effective. ■

This result seems to rule out any swap protocol with a non-effective decision function $F_T : \{1, \dots, t_T\} \to \mathcal{O}$. However, we can imagine protocols with transaction triggering mechanism not involving receiving agents (see Section 4.4.2).

4.4 Cross-Chain Swap protocols

This section is devoted to the analysis of existing cross-blockchains swap protocols aiming to move forward the custodial trading performed by centralized exchanged services. Our generic framework allows us to characterize equilibria of two representative recent protocols presented in [5] and [7] respectively. In the case of the protocol proposed in [5] and generalised in [6], following the protocol is the unique subgame perfect equilibrium (in dominant strategies), while in the case of the protocol proposed in [7], following the protocol is a Nash equilibrium.

4.4.1 Sequential Publishing and Commitment

Here we present the swap solution proposed by *Nolan* [5] for permissionless UTXO-based blockchains. Nolan's protocol make use of contracts [294], hash-locks as commitment/locking scheme and time-locks to restore the initial situation consequently to failures in the publishing phase.

Given two asset owners (e.g., Alice and Bob) aiming at cross-swapping two crypto-assets (e.g., x Bitcoins and y Litecoins), the protocol (represented in Fig. 4.2) works as follows :

1. The agent Alice creates a secret s such that $h = H(s)$, and publishes a contract transferring the ownership of her x Bitcoins to Bob on the Bitcoin blockchain. The contract is locked with the hashlock h and a timelock Δ_{Bob} ensuring that : "Bob can claim the asset property providing s before time Δ_{Bob}".
2. When Bob confirms that Alice's contract has been correctly published on the Bitcoin blockchain, he publishes a contract on the Litecoin blockchain with the same hashlock h but with timelock Δ_{Alice} stipulating that : "before time Δ_{Alice}, Alice can claim the asset property with secret s".

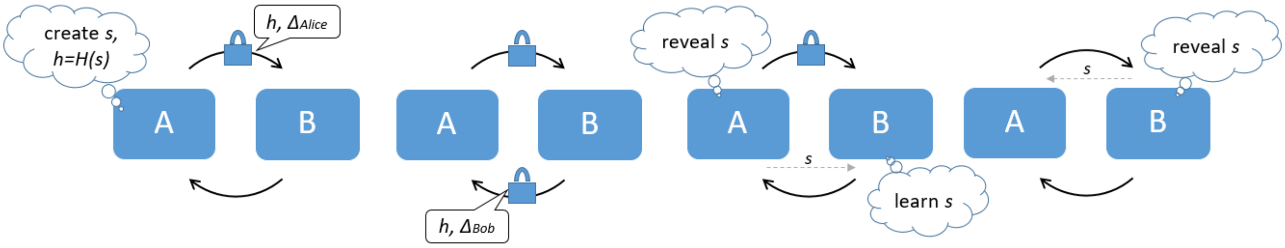


FIGURE 4.2 – Two party 'atomic' cross-chain swap protocol proposed in [5; 6] characterized by sequential publishing and commitment : $\sigma_P = \{(x, B), (y, A)\}$ and $\sigma_T = \{(y, A), (x, B)\}$.

Note that $\Delta_{Alice} < \Delta_{Bob}$

3. When Alice confirms that Bob's contract has been correctly published on the Litecoin blockchain, she sends s to Bob's contract (before time Δ_{Alice}), acquiring the y Litecoins and revealing s to Bob.
4. Bob then sends s (in the time interval $[\Delta_{Alice}, \Delta_{Bob}]$) to Alice's contract, acquiring the x Bitcoins and completing the swap.

According to our formalization the two party cross-chain swap is represented as follows :

$$\sigma_P = \{(x, B), (y, A)\}, \quad F_P(j) = \{A, B\}, \quad j = \{1, 2\};$$
$$\sigma_T = \{(y, A), (x, B)\} \quad F_T(k) = \{A, B\}, \quad k = \{1, 2\}.$$

The protocol assumes that swap participants (i) actively monitor the involved blockchain in order to confirm contracts publication and, (ii) adopt a common hashing method. *Herlihy* [6] extends the protocol for multi-assets and multi-agents swaps (with the secret creators forming a *feedback vertex set* L i.e., a subset of V whose deletion leaves $\mathcal{D} = (V, E)$ acyclic) and analyzes under which conditions it is 'atomic'. *Atomicity*, in this case, is defined in a game theoretical fashion : in [6] a swap protocol is defined as atomic if :

- "following the protocol" is a *nash equilibrium strategy* and,
- no conforming party is affected (in terms of payoff) by a protocol deviation.

Moreover, multi-asset swaps that are atomic according to [6] have a *strongly connected* corresponding digraph $\mathcal{D}$ (i.e., for every pair of distinct owners there is always a path from one to the other and viceversa).

4.4.1.1 The Separating Agent

In multi-players protocol implementations, the role of the secret creator or *leader*, denoted as l, becomes crucial. Let us consider only single leader swap protocols. The latter initiates the publishing and most importantly the commitment phase of the protocol by disseminating the secret s. A swap leader reveals the secret whenever all the contracts transferring her the ownership of the desired crypto-assets are correctly published. The sequentiality of the publishing together with the leader role ensures the required separation between the two phases of the blockchain swap. Let us formalize this concept in the following proposition :

Proposition 4.5 *A blockchain swap protocol verifying the atomicity definition proposed in [6] and characterized by :*

(i) a leader participant initiating the publication phase $F_P(1) = \{l\}$ together with the commitment one $F_T(1) = \{l\}$ when all the contracts where she is directly involved are published,

(ii) a sequential publishing phase where asset owners are called to publish as soon as all the contracts transferring them the desired assets' property are published,

satisfies the commitment requirement of Definition 4.8.

Proof. *We can state that the last contract(s) to be published are the ones involving the leader acting as a receiver; $O^{t_P} = \{l\}$. If a contract is not correctly published ($\exists \bar{j} \in \{1, \ldots, t_P\} : O^{\bar{j}} \cap b_0(A^{\bar{j}}) \neq \emptyset$), either the receiver is the leader, hence the commitment cannot start for (i), or the receiver is a different participant which for (ii) cannot publish her contracts. Indeed, since the graph is strongly connected, there is a sequence of participants, from the receiver to the leader, not activating their contracts.* ■

4.4.1.2 Protocol Strategic Behavior

The blockchain swap protocol presented above works with a sequential commitment phase where swap participants trigger contracts transferring them the ownership of the desired assets. Therefore, the decision function characterizing the protocol is effective. Note that, thanks to Proposition 4.4, we can derive a stronger result than the one proposed by the original paper concerning the players strategic behavior.

Corollary 4.0.1 *In the blockchain swap protocol (σ_P, σ_T) presented in [5] (and generalised in [6]), the strategy profile $(\hat{s}_1, \ldots, \hat{s}_n)$ specifying action 1 (follow the protocol) at any node is the unique subgame perfect equilibrium (in dominant strategies).*

Let us provide (Fig. 4.3) the graphical representation of the extensive form game associated to the blockchain swap protocol (σ_P, σ_T) presented in [5]. We do consider two owners $\mathcal{O} = \{A, B\}$ swapping

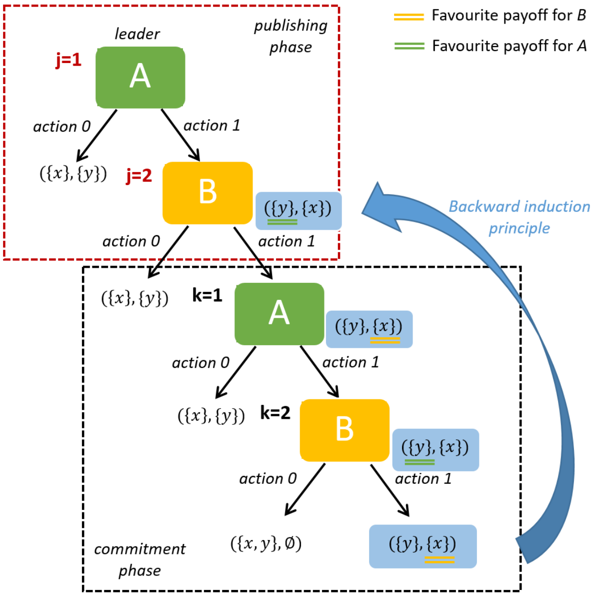


FIGURE 4.3 – Extensive form game associated to the blockchains swap protocol, presented in [5]. Publishing and commitment phases are sequentially executed one after the other. Outcomes represent the payoffs (assets) owned by Alice and Bob respectively. The figure represents the backward induction process for each decision step.

two assets $\mathcal{A} = \{x, y\}$ where $t_P = t_T = 2$. Alice, as leader of the swap protocol, is the first player to decide between *action 0* and *action 1*. Whenever one of the two parties opts for action 0 during the publishing phase, due to the commitment requirement (i.e., Definition 4.8), the games terminates with the original ownership configuration as the outcome : $b_{1,2}^{\sigma_P} = b_0 = (A, B)$ for $\mathcal{A} = \{x, y\}$. If Alice does not start the commitment phase the outcome $b_1^{\sigma_T}$ coincides with $b_0 = (A, B)$. When Bob is called to decide, Alice's previous moves are known ; if Bob decide to follow the protocol the swap takes place, $b_2^{\sigma_T} = b_* = (B, A)$, otherwise Alice acquires both the desired asset and the originally owned one leaving Bob empty-handed, $b_2^{\sigma_T} = (A, A)$. Outcomes in Fig. 4.3 represent the payoffs of Alice and Bob respectively.

Subgame perfect equilibria are computed by applying the *backward induction* process presented in Section 2.3.2. By reasoning from the end to the beginning of the game, at each decision step the strategy providing a better payoff (i.e., the one providing greater utility) is selected. Then, considering the game associated to the swap protocol presented in [5], the sequence of optimal actions is the one specifying action 1 at each decision step (see Fig. 4.3).

4.4.1.3 Atomicity Violations

In [7] the authors observe that the protocol presented by *Nolan* [5] is not immune to violation of the all-or-nothing atomicity as in Definition 4.7. More precisely, a time-lock expiration before commitment can lead an honest swap participant to deviate from the protocol. As in Fig. 4.3 if Bob does not commit the transfer at the last decision step (i.e., if the time-lock expires before) the outcome is $b_2^{\sigma_T} = (A, A)$ which differs from both b_0 and b_*. Crash failures together with network delays are some of the possible causes of time-lock expiration before commitment making the deviating party ending up with an asset loss. Rational swap participants may be induced to deviate from the prescribed swap protocol (i.e., behaving as Byzantine agents) by external factors. Hence, the need to analyze the robustness of cross-chain swap protocols (those that are vulnerable to atomicity violations e.g., [5]) to Byzantine behaviors (see Section 5.4 of Chapter 5). A snap commitment represents a solution to atomicity violations in asynchronous environments.

4.4.2 Concurrent Publishing and Snap Commitment Protocols

In [7] authors propose an atomic swap protocol characterized by a concurrent publishing phase and a snap commitment. The protocol AC3TW works with a centralized trusted authority τ called Trent, acting as a separating agent that (i) verifies the correctness of the publishing phase and, (ii) witness the redemption contracts. Thanks to the trusted witness Trent the protocol benefits from all-or-nothing atomicity and faster publishing phase with respect to the sequential one (i.e., increasing overhead proportional to the number of contracts involved in the swap).

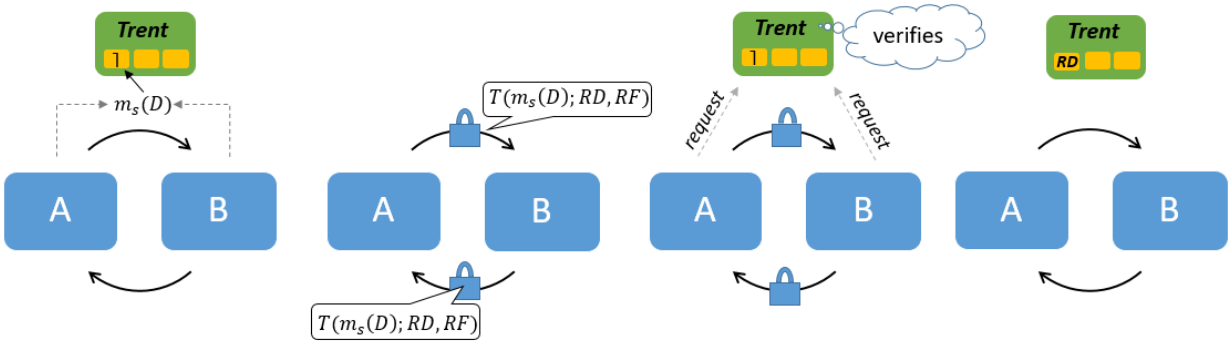


FIGURE 4.4 – Two party AC3TW cross-chain swap protocol proposed in [7] characterized by concurrent publishing and snap commitment. It is represented by the following formalization : $\sigma_P = \{(\{x, y\}, \{A, B\})\}$ and $\sigma_T = \{(\{x, y\}, \{A, B\}, \{X^1(x) = B, X^1(y) = A\})\}$.

The protocol constructs for every possible swap configuration a directed graph $\mathcal{D} = (V, E)$ similar to the one of [6] (see Section 4.2 for more details). $\mathcal{D}$ is multisigned by all swap participants in the set V generating a graph multisignature $m_s(\mathcal{D})$. The signatures order is irrelevant, the multisignature represents the participants' agreement on $\mathcal{D}$.

Given two asset owners (e.g., Alice and Bob) aiming at swapping x Bitcoins for y Ethereum here below the steps characterizing the two party AC3TW protocol (represented in Fig. 4.4) :

(1) Alice and Bob create the digraph $\mathcal{D}$ and multisign it generating $m_s(\mathcal{D})$.

(2) The multisignature is registered and stored by the centralized trusted authority, Trent, only if not registered before and it is set to a null value $\rceil$.

(3) Alice publishes the contract C_1 on the Bitcoin blockchain stating that :

   - if Bob provides Trent's signature to the redemption instance, i.e., if he provides $T(m_s(\mathcal{D}); RD)$ then, x Bitcoins' ownership is transferred from Alice to Bob.
   - if Alice provides $T(m_s(\mathcal{D}); RF)$ then, the x Bitcoins are transferred back to Alice.

(4) Concurrently, Bob publishes a contract C_2 on the Ethereum network stating the following :

   - if Alice provides $T(m_s(\mathcal{D}); RD)$ then, y Ethereums' ownership is transferred from Bob to Alice.
   - if Bob provides $T(m_s(\mathcal{D}); RF)$ then, the y Ethereums are transferred back to Bob.

(5) After the publication of $C_i : i = 1, 2$ either Alice or Bob requests Trent to trigger a redemption commitment scheme to redeem the assets. Trent issues $T(m_s(\mathcal{D}); RD)$ only if both C_1 and C_2 are correctly published in their corresponding blockchains and the value of $m_s(\mathcal{D})$ stored by Trent is $\rceil$.

(6) Whenever a contract is not correctly published any participant can request Trent to trigger a refund commitment scheme. $T(m_s(\mathcal{D}); RF)$ is issued only if $m_s(\mathcal{D})$ has value $\rceil$.

(7) Depending on the case, Trent sets the value of $m_s(\mathcal{D})$ to $T(m_s(\mathcal{D}); RD)$ or $T(m_s(\mathcal{D}); RF)$ accordingly.

According to our formalization the two party cross-chain AC3TW protocol is represented as follows :

$$\sigma_P = \{(\{x, y\}, \{A, B\})\}, \quad F_P(j) = \{A, B\}, \quad j = \{1\};$$

$$\sigma_T = \{(\{x, y\}, \{A, B\}, \{X^1(x) = B, X^1(y) = A\})\}$$

$$F_T(k) = \tau, k = \{1\}.$$

All-or-nothing atomicity is achieved by the fact that the redemption $T(m_s(\mathcal{D}); RD)$ and the refund $T(m_s(\mathcal{D}); RF)$ events are mutually exclusive. The protocol meets the commitment requirement due to Trent's witnessing activity. The latter reduces as well the *interactivity* (i.e., the active participation of the swap participants) of the swap protocol [286] with respect to Nolan's swap implementation where asset owners have to be constantly on-line monitoring the involved blockchains. The two contracts of the AC3TW protocol are respectively a Bitcoin contract C_1 and an Ethereum smart contract C_2 (see [294; 295; 296] for more details) therefore, the protocol works in both UTXO-based and account-based blockchains [14]. Moreover, concerning the protocol strategic behavior we have that "following the protocol" is a Nash equilibrium.

Proposition 4.6 *Let $\langle N, (S_i)_{i \in N}, (u_i)_{i \in N} \rangle$ be the normal form game associated with the swap problem $\langle \mathcal{A}, \mathcal{O}, b_0, b_*, (u_i)_{i \in \mathcal{O}} \rangle$ and the blockchain swap protocol (σ_P, σ_T) characterized by a concurrent publishing and a snap commitment where the decision function F_T is such that $F_T(k) = \tau \ \forall k \in \{1, \ldots, t_T\}$. Then, the strategy profile $(\hat{s}_1, \ldots, \hat{s}_n)$ that specifies action 1 (follow the protocol) for every player i is a Nash equilibrium.*

Proof. *Given a strategy profile $s = (s_i)_{i \in N} \in \prod_{j \in N} S_i$ whenever $\exists i \in N$ such that s_i is action 0 then, due to Trent witnessing, the protocol ends up in the original configuration b_0. The outcome corresponding to the desired configuration b_* is reached whenever action 1 is chosen by all the players. Hence, since $u_i(b_0^{-1}(i)) < u_i(b_*^{-1}(i)) \ \forall i \in N$, action 1 is the dominant best response strategy to all $s_{-i} \ \forall i \in N$.* ■

Let us analyze the normal form game associated to the blockchain swap protocol presented in [7] in the case of two owners $\mathcal{O} = \{A, B\}$ swapping two assets $\mathcal{A} = \{x, y\}$ where $t_P = t_T = 2$. Normal form games represent situations where players make decision simultaneously. In this cases, a matrix representation (Fig. 4.5) allows to quickly analyze each possible outcome. In this case, Alice and Bob have to decide between *action 0* and *action 1* during the publishing phase. Once the transfers are

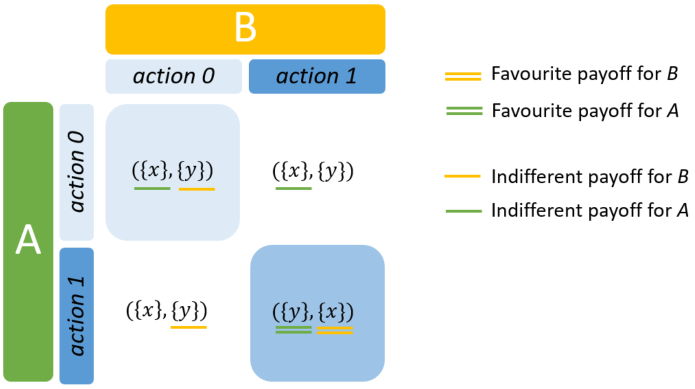


FIGURE 4.5 – Normal form game associated to the blockchains swap protocol, presented in [7]. Alice and Bob decide whether to publish or not the contract on the corresponding blockchain. Outcomes represent the payoffs of Alice and Bob respectively. By eliminating dominated strategies the emphasized outcomes are the equilibria of the game. The dominant strategy, the one providing strictly greater payoffs, is a Nash equilibrium.

correctly published (i.e., action 1 is chosen by both players) Trent commit the swap, $b_1^{\sigma_T} = b_* = (B, A)$ for $\mathcal{A} = \{x, y\}$. On the other hand, if one of the two parties chooses action 0 the outcome is $b_1^{\sigma_P} = b_0 = (A, B)$.

In order to identify the game's equilibria, *dominated strategies* (i.e., strategies providing a lower utility than others) have to be eliminated. Two different equilibria are computed : "following the protocol" is a dominant strategy always providing a greater utility for all the other player's strategies, "do nothing" (i.e., choosing action 0) is a weakly dominant strategy that provides the same payoffs for all the other player's strategies. Since dominant strategies are always Nash equilibria [250], the startegy profile specifying action 1 for every player of the game is a Nash equilibrium.

4.5 Summary

Summary. This contribution formalizes the distributed cross-chain swap problem in the blockchain context where parties exchange assets across multiple blockchains (i.e., inter-operate between different blockchains). To the best of our knowledge this work is the first to propose a complete framework allowing to analyze existing cross-chain swap protocols as normal form games (i.e., non-cooperative games, see Section 2.3.) We prove that (i) following a swap protocol characterized by an *effective* decision function (e.g. the protocol proposed in [5] and generalized in [6]) is a subgame perfect equilibrium in dominant strategies while (ii) following a swap protocol characterized by concurrent publishing and snap commitment (e.g. the protocol proposed in [7]) is a Nash equilibrium. Equilibria represent stable solution where no player has incentive in unilaterally deviating from the prescribed swap protocol. Moreover, the protocol in [5] ensures that no conforming party is affected by a protocol deviation. However, contrary to what stated in [6], these two conditions cannot ensure protocol atomicity as precised in Section 4.4.1.3. Framework in [7] represents an improvement considering a formal adversary model and a *concurrent commitment* guaranteeing protocol atomicity. On the other hand, the next chapter analyzes the robustness of cross-chain swap protocols to Byzantine behaviors (i.e., swap participants who may be induced to deviate or irrationally deviate).

Chapter 5

Blockchain Robustness to Rational and Byzantine Users' Behaviors

Content

In this chapter we propose a game theoretical framework that formally characterizes the robustness of general blockchain systems in terms of resilience to rational deviations and immunity to Byzantine behaviors. Rational blockchain users faced with a blockchain protocol to follow may act altruistically (following the protocol) or maliciously (deviating from the protocol) according to their personal utility. On the other hand, Byzantine users deviate from the protocol in any situation (even acting irrationally). We prove the practical interest of our formal framework by characterizing the robustness of various blockchain protocols (Bitcoin [2], Tendermint [1], Lightning Network [3], a side-chain protocol [4] and a cross-chain swap protocol [5]) to the behaviors of the two main types of blockchain user : *transacting parties* and *validating nodes*. The content of the chapter originates from [18]

5.1 Introduction on Blockchain Robustness

Beyond the traditional blockchain architectures (*layer-1 protocols*), the literature proposes other protocols that respectively define and regulate interactions in an overlaying network (*layer-2 protocols*) and interactions between different blockchains (*cross-chain protocols*). Each of these protocols establishes the instructions that a user must follow in order to interact with or through a blockchain.

In a blockchain system players can be classified, as proposed in [13] for classical distributed systems, in three different categories : (i) players who follow the prescribed protocol i.e., *altruistic*, (ii) those who act in order to maximize their own benefit i.e., *rational*, and (iii) players who may rationally deviate from the prescribed protocol, i.e., *rational Byzantine.* The latest category can be redefined, according to [297], to include any possible arbitrary protocol deviation (including irrational) as *Byzantine.*

Interactions among users are usually modeled with game theory which analyzes the decision-making process in presence of multiple rational agents, called *players* or *agents.*

In the context of blockchain systems, game theoretical frameworks were introduced in [29; 298] to analyze security aspects and incentive compatibility of Nakamoto's consensus protocol (i.e., Proof-of-Work [2]) characterizing the very first blockchain implementation known as Bitcoin. Chapter 3 models the behaviors of users participating to the consensus mechanism (i.e., miners) of the Bitcoin blockchain ; they are considered as individually rational moved by the mere intention to increase their revenues i.e., the rewards earned form the mining activities. Authors in [299; 300; 301] adopt different utility functions for miners and pools that consider costs and relative rewards. Concerning layer-2 and cross-chain protocols, game theoretical analysis are carried out by [302; 17; 6; 303]. These analysis are strictly specific to the particular deployment context than to a generic blockchain. Most of the game theoretical models adopted to design secure and robust blockchain protocols, surveyed in [304], (i) address protocols characterizing specific blockchain implementations, (ii) analyze miners' behaviors in the consensus phase and (iii) adopt Nash equilibria as solution concept.

Concerning *rational agents*, the existing analyzes include the study of the equilibria and the evaluation of their properties. The most studied and adopted solution concept in literature is the Nash equilibrium, i.e., a strategy profile in which no player has interest in individually deviating from her own strategy. A first approach to the analysis of robustness is to compare Nash equilibria, through indices such *Price of Byzantine Anarchy* [305], *Price of Malice* [305] and *Price of Anarchy* [306]. This approach summarizes the outcomes of the games representing protocols, but it does not show explicitly the implementation risks of such systems. A second approach is to analyze peculiar Nash equilibria.

The authors of [307] take probability into account and extend the concept of Nash equilibrium. In [297], *virtual utility* – alternative to the classical game utility – is introduced to capture the blockchain agreement structure.

The analysis of robustness with respect to *Byzantine agents* was modeled in [308] with a Bayesian game. The authors provide the analysis of Tendermint protocol [1]. This method allows making forecasts on the expected outcomes of a game, but it does not provide a comprehensive analysis of the risks. It should be noted that none of the previous works is generic enough to propose a methodology for analyzing the robustness of blockchain protocols to both rational and Byzantine players.

The first generic framework for analyzing the *robustness of distributed protocols* with respect to the behavior of rational and Byzantine players was proposed by the authors of [309] who introduced the concept of *mechanism* (i.e., a pair game-prescribed strategy). Moreover in [309] authors introduced the notions of (i) k-resilience, (ii) practicality and (iii) t-immunity. A strategy profile is defined as *k-resilient* if there is no coalition with at most k players having an incentive to deviate from the prescribed protocol. The category of *practical* strategy profiles is defined when equilibria with weakly dominated strategies are excluded.

In this chapter we follow the line of work opened in [309] and present a game theoretical framework aiming at characterizing the *robustness of blockchain protocols* to rational and byzantine users' behaviors. More precisely, this work focuses on the analysis of blockchain agents who can act rationally (altruistically or maliciously) or maliciously no matter what. Our contributions can be summarized as follows :

a) We prove that t-immunity property defined in [309] is not verified by a large class of blockchain protocols (see Table 5.2). It should be noted that the authors of [309] already observed that "t-immunity is often impossible to be satisfied by practical systems" and left open the definition of a weaker property ;

b) We introduce the new concept of *t-weak-immunity* ; a mechanism is t-weak-immune if any altruistic player receives no worse payoff than the initial state, no matter how any set of t players deviate from the prescribed protocol. This new concept is sufficiently strong to capture the robustness of a large class of blockchain protocols (see Table 5.2) ;

c) We identify and prove necessary and sufficient conditions for a mechanism to be k-resilient and t-weak-immune ;

d) We define a new operator for game composition and prove that it preserves the robustness properties of the individual games ;

e) Using our generic framework and the composition operator we study the robustness to the behaviors of the two main types of blockchain users : *transacting parties* and *validators*, both modeled as rational and Byzantine agents. More precisely, robustness to miners (i.e., validators) behaviors is analyzed for Tendermint [1] and Bitcoin [2] (i.e., layer-1 protocols). While for layer-2 protocols (the Lightning Network protocol [3], the side-chain protocol [4] and the cross-chain protocol [5]) robustness is analyzed considering transacting parties' behaviors.

For each one of the analyzed protocols we provide bounds on the number of Byzantine processes in order to verify t-weak immunity. Furthermore, for the same class of protocols we compute bounds on the number of rational processes in order to achieve k-resilience. Our results are reported in Table 5.2. Interestingly, our analysis allowed us to spot the weakness of the Lightning Network protocol [3] to Byzantine behavior. Therefore, we propose and further analyze an alternative version of the protocol.

The chapter is structured as follows. Section 5.2 is devoted to the definition of mechanism, (k, t)-robustness, necessary and sufficient conditions for optimal resilience and weak immunity and, composition of mechanisms. We apply in Section 5.3 and Section 5.4 the methodology developed in Section 5.2 to prove the robustness of the protocols presented in [1; 2; 3; 4; 5].

TABLE 5.1 – Immunity and resilience properties for Tendermint [1], Bitcoin [2], Lightning Network [3], a side-chain protocol [4] and a cross-chain swap protocol [5; 6] with respect to the number of rational deviating agents (k) and the number of Byzantine deviating agents (t) where n is the total number of players in the game.

Protocol	k-resilience	t-immunity	t-weak immunity	Results
Tendermint	**Yes, k** $< n/3$	**No**	**Yes, t** $< n/3$	Thm. 5.5
Bitcoin	**Yes, k** $< 3n/20$	**No**	**No**	Thm. 5.7
Lightning Network	**Yes, k** $< 3n/20$	**No**	**No**	Thm. 5.9
Closing module	Yes	No	No	Thm. 5.12
(Alternative closing module)	(Yes)	(No)	(Yes)	Thm. 5.13
Other modules	Yes	No	Yes	Thm. 5.10, 5.11, 5.15, 5.18, 5.19
Side-chain (Platypus)	**Yes, k** $< n/3$	**No**	**Yes, t** $< n/3$	Thm. 5.20
Cross-chain Swap	**Yes**	**No**	**Yes**	Thm. 5.23

5.2 Game Theoretical Modeling for Blockchain Robustness

In a distributed protocol, agents who run it can either decide to follow the prescribed protocol or not. In case they do not, they deviate from the prescribed protocol by choosing a *byzantine* behavior. We would like to model these situations and understand whether the players are incentivized to follow the given advice. In [309] the authors introduce a game theoretical framework based on the concept of mechanism and its properties. In the following we recall and extend the framework of [309].

A game is a tuple $\Gamma = \langle N, \mathcal{S}, u \rangle$ in which the set of players N corresponds to the agents involved in a protocol. We map all the possible behaviors of the players and define them as their strategies $\mathcal{S}$. Following the protocol corresponds to one and only strategy $\sigma_i \in \mathcal{S}_i$ for every player i. For the sake of simplicity we assign utility $u_i(s) = 0$ for every $s \in \mathcal{S}$ when the player i is indifferent between the outcome of the strategy profile s and the outcome of the initial state, i.e. the utility given to the players before the game is played. Analogously we assign utility $u_i(s) > 0$ when the outcome of the strategy profile s corresponds to the final state provided by the protocol and $u_i(s) \leq 0$ when the outcome of s is worse than the initial state. The value of the utility corresponds to the marginal utility with respect to the initial state. The choice of the utility function is arbitrary, once the constraints above introduced are fulfilled.

Given the strategy profile $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_i, \ldots, \sigma_n) \in \mathcal{S}$ that corresponds to every player i following the protocol by playing strategy σ_i we define the mechanism (Γ, σ).

Definition 5.1 (mechanism [309]) *A mechanism is a pair* (Γ, σ) *in which* $\Gamma = \langle N, \mathcal{S}, u \rangle$ *is a game and* $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_i, \ldots, \sigma_n) \in \mathcal{S}$ *is a strategy profile.*

Every player is advised to play strategy $\sigma_i \in \mathcal{S}_i$. The game Γ shows all the possible strategies available to the players.

Players have a very low incentive to play weakly dominated strategies (see Definition 2.6) since they always have available a different strategy that provides no lower outcome in any scenario. A practical mechanism, formally defined below, ensures that these strategies are not included.

Definition 5.2 (practical mechanism [309]) *A mechanism* (Γ, σ) *is practical if* σ *is a Nash equilibrium of the game* Γ *after the iterated deletion of weakly dominated strategies.*

Evaluating the resilience of a distributed protocol to Byzantine behaviors corresponds to identifying the properties of the mechanism (Γ, σ). Users can decide to choose a Byzantine behavior for two different reasons. On one hand they can cooperate in order to find a strategy profile that provides a better outcome than the one given by the protocol, i.e. that increases any of their utilities. A mechanism which is optimal resilient, i.e., *practical* (see Definition 5.2) and *strongly resilient* (see Definition 5.3), discourages these behaviors. On the other hand some agents can behave maliciously for any reason and bring other players to unpleasant scenarios. In [309] a mechanism is t-immune to this behavior if it provides not inferior utility in the case when at most t players play a strategy different from the one prescribed by the mechanism. This condition has been already identified as being too strong in practice therefore we introduce the property of *t-weak-immunity* (see Definition 5.25), which means that a player i who chooses the prescribed strategy $\sigma_i \in \mathcal{S}_i$ is never lead to a worse state than the initial one, under the hypothesis that at most t players are byzantine.

In [309] the authors introduce a generalization of Nash equilibrium, k-resilient equilibrium defined formally below. The definition is a generalization of the concept of Nash equilibrium, which can be considered as a 1-resilient equilibrium. Indeed, in a Nash equilibrium no coalition formed by a single player has an incentive to change strategy. In a k-resilient equilibrium there is no coalition of k players that have an incentive to simultaneously change strategy to get a better outcome, i.e. when any of the players identifies a larger utility. Given a coalition of rational players $C \subseteq N$ of size up to $k : 1 \leq k < |N|$, the strategy profile $\sigma \in \mathcal{S}$ and any other of their strategy profiles $\tau_C \in \mathcal{S}_C$ we can define k-resilience as follows.

Definition 5.3 (k-resilient equilibrium [309]) *A strategy profile $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_i, \ldots, \sigma_n) \in \mathcal{S}$ is a k-resilient equilibrium if for all $C \subseteq N$ with $1 \leq |C| \leq k$, all $\tau_C \in \mathcal{S}_C$ and all $i \in C$, we have $u_i(\sigma_C, \sigma_{-C}) \geq u_i(\tau_C, \sigma_{-C})$.*

We say that a mechanism (Γ, σ) is *k-resilient* if σ is a k-resilient equilibrium for Γ.

If every strict subset of the players has no incentive to change strategy we say that the strategy profile is *strongly resilient* (formally, if it is k-resilient for all $k \leq n - 1$). We say that a mechanism (Γ, σ) is *strongly resilient* if σ is *strongly resilient.*

A mechanism (Γ, σ) is *optimal resilient* if it is practical and strongly resilient.

One of the basic assumption of game theory is that agents are rational. However, in real applications it might happen that agents behave irrationally. There are different reasons for this. Agents might have some limits that do not let them identify and choose rational behaviors. We always work under the assumptions that everything works, but there might be some technical failures that make some actions inaccessible to players. Lastly, the game might be not independent from other games. For instance, some agents might be subject to bribes which entice them to play an irrational strategy. Therefore it is interesting to study strategies that are immune to this type of behaviors. A strategy profile is t-immune if it provides not inferior utility in the case when at most t players play a strategy different from the one prescribed by the mechanism.

Definition 5.4 (t-immunity [309]) *A strategy profile $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_i, \dots, \sigma_n) \in \mathcal{S}$ is t-immune if for all $T \subseteq N$ with $|T| \leq t$, all $\tau_T \in \mathcal{S}_T$ and all $i \in N \setminus T$, we have $u_i(\sigma_{-T}, \tau_T) \geq u_i(\sigma)$. A mechanism (Γ, σ) is t-immune if σ is t-immune in the game Γ.*

The concept of k-resilience denotes the tendency of a set of k players to cooperate to move to a equilibrium different from the one prescribed. On the other hand, the concept of t-immunity evaluates the risk of a set of t players to defect and play a different strategy that can damage the other players. The two concepts are complementary. In [309] the authors introduced the notion of (k, t)-*robust* mechanism. A mechanism is (k, t)-*robust* if it is k-resilient and t-immune.

The property of t-immunity (see Definition 5.4) is too strong and difficult to be verified in practice because it requires that the protocol provided the *best outcome* no matter which strategy a set of t players choose. In [310] the author generalizes it with the definition of (t, r)-immunity, i.e., that players receive at least $u(\sigma) - r$ no matter what the other players do. For our purposes we need a more specific definition, that is valid for all players and that is related to a threshold, that we fix equal to zero. Since zero is the utility provided to players in their initial state, the property of immunity corresponds to guaranteeing at least the value of the initial state to every player. Given a coalition of Byzantine players $T \subseteq N$ of size up to $t : 1 \leq t < |N|$, their strategy profile $\tau_T \in \mathcal{S}_T$ and the set of strategies σ_{-T} of altruistic players $i \in N \setminus T$ we can define t-weak-immunity as follows.

Definition 5.5 (t-weak-immunity) *A strategy profile $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_i, \dots, \sigma_n) \in \mathcal{S}$ is t-weak-immune if for all $T \subseteq N$ with $|T| \leq t$, all $\tau_T \in \mathcal{S}_T$ and all $i \in N \setminus T$, we have $u_i(\sigma_{-T}, \tau_T) \geq 0$. A mechanism (Γ, σ) is t-weak-immune if σ is t-weak-immune in the game Γ.*

A player that joins a mechanism that is t-weak-immune knows that she does not suffer any loss (i.e., outcome with negative utility) if there are at most t Byzantine players in the game. Under the assumption that a protocol provides positive outcomes, a t-immune strategy is always t-weak-immune. As the denomination might suggest, this new property is weaker. Formally, it is possible to consider it as one of its generalizations. Indeed, if we consider the equivalent game $\Gamma' = \langle N, \mathcal{S}, u' \rangle$ with $u' = u - u(\sigma)$, the definition of t-immunity and t-weak-immunity are identical.
We say that a mechanism is *weak immune* if it is t-weak-immune for all $t \in N$ and that a mechanism is (k, t)-robust if it is k-resilient and t-weak-immune.

In Section 5.2.1 we provide necessary and sufficient conditions to prove that a mechanism satisfies the property of optimal resilience and t-weak-immunity. Then, we have to take into account that players run complex protocols composed of a set of modules. We introduce in Section 5.2.2 the operator *composition* of games (see Definition 5.6), i.e., the game that corresponds to different games run at the same time by the same players. We prove that the properties above introduced are invariant with respect to this operator, i.e., if two protocols are independent one from another they preserve their properties when played at the same time.

5.2.1 Necessary and Sufficient Conditions for Optimal Resilience and Weak Immunity

In the following we study the necessary and sufficient conditions for mechanisms to be *optimal resilient* and *weak immune*.

According to [309] if every strict subset of players has no incentive to change their strategy we say that the strategy profile is *strongly resilient.* (Γ, σ) is a *strongly resilient mechanism* if σ is strongly resilient. A mechanism (Γ, σ) is *optimal resilient* if it is practical and strongly resilient. The concepts of k-resilience and practicality are strictly connected with the properties of Nash equilibria, which have been fully studied (see for example [247; 311; 248; 249]). Therefore, connecting these two notions, through necessary and sufficient conditions, allow us to directly exploit the properties of Nash equilibria, such as *strength* [247] and *stability* [248; 249].

Proposition 5.1 (strong resilience) *If a mechanism (Γ, σ) is strongly resilient, then $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_i, \ldots, \sigma_n)$ is a strong equilibrium of Γ.*

Proof. *If (Γ, σ) is a strongly resilient mechanism, then for all $C \subset N$ and for all i such that*

$u_i(\sigma_C, \sigma_{-C}) \geq u_i(\tau_C, \sigma_{-C})$ (see Definition 5.3). Therefore, for all $C \subset N$ there always exists i such that $u_i(\sigma_C, \sigma_{-C}) \geq u_i(\tau_C, \sigma_{-C})$, which corresponds to the Definition 2.5 of strong equilibrium. ■

Strong Nash equilibria are easy to be identified, but they are very rare ; indeed, they do not always exist [247]. Therefore, the property of strongly resiliency is even more rare. We thus take into account a different concept of solution, that of stable Nash equilibrium, which tries to identify those Nash equilibria that are more likely to be played. According to definition provided in [248], stable equilibria fulfill different properties, among which they survive the iterated deletion of weakly dominated strategies. The concept of stable equilibria, which is well studied in literature [248; 249] extends the concept of practical mechanism.

Proposition 5.2 (practicality) *If $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_i, \ldots, \sigma_n)$ is a stable equilibrium of Γ, then the mechanism (Γ, σ) is practical.*

Proof. *Stable equilibria survive after the iterated deletion of weakly dominated strategies, therefore the mechanism is practical.* ■

In [248] the authors proves that there always exists at least one stable Nash equilibrium, that leads us to the following corollary.

Corollary 5.0.1 *For any game Γ there is always at least one $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_i, \ldots, \sigma_n) \in \mathcal{S}$ such that the mechanism (Γ, σ) is practical.*

Indeed, since for every game $\Gamma = \langle N, \mathcal{S}, u \rangle$ there always exists a stable equilibrium $\sigma \in \mathcal{S}$, from Proposition 5.2 we have that (Γ, σ) is practical.

We now know prove that the properties of strongly resiliency and practicality are independent, and therefore both of them have to be studied. This result comes from the fact that also strength and stability are independent [249].

Proposition 5.3 *The property of strongly resiliency and practicality are independent.*

Proof. *In order to prove the independence we have to identify 4 examples of mechanism with the following properties : (i) strongly resilient and practical, (ii) strongly resilient and not practical, (iii) not strongly resilient and practical, (iv) not strongly resilient and not practical :*

1. *We define the mechanism (Γ, σ) such that for all i we have that $u_i(\sigma) = 1$ and $u_i(\tau) = 0$ for all $\tau \neq \sigma$. The mechanism is strongly resilient. The strategy profile σ is the only Nash equilibrium. Since there always exists a stable Nash equilibrium, σ is stable and thanks to Proposition 5.2 we have that (Γ, σ) is practical.*
2. *Let us consider the mechanism (Γ, σ), in which Γ has two players, for all i and for all $\tau \in \mathcal{S}$ we have that $u_i(\tau) = 1$, but for $\overline{\tau} = (\sigma_1, \overline{\tau}_2)$ with $\overline{\tau}_2 \neq \sigma_2$ which provides utility $u_i(\overline{\tau}) = (0, 1)$. The mechanism (Γ, σ) is strongly resilient, because $u_i(\sigma) \geq u_i(\tau)$ for all i and all $\tau \in \mathcal{S}$. However, it is not practical, as player 1 would not consider $\sigma_1 \in \mathcal{S}_1$, but a different strategy that always provides utility equal to 1.*
3. *Since strength and stability are independent [249], there always exists a game Γ in which an equilibrium σ is stable, but not strong. The mechanism (Γ, σ) is not strongly resilient (thanks to Proposition 5.1 and it is practical, since it is stable (Proposition 5.2).*
4. *It is enough to define a mechanism (Γ, σ) such that σ is not a Nash equilibrium of Γ.*

■

The following proposition provides a necessary and sufficient condition to determine if a mechanism is weak immune.

Proposition 5.4 (weak immunity) *A strategy profile $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_i, \ldots, \sigma_n) \in \mathcal{S}$ is weak immune if and only if for all $i \in N$ in the game $\Gamma_i = \langle N', \mathcal{S}', u' \rangle$ with $N' = \{i, j\}$, $\mathcal{S}'_i = \mathcal{S}_i$, $\mathcal{S}'_j = \mathcal{S}_1 \times \mathcal{S}_2 \times \cdots \times \mathcal{S}_{i-1} \times \mathcal{S}_{i+1} \times \cdots \times \mathcal{S}_n$, $u'_i = u_i$ and $u'_j = -u_i$ the best response $\tau'_j \in S'_j$ to u'_i gives outcome $u'_i(\sigma_i, \tau'_j) \geq 0$.*

Proof. *Let us prove the if part. Since τ'_j is a best response to σ_i, by definition $u'_j(\sigma_i, \tau'_j) \geq u'_j(\sigma_i, \tau')$ for all $\tau' \in \mathcal{S}_j$. Therefore $u'_i(\sigma_i, \tau'_j) \leq u'_i(\sigma_i, \tau')$ and so for all $\tau' \in \mathcal{S}_j$ we have that $u'_i(\sigma_i, \tau') \geq 0$. By construction for every $\tau_{-i} \in \mathcal{S}_{-i}$ there is one and only one $\tau' \in \mathcal{S}'_j$, which gives $u_i(\sigma_i, \tau_{-i}) = u'_i(\sigma_i, \tau'_j)$. Hence we have that $u_i(\sigma_i, \tau_{-i}) \geq 0$ for all $\tau_{-i} \in \mathcal{S}_{-i}$. The proof for the only if part is analogous, since we can find a one-to-one correspondence among strategies in $\mathcal{S}$ and $\mathcal{S}'$.* ■

The principle is to fix one player $i \in N$ at a time and consider all the other players as a unique adversarial player j that sets her strategy in order to reduce the utility of player i. The game Γ_i in which player i faces an adversarial player j belongs to a specific class of games, called *two-player zero-sum games* [312], whose Nash equilibria are always in the form $(v, -v)$ with $v \in \mathbb{R}$. The term v is called *value of the game* and corresponds to the minimum value that player i is able to achieve. Proposition 5.4 states that a strategy profile is weak immune if and only if the *best response* (i.e., the strategy producing the most favorable outcome) for the adversarial player j assigns to player i a positive outcome $v \geq 0$. This condition allows us to check the weak immunity property by looking at only N outcomes from N games, which is more efficient than considering all the possible outcomes of the game Γ. We see in Section 5.4.1.2 how this condition allows us to verify the weak immunity of a mechanism.

5.2.2 Composition of Games and Mechanisms

Blockchains systems are complex protocols designed in a modular way. In order to study the robustness of such complex protocols we analyze the robustness of the individual modules and infer the properties of the system by composition.

We introduce therefore the notion of *composition of games*. Given two different games A and B, the game $A \odot B$ corresponds to players picking a strategy from each game and receiving as utility the sum of the utilities of the two games. The games are intended to be played separately and independently.

Definition 5.6 *Given $A = \langle N, \mathcal{S}_A, u_A \rangle$ and $B = \langle N, \mathcal{S}_B, u_B \rangle$ two games in normal form with the same set of players N, two different sets of strategies $\mathcal{S}_A = \{\mathcal{S}_{Ai} : i \in N\}$ and $\mathcal{S}_B = \{\mathcal{S}_{Bi} : i \in N\}$ and two different utility functions : $u_A : \mathcal{S}_A \to \mathbb{R}^N$ and $u_B : \mathcal{S}_B \to \mathbb{R}^N$ then, it is possible to define a new game $C = A \odot B$, called composition of A and B, which is characterized as follows. $C = \langle N, \mathcal{S}_C, u_C \rangle$, where :*

- *N is the set of the players,*
- *$\mathcal{S}_C := \{(s_{Ai}, s_{Bi}), s_{Ai} \in \mathcal{S}_{Ai}, s_{Bi} \in \mathcal{S}_{Bi}, \forall i \in N\}$ is the set of strategies,*
- *$u_C(\{(\sigma_{Ai}, \sigma_{Bi})\}) := u_A(\{\sigma_{Ai}\}) + u_B(\{\sigma_{Bi}\})$ is the utility function.*

In the context of non-cooperative games linear transformations of utility functions ($u'_i = a \cdot u_i + b$ with $a \in \mathbb{R}^+$ and $b \in R$) are considered invariant transformations since they preserve the main properties of the game [313]. Therefore, defining the utility function of the composition of games as the sum of the utility functions is equivalent to defining it for any linear combination. It is possible to extend the definition of composition of games to pairs of games in which different sets of players are involved. Indeed, for instance if a player i is involved in game A but not in game B, it is possible to extend game $B = \langle N, \mathcal{S}_B, u_B \rangle$ to $B = \langle N', \mathcal{S}'_B, u'_B \rangle$ in which player i is added ($N' = N \cup \{i\}$) and she is assigned a "null" strategy ($\mathcal{S}'_B = \mathcal{S}_B \times \{\sigma_\emptyset\}$) not influencing the utilities of the outcomes. Formally, for all $s \in \mathcal{S}_B$ and for all $j \in N' \setminus \{i\}$, $u'_j(s, \sigma_\emptyset) = u_j(s)$, while for $i \in N'$ we have that $u_i(s, \sigma_\emptyset) = 0$. Intuitively it is possible to extend the definition of composition of games to more than two games. In Section 5.4.1.5 we use the notation $A \odot B \odot C$ to represent either game $A \odot (B \odot C)$ or $(A \odot B) \odot C$. We do not prove the associative property of this operator, but it is intuitive that the two games are the same, except for a different strategy labelling.

The following theorems allow us to model the building blocks of complex protocols, study the properties of the subsequent mechanisms and finally, through the composition of mechanisms, deduce the properties of the composed protocol.

Theorem 5.1 *Let $A = \langle N, \mathcal{S}_A, u_A \rangle$ and $B = \langle N, \mathcal{S}_B, u_B \rangle$ be two games in normal form representation. Then, $\{(\sigma_{Ai}, \sigma_{Bi})\}$ is a Nash equilibrium for $A \odot B$ if and only if $\{\sigma_{Ai}\}$ and $\{\sigma_{Bi}\}$ are Nash equilibria respectively for A and B.*

Proof. *Let us prove the if part. If $\{\sigma_{Ai}\}$ and $\{\sigma_{Bi}\}$ are Nash equilibria for A and B, then $\forall j$ and for any other pair of strategies for player j, σ'_{Aj} and σ'_{Bj} we have that :*

$$u_A(\{\sigma_{Aj}, \sigma_{A-j}\}) \geq u_A(\{\sigma'_{Aj}, \sigma_{A-j}\}) \text{ and } u_B(\{\sigma_{Bj}, \sigma_{B-j}\}) \geq u_B(\{\sigma'_{Bj}, \sigma_{B-j}\})$$

where $-j := \{i \in N : i \neq j\}$. Hence, for any other $\{(\sigma'_{Aj}, \sigma'_{Bj}), (\sigma_{A-j}, \sigma_{B-j})\}$ it is possible to deduce that :

$$u_{A\odot B}(\{(\sigma_{Ai}, \sigma_{Bi})\}) := u_A(\{\sigma_{Ai}\}) + u_B(\{\sigma_{Bi}\}) \geq$$

$$\geq u_A(\{\sigma'_{Aj}, \sigma_{A-j}\}) + u_B(\{\sigma'_{Bj}, \sigma_{B-j}\}) =: u_{A\odot B}(\{(\sigma'_{Aj}, \sigma'_{Bj}), (\sigma_{A-j}, \sigma_{B-j})\})$$

that is, $\{(\sigma_{Ai}, \sigma_{Bi})\}$ is a Nash equilibrium for $A \odot B$.
Let us prove the only if part by contradiction, i.e., $\exists\{(\sigma_{Ai}, \sigma_{Bi})\}$ that is a Nash equilibrium for $A \odot B$ but at least one among $\{\sigma_{Ai}\}$ and $\{\sigma_{Bi}\}$ is not a Nash equilibrium for A or B. Let us suppose that $\{\sigma_{Ai}\}$ is not a Nash equilibrium for A : $\exists j, \exists \sigma'_A : u_A(\{\sigma_{Aj}, \sigma_{A-j}\}) < u_A(\{\sigma'_{Aj}, \sigma_{A-j}\})$ then,

$$u_{A\odot B}(\{(\sigma_{Ai}, \sigma_{Bi})\}) := u_A(\{\sigma_{Ai}\}) + u_B(\{\sigma_{Bi}\}) <$$

$$< u_A(\{\sigma'_{Aj}, \sigma_{A-j}\}) + u_B(\{\sigma_{Bj}, \sigma_{B-j}\}) =: u_{A\odot B}(\{(\sigma'_{Aj}, \sigma_{Bj}), (\sigma_{A-j}, \sigma_{B-j})\}$$

which contradicts the hypothesis that $\{(\sigma_{Ai}, \sigma_{Bi})\}$ *is a Nash equilibrium for* $A \odot B$. ■

The Nash equilibria can be identified by selecting equilibria within the single games. It is not possible to create other Nash equilibria nor to lose them in the process of composition of the games.

Concerning robustness properties for composition of games, we can state the following results on resiliency and weak immunity for two composed games. The results can be generalized for the composition of multiple games.

Theorem 5.2 *Let* $A = \langle N, \mathcal{S}_A, u_A\rangle$ *and* $B = \langle N, \mathcal{S}_B, u_B\rangle$ *be two games,* (A, σ_A) *and* (B, σ_B) *two practical mechanisms. Then,* $(A \odot B, \{\sigma_{Ai}, \sigma_{Bi}\})$ *is a practical mechanism.*

Proof. *Thanks to Theorem 5.1 we have that* $\{\sigma_{Ai}, \sigma_{Bi}\}$ *is a Nash equilibrium for* $A \odot B$. *It is sufficient to prove that it survives the iterated deletion of weakly dominated strategy. Indeed, every strategy in the form* (τ^*_{Ai}, τ_{Bi}) *or* (τ_{Ai}, τ^*_{Bi})*, where* τ^*_A *is weakly dominated in* A *and* τ^*_B *is weakly dominated in* B *for some player* i*, is weakly dominated by another Nash equilibrium in* $A \odot B$ *for the very same player* i*. The strategy profile* $\{\sigma_{Ai}, \sigma_{Bi}\}$ *survives the iterated deletion of these weakly dominated strategies. It is now sufficient to prove that there is no other weakly dominated strategy. By contradiction we assume that there is a player* i *such that there exists* $(\bar\sigma_{Ai}, \bar\sigma_{Bi}) \in \mathcal{S}_{A\odot B}$ *that weakly dominates* $(\sigma_{Ai}, \sigma_{Bi})$*. Therefore, considering the utility* u *for the player* i*, for every* $(\tau_{A,-i}, \tau_{B,-i}) \in \mathcal{S}_{A\odot B,-i}$ *we have that :*

$$u_{A\odot B}(\{(\bar\sigma_{Ai}, \bar\sigma_{Bi}), (\tau_{A,-i}, \tau_{B,-i})\}) \geq u_{A\odot B}(\{(\sigma_{Ai}, \sigma_{Bi}), (\tau_{A,-i}, \tau_{B,-i})\}).$$

Since σ_{Ai} *is not dominated by* $\bar\sigma_{Ai}$ *in the game* A*, there exists* $\bar\tau_{A,-i} \in \mathcal{S}_{A,-i}$ *such that* $u_A(\bar\sigma_{Ai}, \bar\tau_{A,-i}) < u_A(\sigma_{Ai}, \bar\tau_{A,-i})$*. Analogously there exists* $\bar\tau_{B,-i} \in \mathcal{S}_{B,-i}$ *such that* $u_B(\bar\sigma_{Bi}, \bar\tau_{B,-i}) < u_B(\sigma_{Bi}, \bar\tau_{B,-i})$*. Therefore we have that :*

$$u_{A\odot B}(\{(\bar\sigma_{Ai}, \bar\sigma_{Bi}), (\bar\tau_{A,-i}, \bar\tau_{B,-i})\}) < u_{A\odot B}(\{(\sigma_{Ai}, \sigma_{Bi}), (\bar\tau_{A,-i}, \bar\tau_{B,-i})\}),$$

which contradicts the assumption. ■

Theorem 5.2 formalizes the intuition that if two mechanisms are practical then, playing both selected strategy profiles is still a practical mechanism. Following propositions prove the resilience and immunity of the games composition.

Theorem 5.3 *Let $A = \langle N, \mathcal{S}_A, u_A \rangle$ and $B = \langle N, \mathcal{S}_B, u_B \rangle$ be two games, (A, σ_A) and (B, σ_B) two mechanisms respectively k-resilient and k'-resilient. Then, $(A \odot B, \{\sigma_{Ai}, \sigma_{Bi}\})$ is a $\min(k, k')$-resilient mechanism.*

Proof. *We know that for all $C \subseteq N$ with $1 \leq |C| \leq k$, all $\tau_{A,C} \in \mathcal{S}_{A,C}$ and all $i \in C$, we have $u_{Ai}(\sigma_{A,C}, \sigma_{A,-C}) \geq u_i(\tau_{A,C}, \sigma_{A,-C})$. Analogously, for all $C' \subseteq N$ with $1 \leq |C'| \leq k'$, all $\tau_{B,C'} \in \mathcal{S}_{B,C'}$ and all $i \in C'$, we have $u_{Bi}(\sigma_{B,C'}, \sigma_{B,-C'}) \geq u_i(\tau_{B,C'}, \sigma_{B,-C'})$. Hence, we have that for all $S \subseteq N$ with $1 \leq |S| \leq \min(k, k')$, all $(\tau_{A,S}, \tau_{B,S}) \in \mathcal{S}_{A,S} \times \mathcal{S}_{B,S}$ and all $i \in S$:*

$$u_{Ai}(\sigma_{A,S}, \sigma_{A,-S}) + u_{Bi}(\sigma_{B,S}, \sigma_{B,-S}) \geq u_i(\tau_{A,S}, \sigma_{A,-S}) + u_i(\tau_{B,S}, \sigma_{B,-S}).$$

We recall that $\mathcal{S}_{A \odot B,S} = \mathcal{S}_{A,S} \times \mathcal{S}_{B,S}$, thus for all $S \subseteq N$ with $1 \leq |S| \leq \min(k, k')$, all $(\tau_{A,S}, \tau_{B,S}) \in \mathcal{S}_{A \odot B,S}$ and all $i \in S$:

$$u_{A \odot B,i}(\{\sigma_{A,S}, \sigma_{B,S}\}, \{\sigma_{A,-S}, \sigma_{B,-S}\}) \geq u_{A \odot B,i}(\{\tau_{A,S}, \tau_{B,S}\}, \{\sigma_{A,-S}, \sigma_{B,-S}\}).$$

■

If a mechanism is k-resilient, then the protocol is followed by every player whenever at most k rational players are allowed. If there is more than one mechanism, the threshold on the maximum number of rational players allowed is the minimum among the rational player numbers k, k' in the individual mechanisms.

Theorem 5.4 *Let $A = \langle N, \mathcal{S}_A, u_A \rangle$ and $B = \langle N, \mathcal{S}_B, u_B \rangle$ be two games, (A, σ_A) and (B, σ_B) two mechanisms respectively t-weak-immune and t'-weak-immune. Then, $(A \odot B, \{\sigma_{Ai}, \sigma_{Bi}\})$ is a $\min(t, t')$-weak-immune mechanism.*

Proof. *In game A, for all $T \subseteq N$ with $|T| \leq t$, all $\tau_{A,T} \in \mathcal{S}_{A,T}$ and all $i \in N \setminus T$, we have $u_{Ai}(\sigma_{A,-T}, \tau_{A,T}) \geq 0$. In game B, for all $T \subseteq N$ with $|T| \leq t'$, all $\tau_{B,T} \in \mathcal{S}_{B,T}$ and all $i \in N \setminus T$, we have $u_{Bi}(\sigma_{B,-T}, \tau_{B,T}) \geq 0$. Therefore we have that for all $T \subseteq N$ with $1 \leq |T| \leq \min(t, t')$, all $(\tau_{A,T}, \tau_{B,T}) \in \mathcal{S}_{A,T} \times \mathcal{S}_{B,T}$ and all $i \in N \setminus T$:*

$$u_{A \odot B,i}(\{\sigma_{A,T}, \sigma_{B,T}\}, \{\tau_{A,-T}, \tau_{B,-T}\}) = u_{Ai}(\sigma_{A,T}, \tau_{A,-T}) + u_{Bi}(\sigma_{B,S}, \tau_{B,-S}) \geq 0$$

■

If a player combines two mechanisms which are weak immune for respectively at most t and t' Byzantine players, then it means that she is considering a mechanism which can provide non-negative outcomes if there are at most a number of Byzantine users equal to $\min(t, t)'$.

The following corollaries generalize the results reported in Theorem 5.3 and Theorem 5.4.

Corollary 5.4.1 *Let $A_1, A_2, \dots A_n$ with $n \in N$ be games and let $(A_1, \sigma_{A_1}), (A_2, \sigma_{A_2}), \dots (A_n, \sigma_{A_n})$ be the corresponding mechanisms respectively $k_1, k_2, \dots k_n$-resilient.*
Then, $(A_1 \odot A_2 \odot \cdots \odot A_n, \{\sigma_{A_1}, \sigma_{A_2}, \dots, \sigma_{A_n}\})$ is a $\min(k_1, k_2, \dots, k_n)$-resilient mechanism.

Corollary 5.4.2 *Let $A_1, A_2, \dots A_n$ with $n \in N$ be games and let $(A_1, \sigma_{A_1}), (A_2, \sigma_{A_2}), \dots (A_n, \sigma_{A_n})$ be the corresponding mechanisms respectively $t_1, t_2, \dots t_n$-weak-immune.*
Then, $(A_1 \odot A_2 \odot \cdots \odot A_n, \{\sigma_{A_1}, \sigma_{A_2}, \dots, \sigma_{A_n}\})$ is a $\min(t_1, t_2, \dots, t_n)$-weak-immune mechanism.

5.3 Robustness to Validating Users' Behaviors in Layer-1 protocols

In this section we prove the effectiveness of our framework by analyzing the robustness of different blockchain layer-1 protocols. Section 5.3.1 and 5.3.2 analyze Tendermint [1] [1] and Bitcoin [2], respectively. Blockchain users modeled as rational and Byzantine agents are *validating nodes* (i.e., Tendermint validating nodes and Bitcoin miners analyzed in Chapter 3) who participate to the blockchain consensus phase.

5.3.1 Tendermint

Tendermint's consensus protocol (i.e., Tendermint-core [1; 8]) is split into three rounds : the Pre-Propose round, the Propose round and the Vote round. During the Pre-Propose round, the proposer presents a block, to the other participants. During the Propose round, each participant chooses whether to accept or not the block and broadcasts her decision. If the votes for the proposal exceed a predetermined threshold ν then participants start the Vote phase. If the block receives more than ν votes, it is validated. Tendermint's consensus algorithm sets $\nu = n - f = \frac{2}{3}n$; the threshold representing the number of non-faulty actors (as n denotes the total number of nodes and f the total number of faulty nodes) is set to $\frac{2}{3}$ of the network participants.

The set of actions available to consensus participants is described in Figure 5.1.

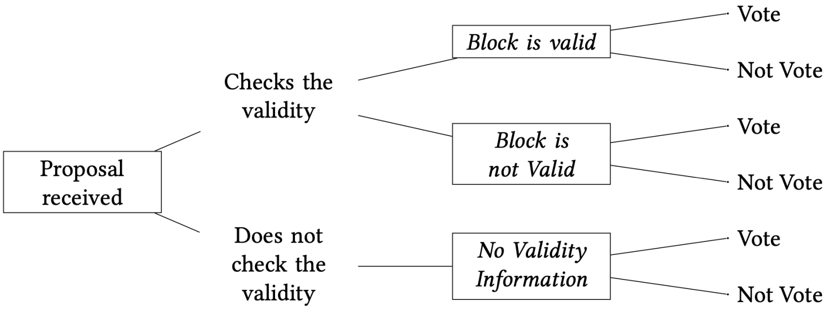


FIGURE 5.1 – Strategies available to participants [8].

Definition 5.7 *The Tendermint game is a mechanism $(\Gamma^{tc}, \sigma^{tc})$ such that the game Γ^{tc} represents the decision-making problem and the strategy σ^{tc} is the prescribed consensus protocol. Once a proposal v*

[1] This contribution analyzes the one shot consensus protocol at the core of the Tendermint blockchain.

is received, N players choose either to check or not to check the validity of the value, then they can choose either to Vote or Not to Vote for it. At the very first stage of the game (stage a) a player can choose either to check (C) the validity or not check (NC). If she checks it, she can choose to Vote or Not Vote for it, in case value v is valid (stage b) or not (stage c). If she does not check it (stage d), she can choose to Vote (V) or Not Vote (NV) for it. Every strategy τ is represented by a vector (a, b, c, d) in which $a \in \{C, NC\}$, $b, c, d \in \{V, NV\}$. The utility for player i is $u_i(\tau) = 1$ if a valid block is approved or a non-valid block is not approved, $u_i(\tau) = 0$ if a valid block is not approved and $u_i(\tau) < 0$ if a non-valid block is approved.

The strategy prescribed by Tendermint consensus protocol is $\sigma^{tc} = (C, V, NV, NV)$ i.e., to check for the validity of the proposal and then if the block is valid to vote for it, otherwise not vote for it. If the number of rational or byzantine players allowed is $f < \frac{1}{3}n$, the other players have the necessary threshold to validate a block. Indeed, they can veto any validation of blocks proposed by malicious nodes. The mechanism $(\Gamma^{tc}, \sigma^{tc})$ is thus not f-weak-immune for any $f \geq \frac{1}{3}n$ and we can state the following results.

Theorem 5.5 *The mechanism $(\Gamma^{tc}, \sigma^{tc})$ is (f, f)-robust for any $f < \frac{1}{3}n$.*

Proof. *First, let's consider the case in which the proposer puts forward a non-valid block. If $f < \frac{1}{3}n$ is the number of players who deviate, then at most $\frac{1}{3}n$ will vote for the non-valid block, which is less then the threshold $\nu = \frac{2}{3}n$ asked by the consensus algorithm to validate the block.*
Let's thus consider the case of the proposer putting forward a valid block. The $n - f$ altruistic player will vote in favour of validating the block. Since $n - f \geq \frac{2/3}{n} + 1$, the threshold ν is overcome. ■

If there are at least $f \geq \frac{1}{3}n$ byzantine players, it is possible to this set of players to veto any validation of blocks. The mechanism $(\Gamma^{tc}, \sigma^{tc})$ is thus not f-weak-immune for any $f \geq \frac{1}{3}n$. From now on, we exclude the case that no blocks are validated.

Theorem 5.6 *The mechanism $(\Gamma^{tc}, \sigma^{tc})$ is not f-weak-immune for any $f \geq \frac{1}{3}n + 2$.*

Proof. *It is enough to prove that if there are $f = \frac{1}{3}n + 2$ byzantine players, a non-valid block is approved. Let us suppose that the players are split in 3 sets : altruistic players are divided in two set A and A' of dimension $\frac{1}{3}n - 1$, while byzantine players are part of the third set B. Let us suppose that the proposer is a byzantine player. She sends two different incompatible values v and v' to the players* *respectively in A and A'. Then, during the Propose and Vote phase, all the players in B broadcast to player A and A' respectively their vote in favour of v and v'. Both players in A and A' are satisfied, as the threshold of $\frac{n}{3} + 1$ votes is met, so they both broadcast the values v and v' which are however incompatible.* ■

We considered the case for a generic n, which cannot give results about weak immunity for values $\frac{n}{3}$ and $\frac{1}{3}n + 1$. Tendermint [1] considers only the case of a specific $n = 3f + 1$ number of players, with threshold $\nu = 2f + 1$. In this specific case, Theorem 5.5 consists in setting t, the number of byzantine players, the following condition for weak immunity : $t < \frac{n}{3} < f + 1$. On the other hand, we can state from Theorem 5.6 that we do not have weak immunity for $t \geq \frac{1}{3}n + 2 > f + 2$. With a similar argument to the one proposed in Theorem 5.6 it is possible to prove that for $t = f + 1$ or $t = f + 2$ Tendermint's protocol does not fulfill weak immunity.

5.3.2 Bitcoin

Bitcoin is a permissionless blockchain based on the Proof-of-Work mechanism [2] where every user has a chance to publish a new block in the distributed ledger. The probability of user to publish/mine a new block (i.e., validator) is proportional to her computational power α. Bitcoin's protocol requires that once a block is mined, it should be broadcast to every other user. In case two or more blocks are mined at the same moment, the players split their effort to mine from any of the blocks (i.e., a *fork* is generated). Hence, published blocks are not automatically validated ; they are considered as valid when belonging to the *longest chain* i.e., the longest branch of the ledger called *main chain*.

As for Tendermint, Bitcoin's protocol can be represented by a mechanism $(\Gamma^{btc}, \sigma^{btc})$. We take into account the worst-case scenario, in which the byzantine users coordinate, thus they are represented by a single player i. The altruistic users act in the same way and can therefore be represented by a second player j. The strategies of the players correspond to choosing (i) where in the chain add a new block and (ii) when to publish the mined blocks. Player j plays only one strategy defined by σ^{btc} i.e., she follows the protocol by mining on the main chain (the longest one) or splitting her effort if there is more than one chain of the same length available. Since the game is stochastic, we group all the equivalent states of the game in the same class. We consider two states as equivalent if they have the same configuration independently from the precise position in the chain (i.e., the difference between the number of mined blocks by the i and j is the same). In the Bitcoin blockchain a best practice is to consider a block as valid if belonging to a chain where at least B (usually, $B = 6$) blocks have been published afterwards, because it is presumably considered impossible to create a longer chain that does not include it. This block is invalidated if a fork is made at the previous block and more than $B+1$ blocks are published starting from it. In this way, the block does not belong to the longest chain anymore and it is not considered valid.

Definition 5.8 *The Bitcoin game is a mechanism* $(\Gamma^{btc}, \sigma^{btc})$ *such that the game* Γ^{btc} *represents the decision-making problem and the strategy* σ^{btc} *is the prescribed protocol. The game* Γ^{btc} *is characterized by two players* i *and* j*, who have respectively mining power* α *and* $1 - \alpha$ *and every state of the game can be represented by the state class* $\{x_k\}_{k \in \{0,1,\ldots,B+1\}}$*, where* x_k *is the number of blocks mined, yet not published, at level* k *by player* i*. The block at level* $k = 0$ *is the only one to be published. The initial state of the game is* $\{x_k = 0\}$ $\forall k \in \{0, 1, \ldots, B+1\}$*, while the final state of the game is represented by the state class with value* $x_{B+1} \geq 1$*. While player* j *has only one possible strategy* σ^{btc}*, player* i *can choose which branches to mine from (i.e. at which level* k *add the block). The utility of the players corresponds to the number of bitcoins they own.*

The game theoretical framework let us state the following results on Bitcoin's mechanism robustness. Any subset of players T with $|T| = t$ having mining power $\alpha > 0$ have a small probability, not negligible, to perform a successful attack, by building a longer chain which does not include a block which was already considered valid (Theorem 5.7).

Proposition 5.5 *The probability for a player with mining power* α *to find* n *blocks before any other player can find at most* m *blocks is*

$$P(n, m) = \sum_{i=n}^{n+m-1} \binom{n+m-1}{i} \alpha^i (1-\alpha)^{n+m-1-i}.$$

Proof. *It's enough to compute that among* $n + m - 1$ *blocks at least* n *are being mined by the player with mining power* α*.*

$$P(n, m) = \sum_{i=n}^{n+m-1} \binom{n+m-1}{i} \alpha^i (1-\alpha)^{n+m-1-i}.$$

■

If $m = 1$ we have that $P(n, 1) = \alpha^n$.

Theorem 5.7 *The Bitcoin mechanism* $(\Gamma^{btc}, \sigma^{btc})$ *is not* t*-weak-immune for any* t.

Proof. *Let us suppose that a player with mining power* $\alpha > 0$ *plays the strategy of block withholding if she has mined more blocks than the main chain. When she reaches more than* B *blocks than the main chain, she publishes all of them, thus invalidating the others. Due to Proposition 5.5, at every new block she has approximately probability* $\alpha^B > 0$ *to perform the attack. When the number of attempts goes to* ∞*, the probability to perform the attack goes to* 1*. Thus it is almost sure that any subset of players* T *of cardinality* $|T| = t$ *with total mining power* α*, with* $\alpha > 0$*, the attack will be successfully performed.* ■

Theorem 5.8 *The Bitcoin mechanism* $(\Gamma^{btc}, \sigma^{btc})$ *is* k*-resilient if* k *players have at most* $\alpha \leq \frac{3}{20}$ *as total mining power.*

Proof. *Let us suppose that* i *is a rational player and* j *is an altruistic player, i.e. it follows the protocol. The goal of player* i *is to maximise the number of bitcoins owned by her. A combination of attacks can make her owning more bitcoins than the ones she would receive by following the protocol. Specifically, she can gain bitcoins by double spending them. She validates a block in which she spends* M *bitcoins, then she creates a fork before the block which creates a longer chain from. Performing such selfish mining attack [300] makes the player* i *lose some bitcoins. Indeed, creating forks includes the risk of creating blocks which can eventually not belong to the longest chain; thus, the reward* R *given by these blocks would get lost. Let us define* $N(\alpha)$*, the average number of blocks lost in the attack by player* i*, who has computation power* α*. The player* i *chooses to perform the attack if* $M > R \cdot N(\alpha)$*, i.e. if the double spended bitcoins are greater than the average reward lost in the attack. The values of* M *and* R *are parameters, while* $N(\alpha)$ *depends on the strategy chosen by* i*. Since* i *is a rational player, she chooses the optimal strategy, i.e. the strategy that minimises* $N(\alpha)$*.*

We thus have to identify the optimal strategy. The probability for player i *to add a new block depends on how the other players are split in mining the other blocks. If there are* m *forks, there is* $\alpha' := \frac{\alpha}{\alpha + \frac{1-\alpha}{m+1}}$ *chance for player* i *to mine the next block and* $1 - \alpha' = \frac{\frac{1-\alpha}{m+1}}{\alpha + \frac{1-\alpha}{m+1}}$ *chance for player* j *to mine the next block. Player* i *has probability* α' *to add a new block at the level* k *that she chooses, i.e. to add* 1 *to the value of any* x_k*. Player* j *has probability* $1 - \alpha'$ *to mine a block, which is added at level* $k = 0$*. The chain is increased by one level, i.e. the number of forks* x_1 *created at level* $k = 1$ *are published. The states are moved by one position, i.e.* $x_k \to x_{k-1}$*. From every state* $s = \{x_k\}_{k \in \{0,1,\ldots,B+1\}}$ *player*

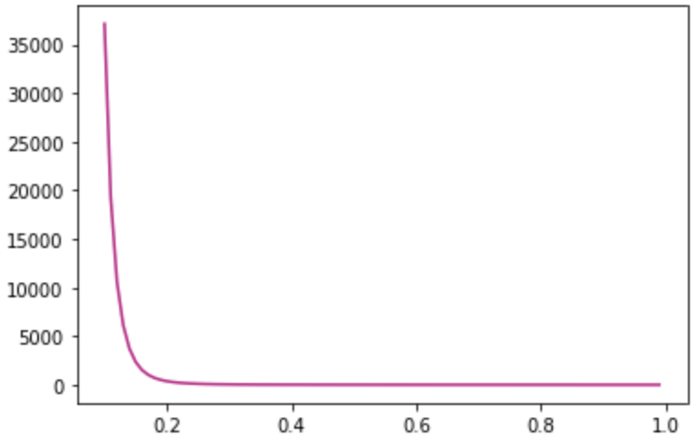


FIGURE 5.2 – Number of blocks $N(\alpha)$ to be mined by player i with computational power α under the optimal attacking strategy.

i has to mine on average N_s blocks before getting to the final state, following the optimal strategy, i.e. the strategy that minimizes the number of blocks to be mined. The problem has infinite states, because at any level k player i can create $x_k \in \mathcal{N}$ forks. We thus fix a maximum number of blocks $L \in \mathcal{N}$ that can be mined at the same level (i.e. we set $x_k \leq L$) and consider the equivalent problem with a finite number of states. In order to find the optimal solution we compute the optimal Bellman operator [314], which provides the solution in close form. We find out that even increasing L, for significant (> 0.05) values of α the optimal strategy is to perform a selfish mining attack and create only one fork at every level. Figure 5.2 shows the average number of blocks that player i has to mine in order to perform an attack.

On average every block contains transactions for $M = 10000$ BTC. Mining a block is worth $R = 6.25$ BTC. Therefore an attack is rationally chosen by player i if $N(\alpha) < \frac{M}{R} = \frac{10000}{6.25} = 1600$. Since $N(0.15) = 2347 > 1600$, we have the proof. ■

The Bitcoin's mechanism can be made more resilient by reducing the number of bitcoins exchanged in a block M, the reward of a block R and the number of blocks B needed for validation.

On the long run the majority of users ($\alpha \geq \frac{1}{2}$) produce the longer chain. However, on the short run a minority of users ($\alpha < \frac{1}{2}$) can make a fork the longer chain with positive probability. The following theorem provides the value of this probability.

Theorem 5.9 *The probability for a byzantine player with computation power α, with $\alpha < \frac{1}{2}$, to prevent* *a transaction to be published within $\Delta > 0$ blocks is :*

$$\Phi_\Delta(\alpha) = \frac{\alpha}{1-\alpha} - \sum_{k=1}^{\Delta-1} (1 - \Phi_{\Delta-k}(\alpha)) \cdot \alpha^k \cdot (1-\alpha)^k \cdot M(k),$$

where $M(k)$ is a function defined in [315] that maps natural numbers to the sequence $1, 1, 2, 5, 13, 42 \ldots$.

Proof. *Let us suppose that a transaction is published on the main chain on the next block, unless the byzantine player succeeds in publishing a block which does not include this transaction. First, let us consider the case $n = 1$; the byzantine player succeeds if she publishes the block before any other player. If she fails, her best strategy is to mine blocks on an alternative chain until it gets to be the longest one. If $\alpha \geq \frac{1}{2}$ the byzantine player will almost surely succeed. Otherwise if $\alpha < \frac{1}{2}$, we have that $\Phi_1(\alpha) = \frac{\alpha}{1-\alpha}$. Indeed, we can model the problem with a Markov chain with states $m \in \mathbb{Z} \cap (-\infty, +1]$, in which $+1$ is the only absorbing state, it is possible to move from state n to state $n+1$ with probability α and from state n to state $n-1$ with probability $1-\alpha$. It is a reformulation of the gambler's ruin Markov chain. The state m represents how many blocks the private chain is ahead of the main one. It is enough for the private chain to be one block ahead to succeed.*

In case $n > 1$ we can make a similar argument, but excluding the cases in which the state $+1$ is achieved too early. We have that : $\Phi_n(\alpha) = \frac{\alpha}{1-\alpha} - \sum_{k=1}^{n-1} p \cdot q$ where p is the probability of not achieving $+1$ with $n-k$ blocks left and q is the probability of getting to $+1$ in $2k$ steps. This leads us to the formula : $\Phi_n(\alpha) = \frac{\alpha}{1-\alpha} - \sum_{k=1}^{n-1}(1 - \Phi_{n-k}(\alpha)) \cdot \alpha^k \cdot (1-\alpha)^k \cdot M(k)$, where $M(k)$ is a function defined in [315] that maps natural numbers to the sequence $1, 1, 2, 5, 13, 42 \ldots$. ■

5.4 Robustness to Transacting Users' Behaviors in Layer-2 protocols

In this section we prove the effectiveness of our framework by analyzing the robustness of three different layer-2 protocols. Section 5.4.1 analyzes Lightning Network [3], a protocol on top of the Bitcoin blockchain. Section 5.4.2 addresses the side-chain protocol Platypus [4] while Section 5.4.3 presents an analysis on the cross-chain swap protocol presented [5] and modeled in Chapter 4. Rational and Byzantine behaviors are modeled for blockchain users that in this section (as well as in Chapter 4) are *transacting parties*.

5.4.1 Lightning Network

Bitcoin faces a problem of scalability, in terms of speed, volume and value of the transactions. A transaction is confirmed only once the block to which it belongs is part of a chain with at least D blocks in front of it (under the convention set by the Bitcoin protocol $D = 6$). On average a new block is validated every T minutes (within Bitcoin, $T = 10$), thus it takes around $T \cdot D = 60$ minutes for a transaction to be confirmed, a value that cannot be reduced. Moreover, the number of transactions in a block is limited. Bitcoin cannot bear a sudden upsurge in volume of transactions. Since not all the requests for transactions can be included in a block, some of them are prioritised. The criterion used to order the transactions is the value of the *fee* that a user pays to the mining pool who validates the block. Therefore performing a lot of transactions on the network can be expensive, since a lot of fees have to be paid.

In order to overcome these issues authors in [3] introduce a layer-2 class of protocols called Lightning Network. The latter allows users to create bidirectional payment *channels* to handle unlimited

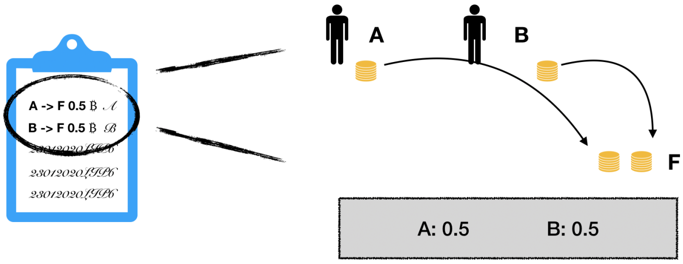


FIGURE 5.3 – A and B open a channel.

transactions in a private manner i.e., off-chain without involving the Bitcoin blockchain. For instance, two users A and B open a channel by publishing on the Bitcoin blockchain two transactions towards a fund F (see Fig 5.3), the amounts of the two transactions constitute the initial balance of the channel. In Section 5.4.1.1 we analyze the protocolar module to open a channel. The fund F can send or receive cryptoassets via blockchain transactions only if both users sign them.

Once the channel is opened, users can exchange by simply privately updating the balance of the channel (see Fig 5.4). The protocol to update the balance is discussed in Section 5.4.1.3.

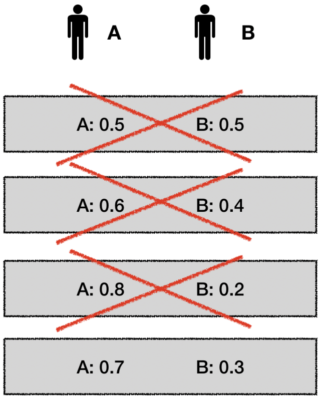


Figure 5.4 – A and B privately update the balance of the channel.

A further construction, called *Hashed Timelock Contract* (HTLC), allows users to create transactions within the channel that can be triggered at will. The structure of the protocol is similar to the one used to update the balance (see Section 5.4.1.4).

When the users are no more interested in exchanging bitcoins they decide to close the channel. Two transactions are published on the Bitcoin blockchain : one from F to A and another one from F to B (see Fig 5.5). The value of the transactions corresponds to the ones of the latest balance. The protocol to close the channel is presented in Section 5.4.1.2.

Lightning Network allows transactions also between users who have not opened a common channel (i.e., *routed payment*). Indeed, two users can perform a transaction through a path of open channels, using other users as intermediate nodes (see Fig 5.6). This protocol is analyzed in Section 5.4.1.5.

In the public Bitcoin blockchain every transaction is signed by the sender. In the Lightning Network every operation is identified by a commitment $\mathcal{C}$ which must be signed by two users, let us say A and B. In the following sections we use the following notations : $\mathcal{C}_{..}$ when the commitment is signed by

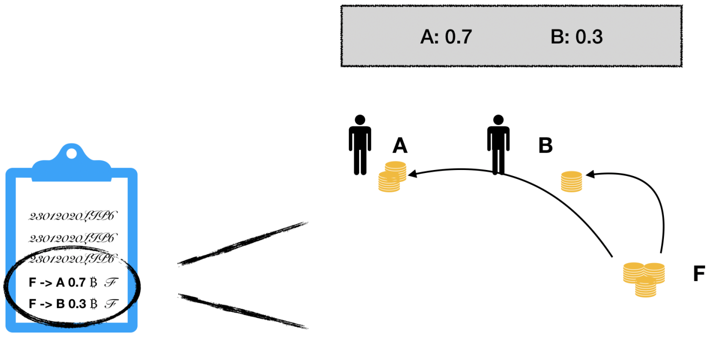


FIGURE 5.5 – A and B close a channel.

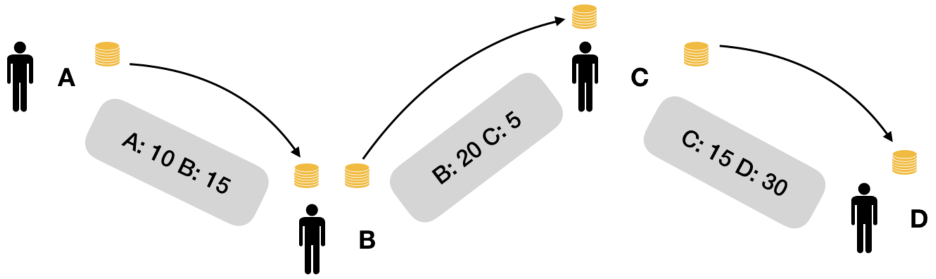


FIGURE 5.6 – A sends 5 ₿ to D through nodes B and C.

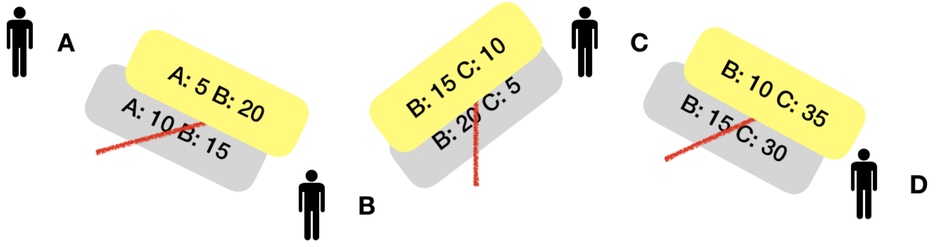


FIGURE 5.7 – All the balances are updated.

nobody ; $\mathcal{C}_{A}$. when the commitment is signed only by user A ; $\mathcal{C}_{\cdot B}$ when the commitment is signed only by user B ; $\mathcal{C}_{AB}$ when the commitment is signed by both users, this is the only case in which the commitment $\mathcal{C}$ is valid.

In practice, the channel consists of a user, let us say F. Every transaction from and to F must be signed by both users A and B.

5.4.1.1 Opening Module

Informally, the protocol asks the users to fund the channel F with two different transactions, respectively valued x_A and x_B, and to create two different commitments that allow them to publish a transaction that makes them close the channel unilaterally. Formally, in order to open a channel the Bitcoin users create a transaction Tx towards F signed by both of them and they create two different commitments that let them close the channel unilaterally. The protocol involves the following steps (see Fig. 5.8) :

1. A creates a transaction $C1b$ that allows F to send x_A to A and x_B to B. B is able to spend x_B only after that Δ blocks are validated (in [3] $\Delta = 1000$). A signs $C1b$ and sends it to B.
2. B creates a transaction $C1a$ that allows F to send x_A to A and x_B to B. A is able to spend x_A only after that Δ blocks are validated. B signs $C1a$ and sends it to B.
3. A creates a transaction Tx that makes A send x_A to F and B send x_B to F. A signs Tx and sends it to B.
4. B signs Tx and publishes it on the Bitcoin blockchain.

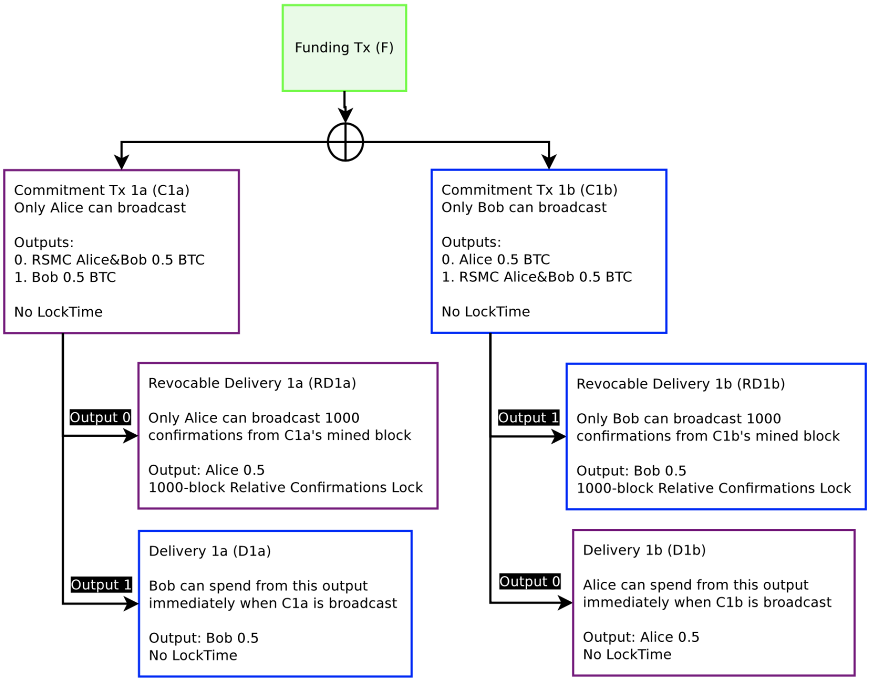

Funding Tx (F)
Commitment Tx 1a (C1a)
Only Alice can broadcast
Outputs:
0. RSMC Alice&Bob 0.5 BTC
1. Bob 0.5 BTC
No LockTime
Commitment Tx 1b (C1b)
Only Bob can broadcast
Outputs:
0. Alice 0.5 BTC
1. RSMC Alice&Bob 0.5 BTC
No LockTime
Output 0
Revocable Delivery 1a (RD1a)
Only Alice can broadcast 1000 confirmations from C1a's mined block
Output: Alice 0.5
1000-block Relative Confirmations Lock
Output 1
Revocable Delivery 1b (RD1b)
Only Bob can broadcast 1000 confirmations from C1b's mined block
Output: Bob 0.5
1000-block Relative Confirmations Lock
Output 1
Delivery 1a (D1a)
Bob can spend from this output immediately when C1a is broadcast
Output: Bob 0.5
No LockTime
Output 0
Delivery 1b (D1b)
Alice can spend from this output immediately when C1b is broadcast
Output: Alice 0.5
No LockTime

FIGURE 5.8 – Scheme of the commitments for the opening of a channel [3].

If a user decides to close the channel unilaterally, she receives her part of funds after a certain interval of time, while the other user receives it immediately. We formalize the protocol with a game in extensive form Γ^{op} (see Definition 5.9), represented by its game tree (see Fig. 5.9). At every node of the tree (i.e., decision step) the player involved in the protocol has two actions available : either following it by signing the commitment required or not following it. The *initial state* corresponds to having no channel opened, while the final state corresponds to having the channel opened. We assign *null* utility to the initial state and positive utility (unitary by convention) to the final state. If at any step the players do not follow the protocol, they get back to the initial state, with outcome $(0, 0)$. If they do follow at every step, they are able to open the channel, with outcome $(1, 1)$. We denote by $\sigma^{op} = (\{C1b_{A\cdot}, Tx_{A\cdot}\}(\{C1a_{\cdot B}, Tx_{AB}\})$ the strategy profile recommended by the protocol.

Definition 5.9 *The opening game Γ^{op} is a game in extensive form, with two players $N = \{A, B\}$ and 4 nodes, labeled by a number (1 is the vertex) :*

1. *A has two actions available : $C1b_{..}$ which provides outcome $(0, 0)$; $C1b_{A\cdot}$ which leads to node 2.*
2. *B has two actions available : $C1a_{..}$ which provides outcome $(0, 0)$; $C1a_{\cdot B}$ which leads to node 3.*
3. *A has two actions available : $Tx_{..}$ which provides outcome $(0, 0)$; $Tx_{A\cdot}$ which leads to node 4.*
4. *B has two actions available : $Tx_{A\cdot}$ which provides outcome $(0, 0)$; Tx_{AB} which provides outcome $(1, 1)$.*

At every node the player involved in the protocol have two actions available : either follow it or not follow it. If at any step they do not follow it, they get back to the initial state, with outcome $(0, 0)$. If they do at every step, they are able to open the channel, with outcome $(0, 0)$. The strategy profile recommended by the protocol is $\sigma^{op} = (\{C1b_{A\cdot}, Tx_{A\cdot}\}, \{C1a_{\cdot B}, Tx_{AB}\})$, in which the actions are played respectively at nodes $(\{1, 3\}, \{2, 4\})$. The protocol is thus represented by the mechanism $(\Gamma^{op}, \sigma^{op})$, whose properties we analyze in the sequel.

Theorem 5.10 *The mechanism $(\Gamma^{op}, \sigma^{op})$ is not immune.*

Proof. *Since we are in a two-player setting, a mechanism is immune (see Definition 5.4) if it is 1-immune, i.e. if both players receive no lower payoff than $u(\sigma^{op}) = (1, 1)$, no matter what the other player chooses. A counterexample is B deviating from $\sigma^{op}_B = \{C1a_{\cdot B}, Tx_{AB}\}$ to $\tau_B = \{C1a_{..}, Tx_{AB}\}$, i.e. B refusing to signing C1a at step 2. For player A the outcome of $u_A(\sigma^{op}_A, \tau_B) = 0 < 1 = u(\sigma^{op})$.*

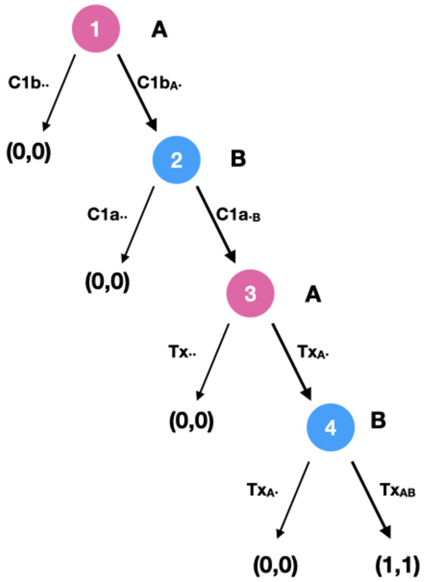


FIGURE 5.9 – The game tree of Γ^{op}

Theorem 5.11 *The mechanism* $(\Gamma^{op}, \sigma^{op})$ *is optimal resilient and weak immune.*

Proof. *The strategy profile* σ^{op} *provides the best outcome for both players* $(1, 1)$. *Therefore, the mechanism* $(\Gamma^{op}, \sigma^{op})$ *is strongly resilient.*
Both σ_A^{op} *and* σ_B^{op} *are dominant strategies respectively for* A *and* B*, because they always get a better outcome, no matter what the other player does. Therefore* σ^{op} *survives after the iterated deletion of weakly dominated strategies : the mechanism is practical. The players never receive negative payoff therefore, if they play* σ_A^{op} *and* σ_B^{op} *they always get a non-negative payoff. This corresponds to the Definition 5.25 of weak immunity.* ■

5.4.1.2 Classical and Alternative Closing Modules

As described in Section 5.4.1.1, both users A and B have a copy of a transaction that allows them to close the channel unilaterally. Indeed, A and B own respectively two commitments $C1a_{.B}$ and $C1b_{A.}$ signed by the other part. If they add their signature, respectively $C1a_{AB}$ and $C1b_{AB}$, they can unilaterally publish a transaction that returns the values stuck in the fund x_A and x_B back to their owners. If a user decides to unilaterally close the channel, she receives her part of the fund after that Δ blocks are validated on the Bitcoin blockchain, while the other user receives it immediately. The protocol recommends to close the channel by creating a new transaction, namely ES, that let the players receive their cryptoassets immediately. We model the situation with the following game in normal form.

Definition 5.10 *The closing game* $\Gamma^{cl} = \langle N, \mathcal{S}, u \rangle$ *of the channel* (x_A, x_B) *with* $x_A, x_B > 0$ *is a game in normal form, with two players* $N = \{A, B\}$ *who have available three different pure strategies each :* $\mathcal{S}_A = \{C1a_{AB}, DN, ES\}$ *and* $\mathcal{S}_B = \{C1b_{AB}, DN, ES\}$*. The value of the utility can be found in the following payoff table.*

			B	
		$C1b_{AB}$	DN	ES
	$C1a_{AB}$	$(\frac{1}{2}, \frac{1}{2})$	$(0, 1)$	$(0, 1)$
A	DN	$(1, 0)$	$(-1, -1)$	$(-1, -1)$
	ES	$(1, 0)$	$(-1, -1)$	$(1, 1)$

First, we assume that the channel (x_A, x_B) is funded by both players i.e., $x_A, x_B > 0$. If one of the two players has no asset involved in the channel, we have to model the situation with a degenerate game, in which she can play any possible strategy. We recommend users to never unilaterally fund the channel. Indeed, if we drop the assumption that both players fund the channel, we have to consider a different modelisation. For instance, if B does not fund the channel we have that $x_B = 0$. No matter what her strategy chooses, she gets nothing. We fix the utility of any outcome to 1 because it corresponds to the outcome of closing the channel. The payoff matrix of the game is the following :

			B	
		$C1b_{AB}$	DN	ES
	$C1a_{AB}$	$(\frac{1}{2}, 1)$	$(0, 1)$	$(0, 1)$
A	DN	$(1, 1)$	$(-1, 1)$	$(-1, 1)$
	ES	$(1, 1)$	$(-1, 1)$	$(1, 1)$

This is a case of degenerate game, in which player B can theoretically choose any possible strategy, even doing nothing DN.

The players have three different strategies : publishing their commitment, seeking a deal to create a new transaction ES or just doing nothing DN. We assign null utility to players who receive their asset after Δ blocks, positive utility (normalized to 1) if they receive it immediately, negative utility if they cannot redeem their cryptoassets. The players receive null payoffs if they get their asset within

Δ blocks, because they return to the initial state. For instance, this is case for player A if the strategy profile chosen by the players is $(C1a_{AB}, ES)$, i.e. if B seeks a deal but A unilaterally closes the channel. The players receive a positive outcome (normalised to 1) if they receive their asset immediately, as for instance if they reach a deal (ES, ES). The players receive a negative outcome (normalised to -1) if their asset is stuck in the channel, such as in the case in which A seeks a deal but B does nothing (DN, ES). In case both users decide to unilaterally close the channel $(C1a_{AB}, C2a_{AB})$, only one between $C1a$ and $C1b$ can be published. They have the same chance $(\frac{1}{2})$ for their transaction to published, leading to any of the state $(0, 1)$ and $(1, 0)$ with equivalent probability. Therefore the utility can be computed as a weighted average : $\frac{1}{2}(0, 1) + \frac{1}{2}(1, 0) = (\frac{1}{2}, \frac{1}{2})$.

The protocol recommends the strategy profile $\sigma^{cl} = (ES, ES)$ i.e., that both players seek a deal. In the following we analyze the properties of the mechanism $(\Gamma^{cl}, \sigma^{cl})$.

Theorem 5.12 *Under the assumption $x_A > 0a$ or $x_B > 0$, the mechanism $(\Gamma^{cl}, \sigma^{cl})$ is optimal resilient, but not weak immune.*

Proof. *The utility $u(\sigma^{cl}) = (1, 1)$ cannot be increased by any other strategy profile, therefore the mechanism $(\Gamma^{cl}, \sigma^{cl})$ is strongly resilient.*
For both player the strategy DN is weakly dominated by the strategy ES. Indeed, no matter what the other player does, the ES always provides the same or even a better utility than DN. If we exclude both strategies DN the players have available only two strategies : $\{C1a_{AB}, ES\}$ and $\{C1b_{AB}, ES\}$. Once again, ES dominates the other strategy by providing a better outcome. The only strategy that survives the iterated deletion of weakly dominated strategies for both players is ES. Therefore the only stable Nash equilibrium is $\sigma^{cl} = (ES, ES)$. Thanks to Proposition 5.2 we can say that a stable equilibrium provides a practical mechanism.
To prove that the mechanism is not weak immune it is sufficient to show a counterexample. Indeed, if A chooses ES as required by the protocol and B chooses the Byzantine strategy DN, player A receives a negative outcome $u_A(\sigma^{cl}_A, DN) = u_A(ES, DN) = -1$. ■

Since the mechanism is not weak immune, it is not immune either. We thus provide an alternative protocol that can satisfy the property of weak immunity.

Theorem 5.13 *Under the assumption $x_A > 0$ or $x_B > 0$, the only weak immune mechanism is (Γ^{cl}, σ^*) with $\sigma^* = (C1a_{AB}, C2a_{AB})$.*

Proof. *In order to identify weak immune mechanisms we apply Proposition 5.4. We consider player A and the game Γ^{cl}_A in which B is the adversarial player whose utility is the opposite of player A's. The payoff matrix of the game Γ^{cl}_A is the following.*

$$\begin{array}{cc|c|c|c|} & & & \boldsymbol{B} & \\ & & C1b_{AB} & DN & ES \\ \hline & C1a_{AB} & (\frac{1}{2},-\frac{1}{2}) & (0,0) & (0,0) \\ \hline \boldsymbol{A} & DN & (1,-1) & (-1,1) & (-1,1) \\ \hline & ES & (1,-1) & (-1,1) & (1,-1) \\ \hline \end{array}$$

The only Nash equilibria of the game in pure strategies is $(C1a_{AB}, DN)$, which provides outcome $(0,0)$. Since this is a zero-sum game, all the Nash equilibria provide the same outcome (v,v) where $v = 0$ is the value of the game. Since the value of the game is non-negative, player A has always a strategy to get at least 0. This strategy is $C1a_{AB}$, which thus is the only one that player A can choose in a weak immune mechanism.
Analogously we can define the game Γ^{cl}_B in which A is the adversarial player, which lets us prove that $C1b_{AB}$ is the only weak immune strategy for player B. Therefore, $(C1a_{AB}, C1b_{AB})$ is the only strategy profile that provides a weak immune mechanism. ■

We believe that Lightning Network should include the alternative protocol (Γ^{cl}, σ^*) as default. In the case in which the channel is unilaterally funded, one of the player is already forced to follow the mechanism (Γ^{cl}, σ^*). Listing all the possible strategies we have determined the only protocol which can be modeled as a weak immune mechanism. It is not possible to create any other protocol that can satisfy this property.

5.4.1.3 Updating Module

Performing a transaction within a channel consists in updating its balance. Technically, the previous commitments ($C1a$ and $C1b$) with balance (x_A, x_B) are replaced by two new commitments ($C2a$ and $C2b$) with different balance (x'_A, x'_B). In order to prevent players from publishing old commitments, they sign two Breach Remedy Transactions ($BR1a$ and $BR1b$), that can invalidate $C1a$ and $C2b$. Indeed, if any party publishes an outdated commitment the other one can retrieve all the cryptoassets in the fund. If, for instance, user A publishes the outdated commitment $C1a$, she can retrieve her fund x_A unless user B publishes $BR1a$ before Δ blocks are validated. Briefly speaking, if any part publishes an outdated commitment the other part can retrieve all the assets in the fund. In practice the players have an incentive to delete outdated commitments to limit the risk of an unintentional leak, that could provoke their publication and thus the loss of all the assets stored in the channel. The protocol to update the balance (see Fig. 5.10) requires the players to sign the commitments in a specific order. The protocol involves the following steps :

1. A creates a transaction *C2b* that allows F to send x'_A to A and to send x'_B to B. B is able to spend x'_B only after that Δ blocks are validated. A signs *C2b* and sends it to B.
2. B creates a transaction *C2a* that allows F to send x'_A to A and to send x'_B to B. A is able to spend x'_A only after that Δ blocks are validated. B signs *C2a* and sends it to B.
3. A creates a transaction $BR1a$ that lets B retrieve x_A in case A publishes *C1a* and B publishes $BR1a$ within the following Δ blocks. Then A sends $BR1a$ to B.
4. B creates a transaction $BR1b$ that lets A retrieve x_B in case B publishes *C1b* and A publishes $BR1b$ within the following Δ blocks. Then B sends $BR1b$ to A.

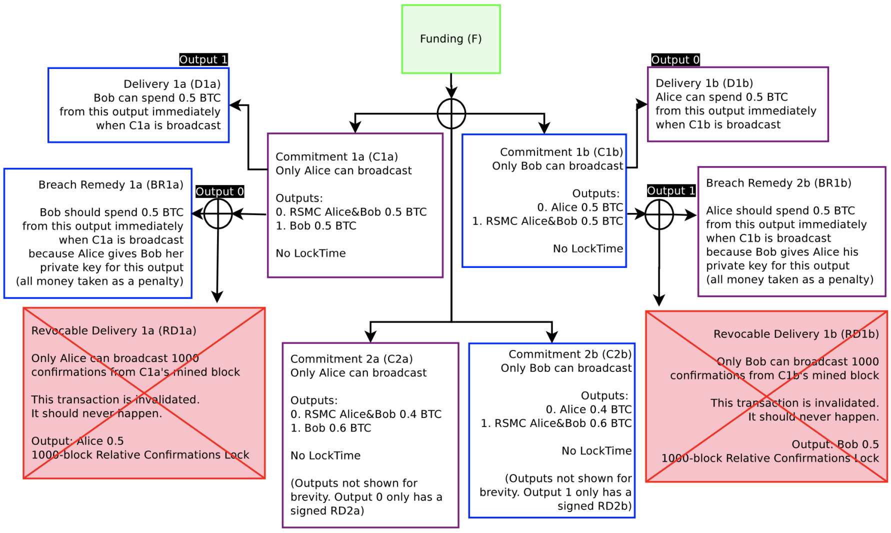

Funding (F)
Output 1
Delivery 1a (D1a)
Bob can spend 0.5 BTC
from this output immediately
when C1a is broadcast
Output 0
Delivery 1b (D1b)
Alice can spend 0.5 BTC
from this output immediately
when C1b is broadcast
Commitment 1a (C1a)
Only Alice can broadcast
Outputs:
0. RSMC Alice&Bob 0.5 BTC
1. Bob 0.5 BTC
No LockTime
Commitment 1b (C1b)
Only Bob can broadcast
Outputs:
0. Alice 0.5 BTC
1. RSMC Alice&Bob 0.5 BTC
No LockTime
Breach Remedy 1a (BR1a)
Bob should spend 0.5 BTC
from this output immediately
when C1a is broadcast
because Alice gives Bob her
private key for this output
(all money taken as a penalty)
Output 0
Output 1
Breach Remedy 2b (BR1b)
Alice should spend 0.5 BTC
from this output immediately
when C1b is broadcast
because Bob gives Alice his
private key for this output
(all money taken as a penalty)
Revocable Delivery 1a (RD1a)
Only Alice can broadcast 1000
confirmations from C1a's mined block
This transaction is invalidated.
It should never happen.
Output: Alice 0.5
1000-block Relative Confirmations Lock
Commitment 2a (C2a)
Only Alice can broadcast
Outputs:
0. RSMC Alice&Bob 0.4 BTC
1. Bob 0.6 BTC
No LockTime
(Outputs not shown for
brevity. Output 0 only has a
signed RD2a)
Commitment 2b (C2b)
Only Bob can broadcast
Outputs:
0. Alice 0.4 BTC
1. RSMC Alice&Bob 0.6 BTC
No LockTime
(Outputs not shown for
brevity. Output 1 only has a
signed RD2b)
Revocable Delivery 1b (RD1b)
Only Bob can broadcast 1000
confirmations from C1b's mined block
This transaction is invalidated.
It should never happen.
Output: Bob 0.5
1000-block Relative Confirmations Lock

FIGURE 5.10 – Scheme of the commitments to update the balance of the channel [3].

We formalize the protocol with a game in extensive form Γ^{up} (see Definition 5.11), represented by the tree in Fig. 5.11. The initial state corresponds to the previous balance (with thus null utility), the final state to the updated balance (with utility equal to 1). One may question that with the updated balance one of the two party is receiving a smaller cryptoasset however, this does not consist in receiving a lower utility since updating the balance guarantees the exchange of a different cryptoasset which is more valuable than the one stored in the channel. We assign a negative value to the states in which players lose their cryptoassets or part of them.

Definition 5.11 *The updating game Γ^{up} is a game in extensive form, with two players $N = \{A, B\}$ and 5 nodes, labeled by a number (1 is the vertex) :*

1. *A has two actions available : $C2b_{..}$ which provides outcome $(0,0)$; $C2b_{A.}$ which leads to node 2.*
2. *B has three actions available : $C2a_{..}$ which provides outcome $(0,0)$; $C2b_{AB}$ which provides outcome $(1,1)$; $C2a_{.B}$ which leads to node 3.*
3. *A has three actions available : $BR1a_{..}$ which provides outcome $(0,0)$; $C2a_{AB}$ which provides outcome $(1,1)$; $BR1a_{A.}$ which leads to node 4.*
4. *B has two actions available : $BR1b_{.B}$ which provides outcome $(1,1)$; $BR1b_{..}$ leading to node 5.*
5. *A has two actions available : $C1a_{AB}$ which provides outcome $(-1,1)$; $C2a_{AB}$ which provides outcome $(1,1)$.*

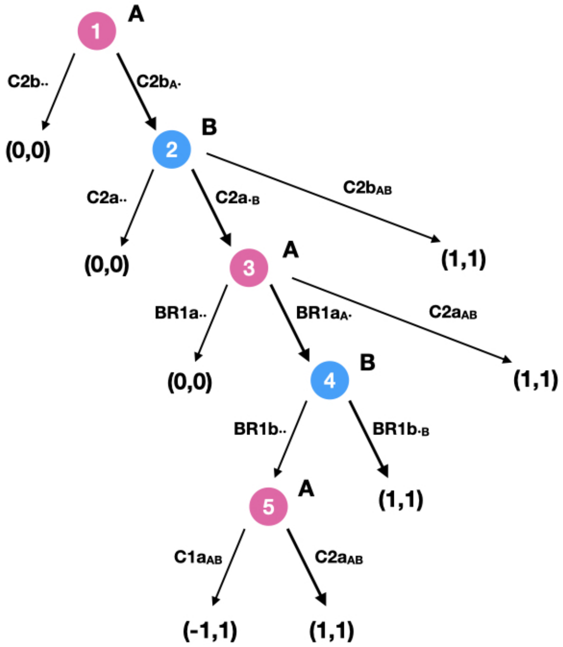


FIGURE 5.11 – The game tree of Γ^{up}

The protocol recommends to sign all the commitments and it is indeed represented by the strategy profile $\sigma^{up} = (\{C2b_{A.}, BR1a_{A.}, C2a_{AB}\}, \{C2a_{.B}, BR1b_{.B}\})$ in which actions are played respectively at nodes $(\{1,3,5\},\{2,4\})$. At nodes 2 and 3 respectively users B and A can enforce the new commitments by publishing them on the Bitcoin blockchain and thus closing the channel. At node 4, user B can refuse to provide the breach remedy transaction to user A, who at node 5 can then publish the new commitment enforcing the closure of the channel. If at node 5 user A publishes the old commitment $C1a$, user B can retrieve all the funds by publishing the breach remedy transaction $BR1a$.

We analyze the properties of the mechanism $(\Gamma^{up}, \sigma^{up})$ under the assumption that it is always possible to publish a transaction within Δ blocks, otherwise it is not possible to validate the breach remedy transactions in time. The mechanism is not immune, indeed if any user refuses to sign a commitment the players return to the original balance that provides lower payoff than the final balance. However, the mechanism satisfies the properties of optimal resilience and weak immunity.

Theorem 5.14 *The mechanism $(\Gamma^{up}, \sigma^{up})$ is not immune.*

Proof. *Since we are considering a game with only two players, a mechanism is immune if it is 1-immune. A mechanism is 1-immune (see Definition 5.4) if any player receives the same outcome by playing the recommended strategy, no matter which strategy the other player chooses. This is not the case of the mechanism $(\Gamma^{up}, \sigma^{up})$, indeed if player A chooses σ_A^{up} and player B chooses $\{C2a.., BR1b._B\} \neq \sigma_B^{up}$ the payoff for player A is $u_A(\sigma_A^{up}, \{C2a.., BR1b._B\}) = 0 < 1 = u_A(\sigma_A^{up}, \sigma_B^{up})$.* ■

The property of immunity is too strong in this case, therefore we consider other weaker properties.

Theorem 5.15 *The mechanism $(\Gamma^{up}, \sigma^{up})$ is optimal resilient and weak immune with probability $1 - \Phi_\Delta(\alpha)$, but it is not immune.*

Proof. *We analyze the mechanism $(\Gamma^{up}, \sigma^{up})$ under the assumption that it is always possible to publish a transaction within Δ blocks, otherwise it is not possible to validate the breach remedy transactions in time. The probability that this happens when a byzantine agent with computational power α attacks the Bitcoin blockchain is $1 - \Phi_\Delta(\alpha)$ (see Theorem 5.9).*
The outcome for the strategy profile σ^{up} is $(1,1)$, which cannot be increased by any other strategy profile. Therefore, the mechanism $(\Gamma^{up}, \sigma^{up})$ is strongly resilient.
In order to prove that the mechanism is resilient, we have to exclude weakly dominated strategies. We proceed by excluding all the actions that are included in a weakly dominated strategy. At node

1, *A receives always a better outcome by picking action* $C2b_{A\cdot}$ *rather than* $C2b_{\cdot\cdot}$, *thus* $C2b_{\cdot\cdot}$ *is never included in a practical mechanism. At node* 2, *B never plays the action* $C2a_{\cdot\cdot}$, *at node* 3 *A never plays* $BR1a_{\cdot\cdot}$ *and at node* 5 *A never plays* $C1a_{AB}$. *The remaining strategy profiles, included* σ^{up}, *provide outcome* $(1,1)$. *Since they all survive the iterated deletion of weakly dominated strategies, they are all practical mechanisms. Thanks to Corollary 5.0.1 we know that there always exists at least one practical mechanism. However, the reader should keep in mind that this might not be unique.*

In order to prove that the mechanism is weak immune we apply Proposition 5.4. We consider one player i at a time and we make the other player j adversarial, by fixing her outcome as the opposite of player i (see Fig. 5.12). Then we prove that the best response of player j to player i never leads her to a negative outcome. We take $i = A$ *and we consider the game* Γ_A^{up} *in which player* $j = B$ *has utility opposite to player* i. *The best response of player* j *to the strategy* σ_A^{up} *picked by player* i *is the strategy* $\{C2a_{\cdot\cdot}, BR1b_{\cdot\cdot}\}$, *i.e. at node* 2 *to avoid to reach a deal by not signing* $C2a$. *The payoff for player A is* $u_A(\sigma_A^{up}, \{C2a_{\cdot\cdot}, BR1b_{\cdot\cdot}\}) = 0$, *which is non-negative. Analogously we have the same result for player* B. *Since both adversarial games provide non-negative payoff, thanks to Proposition 5.4 we get that the mechanism is weak immune.* ■

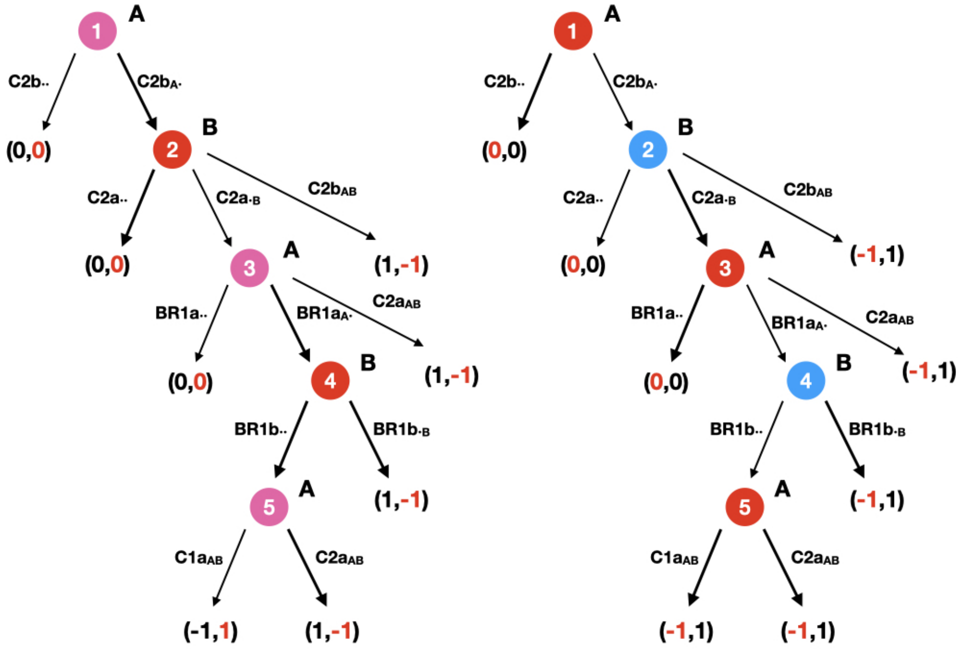


FIGURE 5.12 – The game trees of Γ_A^{up} and Γ_B^{up}

5.4.1.4 Hash-Time Locked Contract Module

A bidirectional payment channel only allows transactions inside a channel. In order to perform transactions through a network of channels Lightning Network introduces an additional construction, called Hash-Time Locked Contract (HTLC). The HTLC allows to create transactions that can be triggered at will. The HTLC makes use of the *hash function*, a deterministic caotic function that maps any input x to a fixed-length string $y = hash(x)$. It is not possible to retrieve x given y in a faster way than trying with a bruce-force method to randomly guess x. Hence if x is chosen among strings of considerable length, it is almost impossible to identify x given by $y = hash(x)$ in a reasonable time. Let us suppose that users A and B open a channel with balance (x_A, x_B) and A wants to send a payment through HTLC to B so that the new balance would be (x'_A, x'_B), with $x_A < x'_A$. A creates a random data R and then computes $H = hash(R)$. Then she sends an update of the contract to B, with a specific characteristic : if B publishes it, she can retrieve the difference $x'_B - x_B$ only if she proves to know x such that $H = hash(x)$ within Δ blocks (in [3] $\Delta = 1000$). A can trigger the contract by providing R to B. If she does not do it, B cannot find $x = R$ and thus has no incentive to publish the contract. The HTLC protocol works as follows (see Fig. 5.13) :

1. A creates a commitment *C2b* that allows F to send x'_A to A, x_B to B after Δ blocks and $x'_B - x_B$ to B if she publishes x such that $H = hash(x)$ to the Bitcoin blockchain within Δ blocks. A signs it and sends it to B.
2. Analogously, B creates a set of commitment *C2a* that allows F to send x'_B to B, x_A to A after Δ blocks and $x'_B - x_B$ to B if she publishes x such that $H = hash(x)$ to the Bitcoin blockchain within Δ blocks. B signs it and sends it to A.
3. A creates a transaction *BR1a* that lets B retrieve x_A in case A publishes *C1a* and B publishes *BR1a* within the following Δ blocks. Then A sends *BR1a* to B.
4. B creates a transaction *BR1b* that lets A retrieve x_B in case B publishes *C1b* and A publishes *BR1b* within the following Δ blocks. Then B sends *BR1b* to A.

The protocol for the HTLC corresponds to the protocol for updating a channel, with the only difference that the new commitments *C2a* and *C2b* provide a different output. Under the assumption that a transaction (or just the key R) can be published within Δ blocks, we can define a game Γ^{htlc} with the very same structure as Γ^{up} (see Definition 5.11 and Fig. 5.11). Following the protocol corresponds to

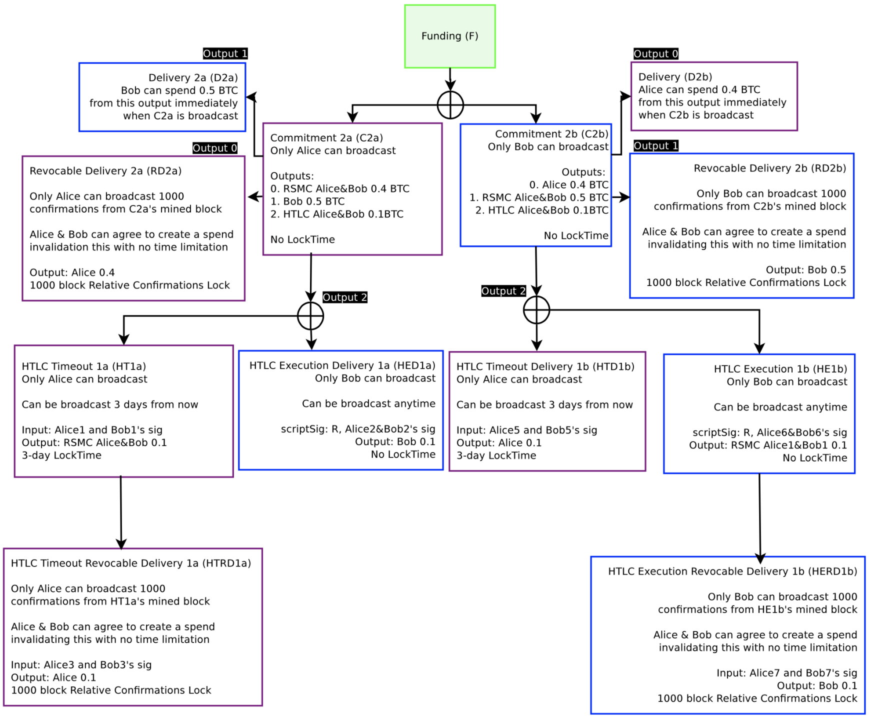

Funding (F)
Output 1
Delivery 2a (D2a)
Bob can spend 0.5 BTC
from this output immediately
when C2a is broadcast
Output 0
Delivery (D2b)
Alice can spend 0.4 BTC
from this output immediately
when C2b is broadcast
Commitment 2a (C2a)
Only Alice can broadcast
Outputs:
0. RSMC Alice&Bob 0.4 BTC
1. Bob 0.5 BTC
2. HTLC Alice&Bob 0.1BTC
No LockTime
Commitment 2b (C2b)
Only Bob can broadcast
Outputs:
0. Alice 0.4 BTC
1. RSMC Alice&Bob 0.5 BTC
2. HTLC Alice&Bob 0.1BTC
No LockTime
Output 0
Revocable Delivery 2a (RD2a)
Only Alice can broadcast 1000
confirmations from C2a's mined block
Alice & Bob can agree to create a spend
invalidation this with no time limitation
Output: Alice 0.4
1000 block Relative Confirmations Lock
Output 1
Revocable Delivery 2b (RD2b)
Only Bob can broadcast 1000
confirmations from C2b's mined block
Alice & Bob can agree to create a spend
invalidating this with no time limitation
Output: Bob 0.5
1000 block Relative Confirmations Lock
Output 2
Output 2
HTLC Timeout 1a (HT1a)
Only Alice can broadcast
Can be broadcast 3 days from now
Input: Alice1 and Bob1's sig
Output: RSMC Alice&Bob 0.1
3-day LockTime
HTLC Execution Delivery 1a (HED1a)
Only Bob can broadcast
Can be broadcast anytime
scriptSig: R, Alice2&Bob2's sig
Output: Bob 0.1
No LockTime
HTLC Timeout Delivery 1b (HTD1b)
Only Alice can broadcast
Can be broadcast 3 days from now
Input: Alice5 and Bob5's sig
Output: Alice 0.1
3-day LockTime
HTLC Execution 1b (HE1b)
Only Bob can broadcast
Can be broadcast anytime
scriptSig: R, Alice6&Bob6's sig
Output: RSMC Alice1&Bob1 0.1
No LockTime
HTLC Timeout Revocable Delivery 1a (HTRD1a)
Only Alice can broadcast 1000
confirmations from HT1a's mined block
Alice & Bob can agree to create a spend
invalidating this with no time limitation
Input: Alice3 and Bob3's sig
Output: Alice 0.1
1000 block Relative Confirmations Lock
HTLC Execution Revocable Delivery 1b (HERD1b)
Only Bob can broadcast 1000
confirmations from HE1b's mined block
Alice & Bob can agree to create a spend
invalidating this with no time limitation
Input: Alice7 and Bob7's sig
Output: Bob 0.1
1000 block Relative Confirmations Lock

FIGURE 5.13 – Scheme of the commitments of the HTLC [3].

the strategy profile σ^{htlc}. Hence we can introduce the following theorem.

Theorem 5.16 *The mechanism* $(\Gamma^{htlc}, \sigma^{htlc})$ *is optimal resilient and weak immune, but not immune, with probability* $1 - \Phi_{\Delta}(\alpha)$.

Proof. *Since the mechanisms* $(\Gamma^{htlc}, \sigma^{htlc})$ *and* $(\Gamma^{up}, \sigma^{up})$ *follow the very same structure, we can apply Theorem 5.15.* ■

5.4.1.5 Routing Module

The *Hashtime Locked Contract* (HTLC) allows to create transactions that can be triggered at will. Summing up what presented in Section 5.4.1.4 for technical details, the protocol for the HTLC works as follows. User A creates a pair (H, R), where H is public and R is its private key. She shares with user B a commitment together with the string H. Once this commitment is published on the Bitcoin blockchain, user B can receive the transaction only if she can provide the private key R within Δ blocks. It is easy to check that R is the private key of H, but it is almost impossible to retrieve R, given H. In this way, user A can trigger the transaction whenever she wants by disclosing R to user

B. The protocol is represented by the mechanism $(\Gamma^{htlc}, \sigma^{htlc})$, that has the very same structure of the updating module (see Section 5.4.1.3) and thus satisfies optimal resilience and weak immunity, but not immunity.

Lightning Network allows payments also between two users, namely A and C, who do not share a channel. The requirement for a *routed payment* is to find a path of channels between the two users, i.e. a sequence of users who two-by-two share a channel. For instance, let us suppose that users A and C have both opened a separate channel with a third user B. In the *routed payment* user B is the intermediate node. The HTLC is implicated in the protocol that allows users to perform routed payments, which works as follows. Let us consider the case of a single intermediate node, namely B : users A and B have an opened channel with balance (x_A, x_B), while B and C have opened a different channel with balance (y_B, y_C). Let us suppose that A wishes to send δ to C. Informally, A sends $\delta + \epsilon$ to B and B sends δ to C, where $\epsilon \geq 0$ is the fee given to the intermediate node B. Since the channel are opened the two payments consists in updating the balance of the two channels : $(x_A, x_B) \to (x_A - \delta - \epsilon, x_B + \delta + \epsilon)$ and $(y_B, y_C) \to (y_B - \delta, y_C + \delta)$. The protocol for routed payments lets the receiver C trigger both payments at the same moment :

1. C creates a random data R and hashes it : $H = hash(R)$. Then, she sends H to A.
2. A creates a HTLC, namely H^{AB} of value $\delta + \epsilon$ locked with H and sends it to B.
3. B creates a HTLC, namely H^{BC} of value δ locked with H and sends it to C.
4. C discloses R to B, hence validating H^{BC}.
5. B discloses R to A, thus validating H^{AB}.

We formalize the protocol with a game in extensive form Γ^{rout}, whose tree is displayed in Fig. 5.14. The initial state consists in the initial balance and it is assigned null utility. The final state corresponds for A and C to fulfill the payment, for B to receive the fee ϵ. The final state has positive payoff, normalised to 1. Any state that consists in a loss of assets is assigned negative payoff. The strategy profile recommended by the protocol is denoted by $\sigma^{rout} = (\{H_A^{AB}\}, \{H_B^{BC}, Y\}, \{Y, Y\})$.

Definition 5.12 *The routing game* Γ^{rout} *is a game in extensive form, with three players* $N = \{A, B, C\}$ *and 5 nodes, labeled by a number (*1 *is the vertex) :*

1. *C has two actions available : either* N*, not sending* H *to A, which provides outcome* $(0,0,0)$*, or* Y*, sending* H *to A, which leads to node* 2*.*

2. *A has two actions available : either $H_{\cdot}^{AB}$, which provides outcome $(0,0,0)$, or H_{A}^{AB}, which leads to node 3.*
3. *B has two actions available : either $H_{\cdot}^{BC}$, which provides outcome $(0,0,0)$, or H_{B}^{BC}, which leads to node 4.*
4. *C has two actions available : either N, not disclosing R to B, which provides outcome $(0,0,0)$, or Y, disclosing R to B, which leads to node 5.*
5. *B has two actions available : either N, not disclosing R to A, which provides outcome $(1,-1,1)$ or Y, disclosing R to A, which provides outcome $(1,1,1)$.*

At node 1 C creates the lock H and its key R. At node 2 and 3 the two HTLCs are created. At node 4 C triggers the payment in the channel that she shares with B. At node 5 B triggers the payment in the channel that she shares with A. If at step 5 B does not trigger the payment, A and C reach the final state, because C has received the payment, also if A has not paid for it.
The recommended strategy profile is $\sigma^{rout} = (\{H_{A}^{AB}\}, \{H_{B}^{BC}, Y\}, \{Y, Y\})$, respectively played at nodes $(\{2\}, \{3, 5\}, \{1, 4\})$. The payoff are as shown only under the assumption that in both HTLCs the

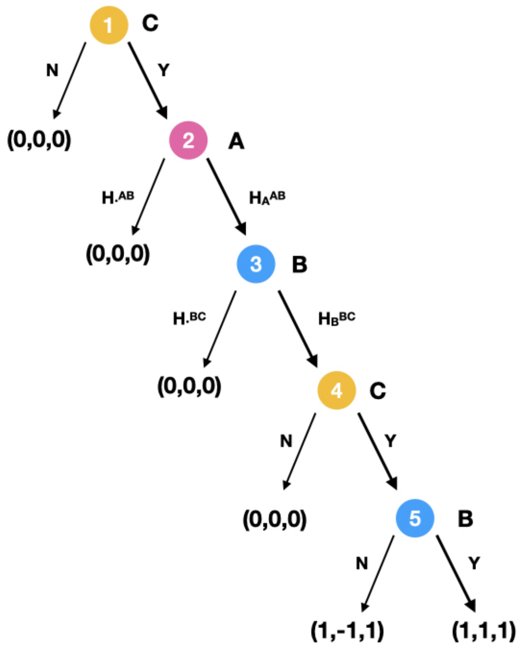


FIGURE 5.14 – The game tree of Γ^{rout}

transactions can be triggered. We analyze the protocol under this assumption.

The following theorems state that the mechanism corresponding to the routed payment protocol is not immune but is weak immune and optimal resilient.

Theorem 5.17 *$(\Gamma^{rout}, \sigma^{rout})$ is not immune.*

Proof. *Since the game Γ^{rout} has three players, the mechanism is immune if it is 1-immune and 2-immune. To prove that the mechanism is not immune, it is enough to prove that it is not 1-immune. A mechanism is 1-immune (see Definition 5.4) if any player who chooses the recommended strategy receives the same outcome, no matter what any Byzantine player can choose. This property is not fulfilled. Indeed, if A picks the strategy $H^{AB}_{\cdot}$, the outcome for C is lower : $u_C(H^{AB}_{\cdot}, \sigma_B^{rout}, \sigma_C^{rout}) = 0 < 1 = u_C(\sigma_A^{rout}, \sigma_B^{rout}, \sigma_C^{rout}) = u_C(\sigma^{rout})$.* ■

The property of immunity is too strong for this protocol, therefore we consider the other properties.

Theorem 5.18 *Under the assumption that in both HTLCs the transactions can be triggered, $(\Gamma^{rout}, \sigma^{rout})$ is optimal resilient and weak immune.*

Proof. *There is no other strategy than σ^{rout} that can improve any of its payoffs $u(\sigma^{rout}) = (1, 1, 1)$. Thus $(\Gamma^{rout}, \sigma^{rout})$ is a strongly resilient mechanism.*

In order to prove that the mechanism is practical, we proceed by excluding the actions that belongs to weakly dominated strategies. At node 5 B never plays N because she would receive -1 rather than 1. Therefore at node 4 C never chooses N because she would receive 0 rather than 1. Analogously at nodes 3, 2 and 1 players do not choose alternative actions, because they would receive 0 rather than 1. The strategy profile σ^{rout} is the only one that survives the iterated deletion of weakly dominated strategies, hence the mechanism is practical.

In order to prove that the mechanism is weak immune we apply Proposition 5.4. We consider one player i at a time and we introduce an adversarial player j that plays at any node which is not played by i (see Fig. 5.15). We define the game Γ_i^{rout} which has the same structure, two players i and j and utility function for j opposite to the one of player i. In games Γ_A^{rout} and Γ_C^{rout} respectively A and C never receive negative payoffs. In game Γ_B^{rout} player B never receives negative payoff if she plays σ_B^{rout}. For Proposition 5.4, since all the adversarial games Γ_i^{rout} do not provide negative payoff if the players follow the recommended strategy σ_i^{rout}, the mechanism is weak immune. ■

The HTLCs introduced in the protocol work independently from the routing protocol. We can model them with two different mechanisms : $(\Gamma^{AB}, \sigma^{AB})$ for H^{AB} and $(\Gamma^{BC}, \sigma^{BC})$ for H^{BC}. The mechanism $(\Gamma^{AB}, \sigma^{AB})$ represents the HTLC deployed on the channel A-B, while the mechanism $(\Gamma^{BC}, \sigma^{BC})$ refers to the HTLC implemented on the channel B-C. The HTLCs belong to two different channels, so they are independent one from another. The assumption from the routing protocol is that in both HTLCs the transactions can be triggered, but this is true only if every transaction can be published within Δ blocks (see Section 5.4.1.4). Under this assumption, the protocol for routed payments is independent from the protocol for HTLC, because it is external with respect to the channel, while the HTLCs work within the channel. The routed payment is thus represented by three independent protocols $(\Gamma^{rout}, \sigma^{rout})$, $(\Gamma^{AB}, \sigma^{AB})$, and $(\Gamma^{BC}, \sigma^{BC})$. Therefore we analyze the properties of its mechanism by defining the composition of the three games $(\Gamma^{rout} \odot \Gamma^{AB} \odot \Gamma^{BC}, \{\sigma_i^{rout}, \sigma_i^{AB}, \sigma_i^{BC}\})$.

Theorem 5.19 *The mechanism* $(\Gamma^{rout} \odot \Gamma^{AB} \odot \Gamma^{BC}, \{\sigma_i^{rout}, \sigma_i^{AB}, \sigma_i^{BC}\})$ *is optimal resilient and weak immune with probability* $1 - \Phi_\Delta(\alpha)$.

Proof. *We analyze the mechanism* $(\Gamma^{up}, \sigma^{up})$ *under the assumption that it is always possible to publish a transaction within* Δ *blocks, otherwise it is not possible to validate the breach remedy transactions in time. The probability that this happens when a byzantine agent with computational power* α *attacks the Bitcoin blockchain is* $1 - \Phi_\Delta(\alpha)$ *(see Theorem 5.9). The operator composition (see Definition 5.6) is invariant with respect the properties of the mechanisms. Thanks to Theorems 5.16 and 5.18 we have that* $(\Gamma^{rout}, \sigma^{rout})$, $(\Gamma^{AB}, \sigma^{AB})$ *and* $(\Gamma^{BC}, \sigma^{BC})$ *are practical. Therefore, with Theorem 5.2 we have that their composition* $(\Gamma^{rout} \odot \Gamma^{AB} \odot \Gamma^{BC}, \{\sigma_i^{rout}, \sigma_i^{AB}, \sigma_i^{BC}\})$ *is practical.*
Analogously, thanks to Theorems 5.16 and 5.18 we have that every single mechanism is k-resilient for all k and t-weak-immune for all t. Theorems 5.3 and 5.4 allow us to say that the composition $(\Gamma^{rout} \odot \Gamma^{AB} \odot \Gamma^{BC}, \{\sigma_i^{rout}, \sigma_i^{AB}, \sigma_i^{BC}\})$ *is k-resilient for all k and t-weak-immune for all t i.e., it is strongly resilient and weak immune.* ■

Recap. All the results of Lightning Network protocol are available in Table 5.2. The protocol is built on top of the Bitcoin blockchain therefore, its properties depend on the Bitcoin's ones. If we exclude the closing protocol, the Lightning Network satisfies optimal resilience and weak immunity. Hence, we can compose (see Definition 5.6) its protocols' games with Bitcoin mechanism's, which provide weaker results, and prove that the Lightning Network satisfies the same properties of the Bitcoin mechanism.

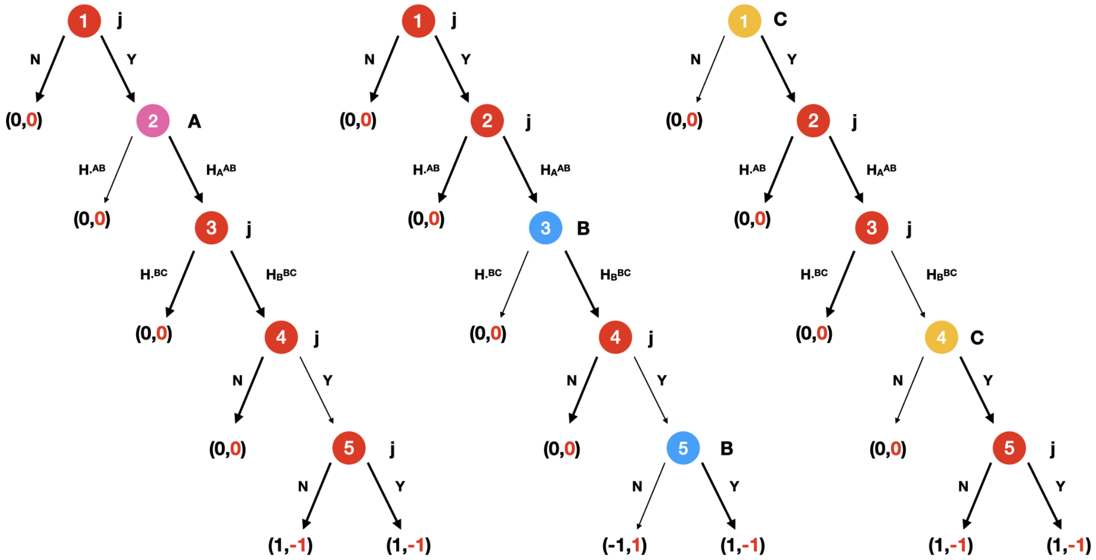


FIGURE 5.15 – The game trees of Γ_A^{rout}, Γ_B^{rout} and Γ_C^{rout}

5.4.2 Side-Chain

A different solution to overcome the scalability and privacy problems of blockchains is offered by Platypus [4], a protocol that allows a group of users to create a childchain (sidechain) that can handle off chain transactions without the need of synchrony among peers. In this section we consider the protocol to create a Platypus chain, described in Fig. 5.16. The protocol let the childchain validators broadcast transactions to the peers until the number of validators that have confirmed the transactions overcome a defined threshold.

It is possible to model this protocol with a game in extensive form Γ^{cr}, in which players are split into two categories : normal users (set U) and the validators (set V). Users' utility is positive if their transactions are successfully published and it is negative if a different wrong transaction is validated instead of hers. Normal users have utility 1 if their transaction is successfully published, 0 if they get back to the initial state, -1 if they lose anything in the process. The validators have utility n, with n the number of valid transactions which are broadcast. The protocol is divided into phases. Every phase consists of players acting at the same time, indeed we work under the assumption that the broadcast of any of the players involved is subsequent to the action of every other player. If this condition is not fulfilled, it would be necessary to consider different phases instead of one, with the same structure.

Definition 5.13 *The creation game is a game Γ^{cr} in extensive form, where $N = U \cup V$ is the set of players, with $|N| = m_v$. Every phase corresponds to a node of the tree, at which players play at the same time.*

- *Phase 1 ; only the player p_0 is involved. The player p_0 has two actions : either complete it Y or not N. If she does not, the outcome is 0 for all players.*
- *Phase 2 ; every player within normal users play at the same time. Everyone dispose of the same two actions : broadcasting their message Y or not N. If the message is not broadcast for player i, her utility is always 0.*
- *Phase 3 ; the validators can choose within a set of actions a_u with $u \subseteq U$ i.e., they can validate all the messages for the users within the set u. The cardinality of the set of their actions is equal to $2^{|U|}$. The utility for the validators corresponds to the number of valid transactions which are broadcast.*
- *Phase 4 ; the validators can choose within a set of actions in the form $(b_t, s_{t'})$, where t and t' are any subset of transactions broadcast in Phase 3. The action b consists in broadcasting the transactions belonging to the set t until $\lfloor 2m_v/3 \rfloor + 1$ validators receive it, while s means to send the transactions in t'.*

We define the mechanism $(\Gamma^{cr}, \sigma^{cr})$, where $\sigma^{cr} \in \mathcal{S}$ is the strategy of following the protocol i.e., for normal users u the strategy is $\sigma^{cr}_u = Y$, while for validators v the strategy is $\sigma^{cr}_v = (a_{u^*}, b_{t^*}, s_{t^*})$, where u^* is the set of users who send a message and t^* is the set of transactions broadcast in Phase 3. We thus analyze the properties of the mechanism.

Theorem 5.20 *The mechanism $(\Gamma^{cr}, \sigma^{cr})$ is not t-immune for any t.*

Proof. *It is enough to prove that the mechanism is not 1-immune. A mechanism is 1-immune if every player does not reduce her utility if only one other player is choosing a Byzantine behavior (see Definition 5.4). This property is not fulfilled, indeed if in Phase 1 the process p_0 chooses N rather than $\sigma^{cr}_{p_0} = Y$, the utility for every player is 0, which is lower than the utility provided by σ^{cr}.* ∎

In [4] it is proved that no wrong transaction can be validated if there are at most $\lfloor \frac{m_v}{3} \rfloor$ corrupted players. This property cannot be expressed with the concept of immunity, which is too strong ; to capture this information we exploit the definition of t-weak-immunity (see Definition 5.25). Within

Algorithm 1 Platypus creation procedure

```
▷ State of the algorithm
$\Omega$, the parentchain
$\Gamma$, the Platypus protocol
$P_\Omega$, the set of processes in the parentchain
$P_\Psi \leftarrow \perp$, the set of processes in the Platypus chain
$V_\Psi \leftarrow \perp$, the set of validators in the Platypus chain
$m_v$, the amount of validators required in $\Psi$
$\mathbb{C}_i$, coins that belong to process $p_i$
$job_i$, boolean defining if $p_i$ is VALIDATOR or just USER
$plid$, the Platypus chain identifier
$msg_i = \langle \mathbb{C}_i, plid, job_i\, \sigma_i \rangle$, signed message to join.
$\sigma_i$, signature of $msg_i$ by $p_i$
$tx_{plcr} \leftarrow \perp$, the Platypus creation transaction

▷ PHASE 1: process $p_0$ initiates request
$msg_0 \leftarrow \mathsf{sign}(\langle \mathbb{C}_0, plid, \mathsf{job}_0 \rangle)$
$\mathsf{multicast}(msg_0)$ to $P_\Omega$

▷ PHASE 2: Rest of processes who want to join reply
when $msg_0$ is received from $p_0$
$msg_i \leftarrow \mathsf{sign}(\langle \mathbb{C}_\mathsf{i}, plid, \mathsf{job}_\mathsf{i} \rangle)$
$\mathsf{multicast}(msg_i)$ to $P_\Omega$

▷ PHASE 3: Validator $p_i \in V_\Psi$ gathers enough validators
when $msg_j$ is received from $p_j$ and $p_j \notin P_\Psi$
$\{P_\Psi, \mathbb{C}_{P_\Psi}\} \leftarrow \{P_\Psi \cup \{p_j\},\ \mathbb{C}_{P_\Psi} \cup msg_j.\mathbb{C}_j\}$
if $(msg_j.job_j = \mathsf{VALIDATOR}$ and $p_j \notin V_\Psi)$ then
    $\{V_\Psi, \mathbb{C}_{V_\Psi}\} \leftarrow \{V_\Psi \cup \{p_j\},\ \mathbb{C}_{V_\Psi} \cup msg_j.\mathbb{C}_j\}$
    if $(|V_\Psi| = m_v)$ then                                   ▷ Enough validators to start transaction
        $tx_{plcr} \leftarrow \mathsf{createPlatypusTx}(\mathbb{C}_{P_\Psi}, \mathbb{C}_{V_\Psi}, plid)$
        $tx_{plcr} \leftarrow \mathsf{sign_i}(tx_{plcr})$
        $\mathsf{multicast}(tx_{plcr}, \{msg_k\}_{p_k \in P_\Psi})$ to $V_\Psi$

▷ PHASE 4: $p_i \in V_\Psi$ signs and broadcasts until it gets enough signatures
when $(tx_{plcr}, \{msg_j\}_{p_j \in P_\Psi})$ is received and not $\mathsf{is\_written}(\Omega, tx_{plcr}, plid)$    ▷ if $tx_{plcr}$ with plid not written in $\Omega$
if $(\mathsf{verify}(tx_{plcr}, \{msg_j\}))$ then $tx_{plcr} \leftarrow \mathsf{sign_i}(tx_{plcr})$
    if $(\mathsf{num\_signers}(tx_{plcr}) < \lfloor 2m_v/3 \rfloor + 1)$ then
        $\mathsf{multicast}(tx_{plcr}, \{msg_j\})$ to $V_\Psi$
    else $\Gamma.\mathsf{send}(\Omega, tx_{plcr})$                                   ▷ enough signatures
```

FIGURE 5.16 – Algorithm to create a chain in Platypus [4].

our model, the upper bound on the number of corrupted players means that no negative payoff is given to the players under the hypothesis that there are at most $\lfloor \frac{m_v}{3} \rfloor$ Byzantine nodes i.e., that the mechanism is $\lfloor \frac{m_v}{3} \rfloor$-weak-immune.

Theorem 5.21 *The mechanism $(\Gamma^{cr}, \sigma^{cr})$ is optimal resilient and $\lfloor \frac{m_v}{3} \rfloor$-weak-immune.*

Proof. *Under the strategy profile σ^{cr} the validators consider all the processes ($u = t = U$), thus their utility reach its maximum $|U|$. The other users have only two strategies, where broadcasting their* *message is the only strategy played at the equilibrium. Therefore, the payoffs generated by σ^{cr} cannot be increased and the mechanism Γ^{cr}, σ^{cr} is strongly resilient.*

For normal users the strategy Y dominates N (the utility is 1 which is larger then 0), while for validators (a_U, b_U, s_U) dominates every other strategy ; any other strategy would provide a payoff lower than $|U|$. Hence, the strategy profile σ^{cr} is the only one with weakly dominating strategies and thanks to Proposition 5.2 we get that the mechanism is practical.

In order to prove weak immunity, we apply Proposition 5.4. We need to prove that every player never gets negative utility when following the protocol, when all the other players become adversarial. Validators do not have negative utility, thus it is enough to prove that neither the other users do. In the worst case scenario for user $u \in U$ a wrong process is validated. To do so, another user $u' \in U$ should be publish it and the validators should approve it. Under the assumption that there at most $\lfloor \frac{m_v}{3} \rfloor$ corrupted processes, in [4] it is proved that this is not possible. The proof follows from the intuition that the Byzantine validators, owning less than a third of the network, cannot validate two different transactions including one which can damage the user u. Therefore, users never get negative utility if there are at most $\lfloor \frac{m_v}{3} \rfloor$ Byzantine players. This corresponds to the definition of $\lfloor \frac{m_v}{3} \rfloor$-weak-immunity (see Definition 5.25). ■

5.4.3 Cross-Chain Swap

In this section we analyze the protocol introduced in [5], that allows two users to swap assets belonging to two different blockchains, which do not communicate with each other. In [6] the authors introduce a theoretical framework proving that the protocol is correct for those players who are altruistic, no matter what the others do. In the following we prove that the cross-chain swap protocol [5] satisfies the (k, t)-weak-robustness.

The protocol, presented in Section 4.4.1, is based on transactions that have to be published within two different time intervals, Δ_{Bob} and Δ_{Alice} on the corresponding blockchain with $\Delta_{Alice} < \Delta_{Bob}$. To recall the protocol we presented before. Alice creates two transactions on the Bitcoin blockchain : TX1, that lets Bob receive an amount of bitcoins if he provides x, and TX2, that gives back the amount to Alice if Bob does not provide x within Δ_{Bob}. Bob creates two transactions on the Litecoin blockchain : TX3, that lets Alice receive an amount of litecoins if she provides x, and TX4, that gives back the amount to B if A does not provide x within Δ_{Alice} hours (in [5] $\Delta_{Alice} = 24$). The theoretical bounds for Δ_{Bob} and Δ_{Alice} are provided in [6]. In a context with two players, the condition is that $\Delta_{Bob} \geq 2\Delta_{Alice}$. From now on we consider the assumption that Δ_{Bob} and Δ_{Alice} fulfill the properties set in [6], and specifically we have that $\min(\Delta_{Bob}, \Delta_{Alice}) = \Delta_{Alice}$.

Since the two blokchains are independent we model the protocol with two different mechanisms $(\mathcal{G}_1, \sigma_1)$ and $(\mathcal{G}_2, \sigma_2)$ (see Definitions 5.14 and 5.15), that represent the actions that the players perform in each blockchain. We set to 0 the utility of the initial state, 1 the utility of every state in which the player receive what is asked, -1 the utility of every state in which the player gives some coins without receiving any. The Bitcoin blockchain is represented by game $\mathcal{G}_1$, while the Litecoin blockchain by $\mathcal{G}_2$ (see Fig. 5.17). We work under the assumption that a transaction can be published within $\min(\Delta_{Bob}, \Delta_{Alice}) = \Delta_{Alice}$ hours.

Definition 5.14 *The Bitcoin game is an extensive form game $\mathcal{G}_1$ with 2 players $N = \{A, B\}$ and 5 nodes (1 is the vertex) :*

1. *A can either Y, pick a random string x, create TX1 and TX2, then send TX2 to B, or doing none of them N. The action Y leads to node 2, while the action N leads to the outcome $(0, 0)$.*
2. *B can either Y, sign TX2, that leads to node 3, or N refusing to do it, with outcome $(0, 0)$.*
3. *A can either do nothing N, with thus outcome $(0, 0)$, or Y publish TX1 on the Bitcoin blockchain, that leads to node 4.*
4. *Both A and B have available two actions : either Y publish TX2 before that x is revealed or N not. If any of the two does so, the outcome is $(0, 0)$. Otherwise, A reveals x and (N, N) leads to node 5.*
5. *B can either Y publish x on the Bitcoin blockhain or N not doing it. If she does, the outcome is $(1, 1)$. If she does not, the outcome is $(1, -1)$.*

The strategy profile recommended by the protocol is $\sigma_1 = (\{Y, Y, N\}, \{Y, N, Y\})$, respectively played at nodes $(\{1, 3, 4\}, \{2, 4, 5\})$. Until x is revealed, the transactions cannot be triggered, therefore they provide null payoff. When x is revealed on the other chain, A has received the litecoins (thus with payoff equal to 1). If at step 5 B reveals x, she triggers the contract and receives the bitcoins (payoff equal to 1). Otherwise she has lost her asset in litecoins (negative payoff -1).

Definition 5.15 *The Litecoin game is an extensive form game $\mathcal{G}_2$ with 2 players $N = \{A, B\}$ and 5 nodes (1 is the vertex) :*

1. *B can either Y, create TX3 and TX4 and send the latter to A, or doing nothing N. The action Y leads to node 2, while the action N leads to the outcome $(0, 0)$.*
2. *A can either Y, sign TX4, that leads to node 3, or N refusing to do it, with outcome $(0, 0)$.*
3. *B can either do nothing N, with thus outcome $(0, 0)$, or publish TX3 on the Litecoin blockchain (Y), that leads to node 4.*
4. *Both A and B have available two actions : either publish TX4 (Y) before that x is revealed or not (N). If any of the two does so, the outcome is $(0, 0)$. Otherwise, A reveals x and (N, N) leads to node 5.*
5. *A can either publish x on the Litecoin blockhain (Y) or not doing it (N). If she does, the outcome is $(1, 0)$. If she does not, the outcome is $(0, 0)$.*

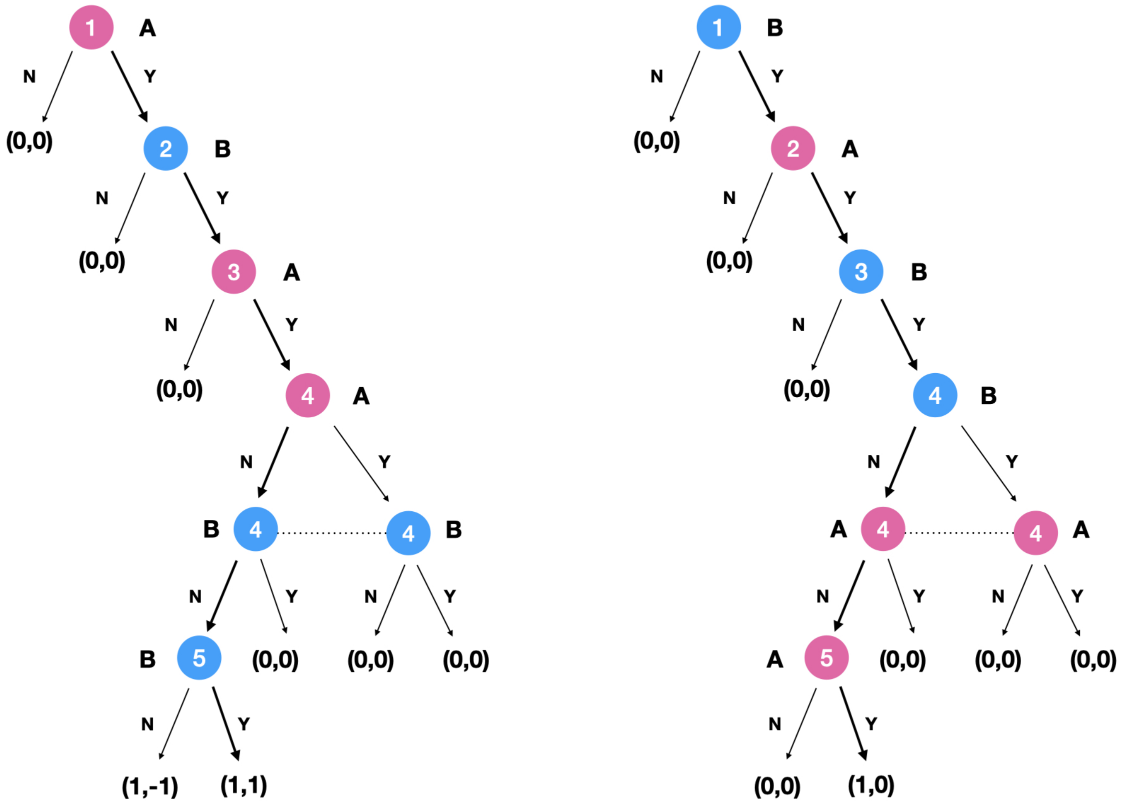


FIGURE 5.17 – The game trees of $\mathcal{G}_1$ and $\mathcal{G}_2$.

The strategy profile recommended by the protocol is $\sigma_2 = (\{Y, N, Y\}, \{Y, Y, N\})$, respectively played at nodes $(\{2, 4, 5\}, \{1, 3, 4\})$. Until x is revealed, the transactions cannot be triggered, therefore they provide null payoff. When x is revealed, A receives the litecoins (thus with payoff equal to 1). B does not know if he receives the asset, hence her payoff is 0.

Since the two blockchains are independent, we consider the composition of the two games $(\mathcal{G}_1 \odot \mathcal{G}_2, \{\sigma_{1i}, \sigma_{2i}\})$ that represents the full protocol and analyze its properties.

Theorem 5.22 *Under the assumption that any transaction can be published within a time interval $[0, \Delta_{Alice}]$, the mechanism $(\mathcal{G}_1 \odot \mathcal{G}_2, \{\sigma_{1i}, \sigma_{2i}\})$ is not immune.*

Proof. *The strategy profile $\{\sigma_{1i}, \sigma_{2i}\}$ provides outcome*

$$u_{\mathcal{G}_1 \odot \mathcal{G}_2}(\{\sigma_{1i}, \sigma_{2i}\}) = u_{\mathcal{G}_1}(\sigma_1) + u_{\mathcal{G}_2}(\sigma_2) = (1, 1) + (1, 0) = (2, 1)$$

If B considers a strategy σ_B^* that lets her play action N at node 2 of the Bitcoin game and action N at node 1 of the Litecoin game, the outcome is

$$u_{\mathcal{G}_1 \odot \mathcal{G}_2}(\{\sigma_{1A}, \sigma_{2A}\}, u_B^*) = u_{\mathcal{G}_1}(\sigma_{1A}, \sigma_{1B}^*) + u_{\mathcal{G}_2}(\sigma_{2A}, \sigma_{2B}^*) = (0, 0) + (0, 0) = (0, 0)$$

thus reducing the payoff for player A. In a two-player game a mechanism is immune if it is 1-immune (see Definition 5.4), but in this case A receives a loss if B performs a specific Byzantin behavior. ■

Theorem 5.23 *Under the assumption that any transaction can be published within an interval of time Δ_{Alice}, the mechanism $(\mathcal{G}_1 \odot \mathcal{G}_2, \{\sigma_{1i}, \sigma_{2i}\})$ is optimal resilient and weak immune.*

Proof. *It is enough to prove that the two mechanisms $(\mathcal{G}_1, \sigma_1)$ e $(\mathcal{G}_2, \sigma_2)$ satisfy the properties and then exploit the properties of the operator composition of games.*
In game $\mathcal{G}_1$ the strategy profile σ_1 is the only one with outcome $(1, 1)$, which is maximal. Thus we have that $(\mathcal{G}_1, \sigma_1)$ is strongly resilient.
Every strategy different from σ_1 is weakly dominated, indeed they bring to either outcome -1 or 0, which is lower than $u_1(\sigma_1) = (1, 1)$. Thus σ_1 is a stable Nash equilibrium and for Proposition 5.2 we have that the mechanism $(\mathcal{G}_1, \sigma_1)$ is practical.
In order to prove weak immunity we apply Proposition 5.4. When following respectively strategies σ_{1A} and σ_{1B} both A and B never get negative utility. Therefore the mechanism $(\mathcal{G}_1, \sigma_1)$ is also weak

immune.
In game $\mathcal{G}_2$ the strategy profile σ_2 produces an outcome $(1, 0)$ which is maximal for both players, thus we have that the mechanism $(\mathcal{G}_2, \sigma_2)$ is strongly resilient.
The strategies within σ_2 are never weakly dominated, because none of the others can provide a better outcome. Hence the mechanism is practical.
Every outcome is non-negative, therefore the mechanism is weak immune.
Since both mechanisms are optimal resilient and weak immune, we can apply Theorems 5.2, 5.3 and 5.4, that ensure the invariance of the properties once the operator composition is applied. The mechanism $(\mathcal{G}_1 \odot \mathcal{G}_2, \{\sigma_{1i}, \sigma_{2i}\})$ is thus optimal resilient and weak immune. ■

The mechanism is not immune, indeed it is sufficient that one player does not create or publish a transaction to stop the protocol. Under the assumption that any transaction can be published within a time interval $[0, \Delta_{Alice}]$ the mechanism is optimal resilient and weak immune.

5.5 Summary

Summary. The contribution proposes the first general game theoretical framework that models the robustness of blockchains towards rational and Byzantine behaviors. Blockchain users faced with a blockchain prescribed protocol may act rationally (following or deviating according to their personal utility) or as a Byzantine agent (i.e., deviating from the protocol for any possible arbitrary reason even irrational). This chapter models the behaviors of blockchain users acting as validating nodes (Section 5.3) and transacting parties (Section 5.4). We identify the necessary and sufficient conditions for a protocol to be robust (defined as the conjunction of two properties : k-resilience and t-weak immunity) and develop a methodology to characterize the robustness of complex protocols via the composition of simpler robust building blocks. The effectiveness of our framework is proved by its capability to capture the robustness of various blockchain protocols such as Bitcoin, Tendermint, Lightning Network, side-chain and cross-chain protocols. Our work continues the work of [309] that introduced the notion of robustness defined in terms of t-immunity and k-resilience. The framework of [309] was never used till our study in the context of blockchain protocols. Using the framework of [309] we prove that a large class of blockchain protocols (see results in Table 5.2) does not satisfy the t-immunity property. It should be noted that our negative result related to the t-immunity property does not depend on the specific choice of a utility function. Therefore, we propose a relaxation of this property i.e., t-weak immunity and analyze the t-weak immunity of a large class of blockchain protocols. For each one of them we prove bounds on the number of Byzantine processes allowed for this property to hold. Furthermore, for the same class of protocols we compute bounds on the number of rational processes in order to achieve k-resilience.

Conclusion

General Conclusions of the Thesis

For a new technology to realize its full potential, a lot of circumstances need to co-exist before network effects can be realized. In order for the technology to bring in systemic efficiencies, a critical mass needs to be attained. As an infrastructure technology, all major players in the market need to collaborate to define standards in a democratic manner. The blockchain community is indeed witnessing unprecedented levels of industry collaboration between players who are otherwise competitors in the space. Because of the cost of moving from one infrastructure technology to the next, an collaborative approach is the most promising way forward. This is the direction we insisted on in the first part of this thesis and during my three-years working experience within the “Blockchain and Crypto-asset Programme” of Caisse des Dépôts Groupe [316].

From a societal perspective, while there has been an exponential increase in the interest around blockchain technologies, there is a huge lack of technical experts. Currently, blockchain engineers become one of the most payed and required jobs, yet there are really few recognized courses to train engineers to fulfill the existing lack of blockchain experts.

On the other hand, there are more and more actors joining existing blockchain network for variety of reasons ; speculation, asset tokenization, blockchain intermediary service, gaming, smart contract applications and so on so forth. Any new node of the network represents a user intended to interact with the network of peers. Interactions, of different nature, might contribute to the durability as well as the destruction of a blockchain system.

Contributions presented in the thesis analyze the different behaviors of blockchain users by considering them as rational agents, fully aware of all actions available to them and capable of choosing the one they feel is the best for themselves. The manuscript takes into consideration different users as well as different blockchains with the scope of providing a general overview of the topic and formal results holding for DLTs in general.

Adopting an agnostic approach, this work analyzes those blockchain features and attributes making blockchain-based use cases more efficient or more innovative. An overview of when to use blockchain and which type of blockchain to use is provided in Chapter 1. After this overview, formal analysis are performed by combining distributed systems theory as well as game theory (Chapter 2) to analyze the behaviors of users operating on blockchain and DLT systems. This way of operating significantly differs considering the type of users. The contributions presented in Chapters 3-5 analyze both users simply adopting the technology as a payment/transfer mean (i.e., transacting parties) and users maintaining the network by participating in the consensus process (i.e., validating nodes also known as miners). Rationality limitations are taken into account by analyzing blockchain robustness to both rational and Byzantine behaviors (Chapter 5).

Industrial Contributions

This thesis was carried out within a French industrial PhD program called CIFRE in which my industrial partner was Caisse des Dépôts Groupe [316] ; a French financial institution. During these three years of thesis, the theoretical research has been accompanied by industrial applications of various nature. Caisse des Dépôts engaged in a mission of accompanying the different types of blockchain users in the adoption of the technology for which the analysis performed in the first part of the thesis was fundamental. Moreover, the public entity cooperates at a national [2] and international [3] level to facilitate the use of the technology in compliance with regulations of the different jurisdictions. Technical reports on new blockchain systems and protocols have been presented in various acculturation initiatives organized by the Think Tank LaBChain consortium chaired by Caisse des Dépôts. Cross-chain swap protocols between different blockchains have been implemented for Delivery-versus-Payment experiments swapping security tokens (i.e., digital securities issued on a blockchain) with stable coins (i.e., tokens pegged to stable assets such as fiat currencies).

[2] Caisse des Dépôts is a member of ADAN [317] association representing professionals in the digital assets and blockchain technologies sector in France.

[3] Caisse des Dépôts is a member of the International Association for Trusted Blockchain Application, INATBA [318].

Future Work Directions

In this section, we present research questions opened up from the contributions presented in this manuscript. We order these research directions from the more general to the more specific.

Rational Behaviors.

Our work raises immediate research questions on the rationality of blockchain users behaviors. After presenting the rationality concept in game theory we encounter throughout the different contributions the limitations of such assumption in real life situations. Indeed, players may act irrationally or may be led to deviate by external phenomena. This opens up an interesting research direction aiming at modeling blockchain systems with fewer assumptions, meaning model real-word blockchain protocols. These analyses are much more complex to handle and quite delicate, but they will probably have huge impacts on the field, and on the understanding of the systems. An additional interesting question concerns the position of results and theoretical analysis with respect to to real world behaviors of participants. Chapter 3 provides such positioning for the pool-hopping behaviors. To do such comparison for a larger set of behaviors, behavioral sciences need to be integrated, requesting collaboration among different research fields.

Blockchain Users' Behaviors.

In this thesis, we propose a game theoretical analysis of blockchain users' behaviors. The work models the robustness of blockchains towards a particular set of rational and byzantine behaviors i.e., protocol deviations (attacks) presented in the literature or observed in real life scenarios. The work presented in Chapter 5 analyzes Bitcoin [2], Tendermint [1], Lightning Network [3], a side-chain protocol [4] and a cross-chain swap protocol [5] however, other types of blockchain protocols should be modeled as well with the developed framework e.g., Algorand [319] or DAG-based blockchains (e.g., Spectre [320], Phantom [321] or IOTA [322]). Moreover, as blockchain represents a disruptive technology, new protocols as well as new attacks to such protocols are discovered on a monthly basis. The analysis performed in Chapter 5 should then be extended not only to new blockchain protocols but also to include new protocol deviations.

Solution Concepts.

In this manuscript, when dealing with rational agents, we adopt the notion of Nash equilibrium in its stable and strong variants. A possible research direction can be the analysis of blockchain users' behaviors with other solution concepts. A possibility would be considering mixed strategies, i.e., each participant instead of choosing deterministically one strategy to play for the game, chooses a distribution of the different strategies. Another path can be to consider cooperation between strategic participants to analyze those problems we modeled with non-cooperative games. Indeed, layer-2 protocols analyzed in Chapter 5, can be modeled as cooperative games where transacting parties form coalitions. Analyzing the resilience of the cross-chain swap protocols to deviating coalitions is an interesting research direction enhancing blockchain interoperability.

Rewarding Miners.

Future works concerning the problem of rewarding miners consist in applying the analysis performed in Chapter 3 to other crypto-currencies with comparable consensus methods, such as Litecoin. However, PoW-based algorithms and remunerating systems are now giving the way to new consensus mechanisms more energy friendly i.e., PoS. Mining is then replaced by virtual mining not performing any computational task requiring the need to form pools. Moreover, PoW consensus is now more oriented to pay-per-share rewarding method as bankruptcy situations are very rare. Pools are now better organized with reserves to face these issues. The mining industry is then becoming more and more Bitcoin-centric leading this research to take a completely new direction towards the problem of rewarding validating nodes participating in alternative consensus mechanisms. Furthermore, modeling the rewarding functions of validators operating in different blockchain systems (e.g., mining Bitcoin while validating transactions in Ethereum) represents a challenging research direction.

Synthèse

En examinant les technologies informatiques, les architectures et les pratiques de conception du dernier demi-siècle, nous pouvons observer une tendance à la fluctuation entre la centralisation et la décentralisation des ressources informatiques telles que la puissance de calcul, le stockage, l'infrastructure, les protocoles et le code. Les ordinateurs centraux sont largement centralisés et concentrent la plupart des ressources informatiques. Aujourd'hui, les capacités de calcul sont distribuées sur les clients et leurs équipements ou encore sur des serveurs distants. Cette approche a donné naissance à l'architecture "client-serveur" qui a permis le développement de l'Internet et des systèmes de bases de données relationnelles.

Les ensembles de données massives, initialement hébergés sur des ordinateurs centraux, peuvent passer à une architecture distribuée, avec des données répliquées d'un nœud à l'autre, ou d'un serveur à l'autre, et des sous-ensembles de données peuvent être consultés et traités via 'clients', puis synchronisés avec l'un des serveurs.

Au fil du temps, les architectures Internet et de cloud computing ont permis un accès mondial à partir d'une variété de dispositifs informatiques, alors que les mainframes ont été largement conçus pour répondre aux besoins des grandes entreprises et des gouvernements. Même si une telle architecture Internet/Cloud est décentralisée en termes de matériel, elle a donné lieu à une centralisation au niveau des applications. Actuellement, nous assistons à la transition d'une informatique, d'un stockage et d'un traitement centralisés vers des architectures et des systèmes décentralisés.

La DLT est l'innovation majeure qui rend ce changement possible. Certains systèmes distribués (par exemple, les blockchains sans permission i.e., *permissionless*) visent à donner le contrôle des actifs numériques aux utilisateurs finaux sans avoir besoin de nœuds intermédiaires. D'autres (par exemple, les blockchains avec permission i.e., *permissioned*) tentent de maintenir une centralisation logique de certaines informations tout en adoptant une architecture décentralisée. Les DLT ne font pas toutes appel à une architecture de blocs et peuvent donc être définies comme des "blockchains" (e.g., *The Tangle* and *BigchainDB* [9; 10]). Les blockchains peuvent donc être utilisées dans divers secteurs avec plusieurs applications. En effet, il est crucial pour les adoptants de comprendre si la technologie correspond ou pas aux problèmes qu'ils visent à résoudre.

La technologie disruptive née en 2008 avec Bitcoin et connue sous le nom de blockchain représente un saut qualitatif important par rapport à la technologie des bases de données distribuées étudiée depuis juin 1970 (date de publication du premier article [11] sur le sujet). La blockchain permet de partager un grand registre de transactions qui sont lues, validées et stockées dans une chaîne de blocs. Les systèmes basés sur la technologie blockchain fonctionnent de manière distribuée, impliquant plusieurs agents ou participants qui devraient être indépendants les uns des autres, et qui peuvent utiliser des communications pair-à-pair (P2P) pour se structurer dans une collectivité de réseau (où les nœuds représentent les utilisateurs qui sont connectés par des transactions). L'adoption du P2P comme paradigme de communication soutient adéquatement l'objectif selon lequel les ressources sont partagées et dispersées sur un réseau qui, par construction, interdit l'existence de fournisseurs ou de serveurs centralisant les tâches. Le résultat est un écosystème décentralisé sans autorité centrale.

Contrairement aux architectures client-serveur traditionnelles [12], les nœuds des réseaux P2P n'ont pas toujours une hiérarchie fixe ; les rôles peuvent ne pas être prédéterminés ou peuvent changer au fil du temps en fonction de l'opération réelle sous-jacente à une communication, c'est-à-dire une transaction blockchain.

Transactions Blockchain et Rôles des Utilisateurs

Chaque fois qu'un utilisateur souhaite interagir avec un autre dans un réseau blockchain, une ou plusieurs transactions sont créées, propagées, validées et confirmées par le réseau. Ce parcours commence au moment où la transaction est créée et se termine lorsque la transaction est enregistrée dans la blockchain. Quatre étapes cruciales du parcours d'une transaction blockchain peuvent être identifiées :

— *Création* : la partie émettrice d'une transaction doit définir, selon le modèle de données, l'origine et la destination de "l'objet du transfert" (i.e., l'actif numérique). Les transactions doivent également préciser les conditions dans lesquelles l'objet de la transaction peut être racheté (i.e., les conditions de mise à jour de l'état du système). Selon le modèle, les critères de rachat peuvent être de simples scripts ou plus généralement des contrats réels (i.e., *smart contracts* ou contrats intelligents).

- *Propagation* : la transaction (dans un éventuel bloc) est propagée aux pairs qui la valident. Une diffusion efficace des transactions a un impact sur la vitesse de traitement des transactions.
- *Validation* : c'est l'étape la plus cruciale puisqu'elle caractérise tous les systèmes existants basés sur une blockchain. Lors de cette étape, les transactions, rassemblées en blocs, doivent adresser les différentes étapes du mécanisme de consensus envisagé pour être considérées comme valides et donc exécutables. Ensuite, le bloc de transactions peut être rattaché à la blockchain, mettant à jour son état. Le bloc de transactions valide est ensuite propagé à l'ensemble du réseau afin de permettre à tous les nœuds de mettre à jour leur propre version du registre.
- *Confirmation* : les blocs de transactions ne donnent lieu à un véritable transfert d'actifs que si, une fois validés et éventuellement publiés sur la blockchain, ils sont confirmés dans la version finale du registre, dont ils ne peuvent plus être éliminés. Pour en faire partie, la procédure de consensus doit arriver à son terme, c'est-à-dire que les nœuds doivent se mettre d'accord sur une seule chaîne de blocs.

Les transitions d'une étape à l'autre caractérisent la technologie. La cryptographie est impliquée dans les techniques de hachage et de génération de clés. Les vérifications et la formation de blocs peuvent relier les deux premières étapes ou les étapes centrales. Ces quatre étapes marquent une forte distinction entre deux niveaux d'utilisation de la technologie : (i) l'utilisation de la technologie pour effectuer des transactions simples qui seront ensuite propagées sur le réseau (i.e., la création et la propagation) et (ii) l'aide à la maintenance du système en participant à la phase de validation et de confirmation des transactions (rassemblées en blocs).

Les 'valideurs' sont tous les pairs impliqués à partir du moment où la transaction est incluse dans un bloc jusqu'à sa publication sur le registre. Les pairs qui collectent les transactions dans les blocs peuvent ne pas entrer dans la phase de validation ; cependant, étant donné que pour de nombreux systèmes de blockchain, la collecte des transactions fait partie du processus de validation, cette séparation n'est pas évoquée dans ce manuscrit. La section 1.3.5 présente une division plus détaillée des rôles que les utilisateurs blockchain assument, nous soulignons dans ce qui suit les principaux.

- *Parties Contractantes* : une transaction blockchain implique deux types d'acteurs différents liés à des utilisateurs uniques ou multiples de la blockchain : le *data-sender* et le *data-receiver*. Les interactions ont lieu au niveau des adresses : le(s) *adresse(s) d'envoi* et le(s) *adresse(s) de réception* suivent numériquement le flux de données (i.e., le transfert d'actifs numériques) entre les parties.
- *Nœuds de Validation* : les acteurs validants exécutent l'algorithme de consensus et sont chargés d'établir l'accord sur les propositions faites par les autres valideurs ou par les nœuds leaders (voir la section 1.3.5). La validation d'un bloc représente le consensus entre les nœuds valideurs sur le bloc à publier et sur son ordre de publication.

Comportements des Utilisateurs Blockchains

En tant que systèmes multi-agents, les systèmes blockchain, caractérisés par différentes couches et protocoles, peuvent être modélisés comme des *jeux* à résoudre à l'aide de la théorie des jeux. Les agents qui caractérisent les blockchains sont des êtres humains ou des systèmes dont les comportements sont programmés par des êtres humains, même si on les appelle *nœuds*, *pairs, agents* ou *joueurs.* Il s'agit bien d'agents qui peuvent être considérés comme rationnels. Être rationnel ne consiste pas banalement à se comporter de manière égoïste sans tenir compte de la satisfaction des autres. En revanche, les agents rationnels cherchent à obtenir le mieux pour eux-mêmes et cela peut parfois correspondre au mieux pour la communauté. La rationalité a deux fortes connotations : (i) elle suppose que les agents sont capables d'ordonner les résultats en fonction de certaines préférences et (ii) qu'ils sont capables d'analyser le problème auquel ils sont confrontés et de prendre les bonnes décisions. Ces deux hypothèses majeures ne sont pas toujours respectées par les utilisateurs blockchain, ce qui les conduit à être classés comme des nœuds altruistes et byzantins selon [13].

Les utilisateurs blockchains jouent des jeux spécifiques en fonction de leur rôle au sein du système blockchain ; ils peuvent participer à la phase de consensus, ils peuvent simplement interagir entre eux via des transactions ou encore ils peuvent être intéressés à attaquer le système pour l'endommager ou avec la simple intention d'en tirer profit. Les comportements qu'ils peuvent avoir peuvent être honnêtes, malhonnêtes ou armés. L'analyse de tous ces différents comportements des utilisateurs blockchain aide à construire des protocoles blockchain robustes caractérisant des systèmes blockchain robustes.

Ce manuscrit analyse les deux principaux types d'utilisateurs blockchains (i.e., les parties contractantes et les nœuds de validation) ainsi que différentes blockchains (i.e., sans permissions et avec permissions) dans le but de fournir un aperçu général du sujet et des résultats formels sur les comportements des utilisateurs blockchains ; les utilisateurs peuvent en effet être honnêtes vis-à-vis des autres utilisateurs ou ils peuvent se comporter de manière malveillante (comme des nœuds byzantins) en attaquant un système blockchain. Les deux types d'utilisateurs blockchain sont d'abord modélisés comme des agents rationnels. Cette modélisation basée sur la théorie des jeux permet de capturer plusieurs comportements. Pour être plus générique (i.e., en incluant également les comportements malveillants irrationnels ou inattendus), les utilisateurs blockchain sont également modélisés comme des agents byzantins ; ceci afin d'évaluer la robustesse de tout type de déviation du protocole dans le contexte de la blockchain. Le manuscrit est structuré comme suit :

- **Chapitre 1** présente un aperçu complet de la technologie blockchain en analysant ses principales caractéristiques ainsi que toutes les couches qui caractérisent l'architecture blockchain. Il se concentre en particulier sur les caractéristiques qui conduisent les utilisateurs blockchain à choisir (i) si adopter la technologie ou non et (ii) quel type de blockchain choisir. Le contenu de ce chapitre provient de [14].
- **Chapitre 2** fournit une introduction à la Théorie des Jeux, en mettant l'accent sur la *rationalité des joueurs prenant part à un jeu.* Cela permet de prévoir les actions des joueurs et donc les résultats du jeu qui peuvent être plus ou moins stables (i.e., des équilibres ou des solutions dans le noyau). Les participants au jeu peuvent jouer en tant que individu ou en tant que groupe, ce qui conduit à deux types de modélisation différents.
- **Chapitre 3** modélise et analyse les comportements d'un type particulier de *nœuds de validation* ; les mineurs du Bitcoin. Les mineurs qui effectuent des tâches de validation dans la blockchain Bitcoin sont modélisés comme des agents rationnels qui peuvent ou non avoir un comportement malveillant, c'est-à-dire effectuer une attaque, dans certaines situations. La situation que nous présentons dans ce chapitre est le processus de rémunération des mineurs pour leur travail de validation. Deux types de comportements malveillants sont analysés : le *pool-hopping* et le *block-withholding.* Le contenu du chapitre est issu de [15] et [16].
- **Chapitre 4** est consacré à l'analyse des comportements des *parties contractantes* visant à échanger des actifs via des swaps cross-chain. Les échanges inter-chaînes rendent possible les interactions

entre plusieurs blockchains, c'est-à-dire l'interopérabilité des blockchains. La théorie des jeux est utilisée pour modéliser les parties prenantes (qui échangent des crypto-actifs) comme des agents rationnels et pour caractériser les équilibres des protocoles cross-chain existants. Le contenu de ce chapitre est pris de [17].

- **Chapitre 5** réunit les deux types d'utilisateurs qui sont analysés dans la section 5.3 (i.e., les nœuds de validation pour les protocoles blockchain *layer-1*) et la section 5.4 (i.e., les parties qui effectuent des transactions par le biais de protocoles blockchain *layer-2*), respectivement, d'une manière plus générale, en allant au-delà des hypothèses fortes (et des limitations répétitives) de la rationalité des agents présentes dans les chapitres précédents. Ce chapitre propose un cadre de théorie des jeux qui caractérise la robustesse des systèmes blockchain en termes de résilience aux déviations rationnelles et d'immunité aux comportements byzantins. Les utilisateurs rationnels confrontés à un protocole blockchain à suivre peuvent agir de manière altruiste (en suivant le protocole) ou malveillante (en déviant du protocole) en fonction de leur utilité personnelle. En revanche, les utilisateurs byzantins s'écartent du protocole dans n'importe quelle situation (même en agissant de manière irrationnelle). Nous prouvons l'intérêt pratique de notre cadre formel en caractérisant la robustesse de deux protocoles blockchain layer-1 (i.e., Bitcoin [2] et Tendermint [1]) et de trois protocoles blockchain layer-2 (i.e., Lightning Network [3], un protocole side-chain [4] et un protocole de swap inter-chaîne [5]). Le contenu de ce chapitre provient de [18].

Le tableau suivant représente la structure d'analyse du manuscrit. Les deux premiers chapitres introduisent les sujets "Blockchain" et "Théorie des jeux".

		Modèle de comportement	
		Rationnel	*Byzantin*
Blockchain	*Nœuds de Validation*	Chapitre 3	Chapitre 5 (Section 5.3)
User	*Parties Contractantes*	Chapitre 4	Chapitre 5 (Section 5.4)

Chapitre 1

Le chapitre fournit un aperçu complet de la technologie blockchain en analysant ses principales caractéristiques ainsi que toutes les couches caractérisant l'architecture blockchain en se concentrant particulièrement sur celles qui aident les utilisateurs potentiels de blockchain à décider (i) s'ils doivent adopter la technologie ou non et, (ii) quel type de blockchain choisir.

Introduction à la blockchain et aux DLTs

La blockchain peut être considérée comme un saut de qualité par rapport à la technologie des bases de données distribuées [19] étudiée depuis les années 70, qui consiste en une base de données de transactions partagée par différents utilisateurs. En général, les technologies de registres distribués (i.e., *Distributed Ledger Technologies*, DLTs) sont conçues pour traiter les bases de données sous la forme de données partagées de manière distribuée, et la blockchain représente une DLT possible pour le faire (voir Fig. 5.18).

Les briques fondamentales de la conception d'une technologie blockchain sont les suivantes : (i) les communications et le stockage des données de transaction sont encadrés par une sécurité cryptographique (les nœuds du réseau doivent s'accorder sur la validité et l'ordre dans lequel les transactions sont répertoriées dans la blockchain) et (ii) les protocoles de consensus distribués résolvent ces problèmes dans un scénario où chaque nœud est amené à voter.

Le premier exemple d'une telle blockchain est Bitcoin, proposé en 2008 par son identité anonyme [20]. Le comportement du Bitcoin trace ce qui peut être défini comme la blockchain 'classique', consistant en une alternative de blockchain *permissionless* permettant un système de paiement numérique, distribué et décentralisé. La blockchain Bitcoin est structurée de manière à protéger l'écosystème

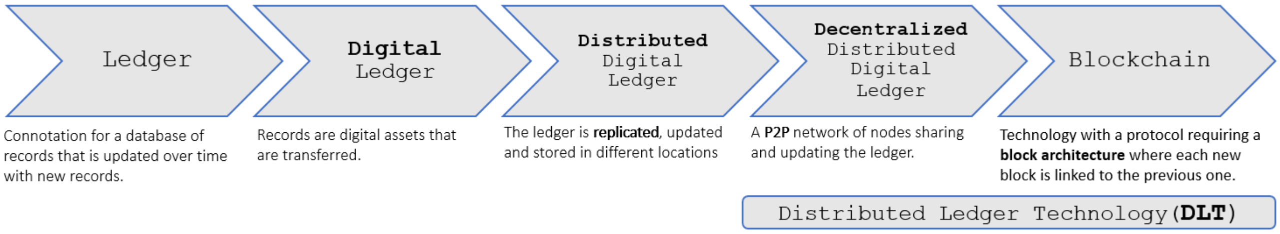

Ledger
Digital Ledger
Distributed Digital Ledger
Decentralized Distributed Digital Ledger
Blockchain
Connotation for a database of records that is updated over time with new records.
Records are digital assets that are transferred.
The ledger is replicated, updated and stored in different locations
A P2P network of nodes sharing and updating the ledger.
Technology with a protocol requiring a block architecture where each new block is linked to the previous one.
Distributed Ledger Technology (DLT)

FIGURE 5.18 – Évolution de la DLT : du registre traditionnel à la blockchain.

contre les attaques lancées par des nœuds malveillants ou simplement rationnels du réseau. Même si le bitcoin reste la crypto-monnaie la plus populaire en circulation, un grand nombre de crypto-monnaies basées sur la blockchain ont été définies – à ce jour, plus de 50 crypto-monnaies alternatives existent. Certaines de ces 'Altcoins' [25] s'appuient sur une blockchain *permissioned* – accessible uniquement aux nœuds autorisés, afin d'offrir un système plus scalable et rapide.

Au-delà du cas du Bitcoin, la technologie blockchain vise à garantir les avantages des tiers de confiance tels que l'intégrité, l'authenticité, la sécurité et la non-répudiation dans un environnement distribué et décentralisé. En plus de l'auditabilité et de la transparence, elle offre l'immuabilité. En plus d'être évidentes pour les systèmes de crypto-monnaies, ces caractéristiques sont utiles pour tout système transactionnel qui doit être utilisé par plusieurs parties indépendantes qui ne se font pas confiance (i.e., *trustless*).

Avec l'introduction des blockchains permissionés, les utilisateurs peuvent choisir de les adopter en imposant des contraintes et en personnalisant le comportement des nœuds du réseau. Alors qu'avec les blockchains classiques, il est possible de construire un système totalement ouvert et décentralisé, les blockchains avec permissions ne permettent qu'à un nombre limité d'utilisateurs d'avoir le droit de valider les transactions. Les valideurs constituent un ensemble de nœuds qui peuvent être élus publiquement ou sélectionnés par une autorité centrale. En limitant le nombre de participants à la procédure de validation, on peut obtenir des améliorations significatives de la scalabilité en utilisant des mécanismes de consensus appropriés. En outre, les modifications apportées aux protocoles (à la fois dans les données de la blockchain et dans la structure de consensus) pour prendre en charge l'exécution de codes *Turing-complete*, facilitent le déploiement d'applications distribuées ('dapps') basées sur des *smart contracts.* Toutefois, comme les blockchains permissionés présentent de nombreuses similitudes avec les bases de données partagées classiques, il peut y avoir des situations où une architecture aussi complexe n'est pas indispensable.

Structure et Caractéristiques de la Blockchain

Au-delà de l'innovation apportée par la blockchain à l'écosystème DLT, un élément fondamental est son mélange intelligent de techniques de cryptage [45] dans le stockage des données - préservant la structure des blocs grâce à l'horodatage [46] - et dans les transactions - authentifiant les transferts avec des signatures numériques. Le registre d'une blockchain consiste en un historique des transactions numériques validées rassemblées en blocs ; chaque bloc de transactions est lié au bloc immédiatement précédent (i.e., *bloc parent*) par une valeur de hachage ; ainsi, en parcourant le registre des transactions, on peut remonter jusqu'au bloc de départ (i.e., *genesis block*), qui n'a pas de bloc parent et contient les premières transactions traitées dans l'historique de la blockchain. La cryptographie caractérise la technologie et lui attribue des propriétés importantes. Nous pouvons mettre en évidence six caractéristiques fondamentales de la blockchain, qui sont dépendantes les unes des autres :

— *Décentralisation* : Les DLT permettent le partage et le stockage de données P2P sans confier la tenue du registre à une autorité centrale. Il ne s'agit pas de supprimer complètement les intermédiaires qui valident les transactions (*disintermédiation*) comme le font les blockchains sans permission, mais plutôt de les décentraliser ainsi que leurs rôles.

— *Immutabilité* : alors que les registres partagés permettent la manipulation des données par une autorité centrale, les registres distribués travaillant avec des informations répliquées protègent les données contre toute sorte de falsification et d'altération ; sauf dans les situations où la majorité des efforts du réseau sont consacrés à modifier le registre [55] ou lorsque les seuils de l'adversaire sont dépassés (voir la section 1.4.1.3). L'immutabilité des données rend les données accessibles et gérables par différentes entités qui ne se font pas confiance.

— *Intégrité, Authenticité and Non-Répudiation* : le hachage des données garantit que les données ne sont pas modifiées pendant leur transmission (i.e., l'intégrité). De plus, l'origine d'une transaction peut être vérifiée par la diffusion de la clé publique de l'expéditeur, tandis que la preuve de l'action d'envoi est représentée par la procédure de signature des données impliquant la clé privée (i.e., l'authenticité et la non-répudiation). Le schéma de signature de la blockchain combinant la cryptographie asymétrique et le hachage de données est présenté dans la section 1.2.4.

— *Auditabilité* : Les transactions dans les systèmes blockchain doivent être validées et vérifiées ; ainsi, chaque transfert de données doit être visible par tous les utilisateurs dans son intégralité. De cette façon, toutes les opérations blockchain sont traçables via des d'audits. Les utilisateurs qui accèdent aux blockchains peuvent voir le registre des données dans son intégralité.

La conjonction des caractéristiques ci-dessus qualifie la technologie à un niveau de sécurité assez élevé, la différenciant de la base de données distribuée classique.

Application de la Blockchain

Cette section est destinée à lancer notre *vademecum* sur la technologie blockchain, à donner au lecteur un guide pour savoir quand utiliser la blockchain, quelle solution utiliser et comment l'utiliser, en fonction des exigences du cas d'utilisation.

Pour développer ce tutoriel, nous avons utilisé un large éventail de documents allant au-delà de la littérature académique, et comprenant des livres, des livres blancs, des rapports techniques, des forums blockchain, des documents de discussion et des encyclopédies en ligne. Nous nous sommes concentrés sur les travaux montrant des applications réelles de la blockchain dans l'industrie allant au-delà des systèmes de paiement numérique bien connus proposés par les crypto-monnaies.

Dans la section 1.5, nous fournissons au lecteur un modèle de décision détaillé pour comprendre *Quand* utiliser la technologie blockchain et *Quel* type de blockchain convient le mieux à un certain cas d'usage. Le modèle de décision est caractérisé par deux chemins de décision (i.e., Quand et Quel) qui peuvent être parcourus consécutivement ou indépendamment ; les points de décision peuvent être des questions directes ou des points de compromis (i.e., *trade-off*). Ces derniers représentent des situations qui impliquent un choix entre deux ou plusieurs aspects, où la perte de valeur d'un aspect constitue une augmentation de valeur pour l'autre ou les autres. Dans l'arbre de décision proposé, les alternatives sont (i) les caractéristiques de la blockchain ou de la base de données traditionnelle pour la question "Quand" et (ii) les caractéristiques de la blockchain avec ou sans permissions pour la question "Quel". Dans ce qui suit, nous rapportons la Fig. 5.19 qui représente un support pour les deux questions.

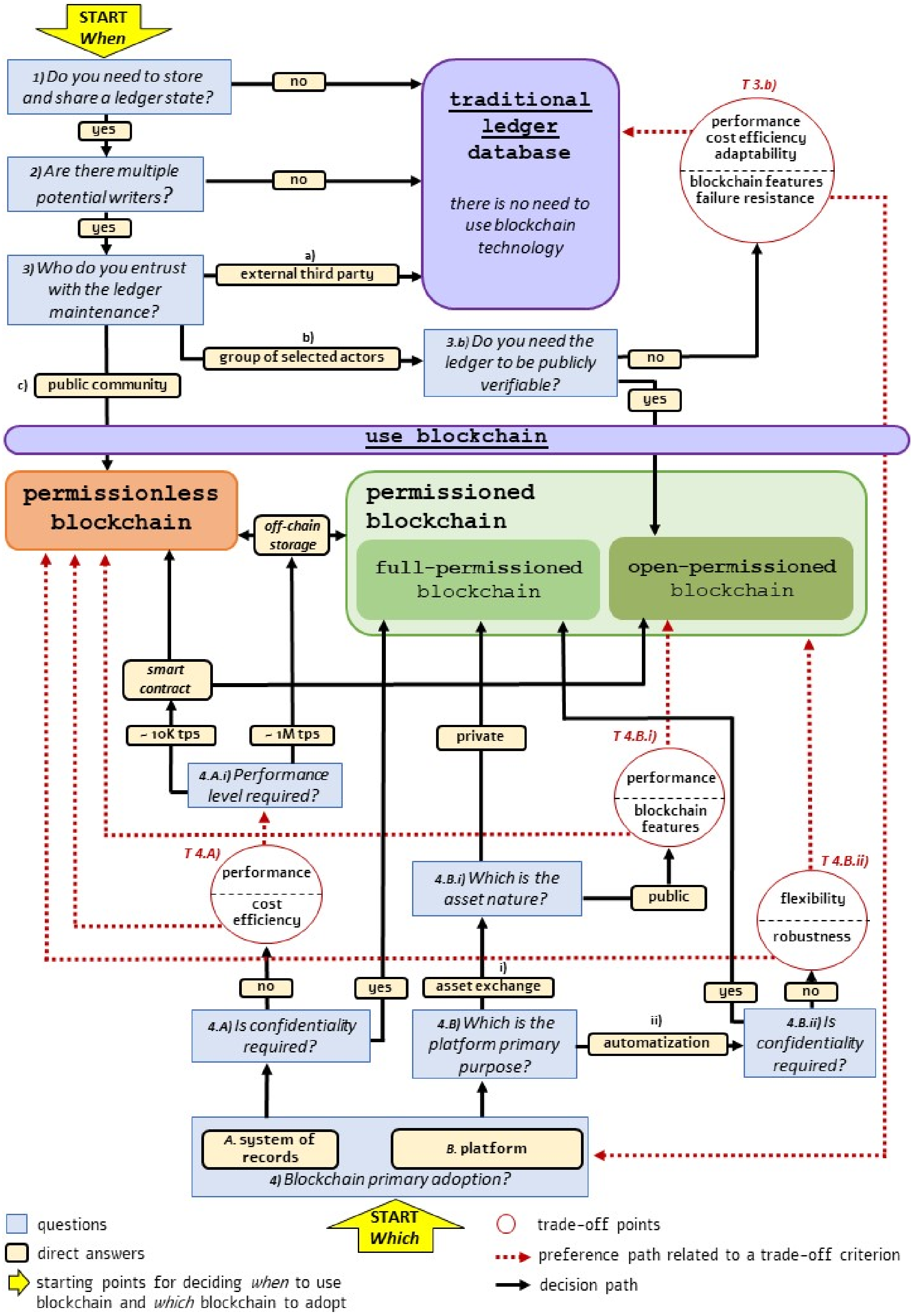

START
When
1) Do you need to store and share a ledger state?
no
yes
2) Are there multiple potential writers?
no
yes
3) Who do you entrust with the ledger maintenance?
a)
external third party
b)
group of selected actors
c)
public community
traditional ledger database
there is no need to use blockchain technology
T 3.b)
performance
cost efficiency
adaptability
blockchain features
failure resistance
3.b) Do you need the ledger to be publicly verifiable?
no
yes
use blockchain
permissionless blockchain
off-chain storage
permissioned blockchain
full-permissioned blockchain
open-permissioned blockchain
smart contract
~ 10K tps
~ 1M tps
4.A.i) Performance level required?
private
T 4.B.i)
performance
blockchain features
T 4.A)
performance
cost efficiency
4.B.i) Which is the asset nature?
public
T 4.B.ii)
flexibility
robustness
no
yes
asset exchange
i)
yes
no
4.A) Is confidentiality required?
4.B) Which is the platform primary purpose?
ii)
automatization
4.B.ii) Is confidentiality required?
A. system of records
B. platform
4) Blockchain primary adoption?
START
Which
questions
direct answers
starting points for deciding when to use blockchain and which blockchain to adopt
trade-off points
preference path related to a trade-off criterion
decision path

Figure 5.19 – Quand utiliser une blockchain, et quel type, au lieu d'adopter un système de base de données traditionnel. Les cercles rouges représentent les points de trade-off entre les aspects cruciaux pour les différents cas d'usage de la blockchain. Les flèches rouges indiquent la conséquence de la priorité donnée à un aspect plutôt qu'à un autre, tandis que les flèches noires signalent les réponses à toutes les questions - accompagnées d'un ordre - de toute personne intéressée par la technologie blockchain.

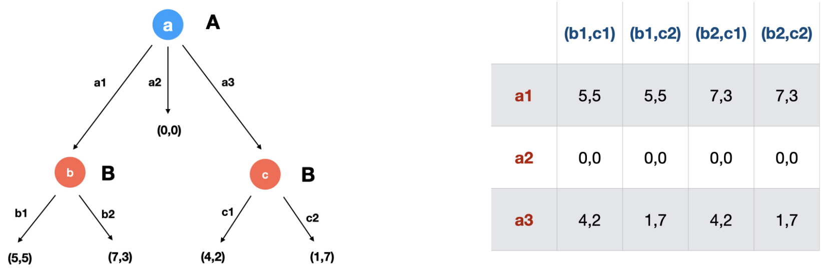
Chapitre 2

Ce chapitre fournit une introduction à la théorie des jeux, une branche de la théorie de la décision qui considère les décideurs comme des agents rationnels. La rationalité des joueurs prenant part à un jeu permet de prévoir leurs actions et donc le résultat du jeu. Les chapitres suivants présentent des problèmes réels dans le contexte de la blockchain que nous modélisons et analysons comme des jeux.

Généralement, plusieurs situations peuvent être modélisées comme des jeux, ce qui conduit à une classification des jeux. Les deux principales classes de jeux sont : les *jeux non coopératifs* et les *jeux coopératifs*. Dans cette thèse, nous traitons les deux classes de jeux : les jeux dans lesquels les joueurs individuels sont en compétition les uns avec les autres (i.e., non coopératifs) et les jeux dans lesquels il existe la possibilité de coopérer. Un *jeu* consiste en un processus interactif de prise de décision entre plusieurs décideurs appelés *joueurs*. Tout ensemble d'actions disponibles pour un joueur dans un jeu, en fonction de la structure du jeu, identifie une *stratégie*. Le *payoff* est le gain quantifiable (selon une fonction d'utilité) qu'un joueur reçoit en fonction du résultat du jeu. La théorie des jeux repose sur l'hypothèse fondamentale de la ***rationalité des agents***. La rationalité implique que (i) chaque joueur est capable **d'ordonner les résultats du jeu en fonction de préférences cohérentes** et que (ii) les joueurs sont pleinement capables **d'analyser le problème (i.e., le jeu) et de prendre les bonnes décisions**.

Jeux non coopératifs

Les jeux finis peuvent être représentés par un *arbre* ou *forme extensive* qui fournit des informations sur le cadre initial, toutes les évolutions possibles et les résultats finaux du jeu (Fig. 5.20 - gch). Cette représentation permet de visualiser un ensemble de joueurs qui agissent en séquence. Chaque jeu sous forme extensive peut être réécrit de manière plus compacte, avec *forme normale* (Fig. 5.20 - dte).



	(b1,c1)	(b1,c2)	(b2,c1)	(b2,c2)
a1	5,5	5,5	7,3	7,3
a2	0,0	0,0	0,0	0,0
a3	4,2	1,7	4,2	1,7

FIGURE 5.20 – Jeux représenté en forme extensive (à gauche) et normale (à droite)

Definition 5.16 (Jeux en forme normale) *Un jeu en forme normale est identifié par un tuple* $\Gamma = \langle N, \mathcal{S}, u \rangle$*, où* N *est un ensemble fini de* n *joueurs,* $\mathcal{S} = \mathcal{S}_1 \times \mathcal{S}_2 \times \cdots \times \mathcal{S}_n$ *où* $\mathcal{S}_i$ *est l'ensemble des stratégies du joueur* i *et* $u : \mathcal{S} \to \mathbb{R}^n$ *est la fonction d'utilité des joueurs.*

En supposant que chaque joueur choisisse une stratégie σ_i $dans \mathcal{S}_i$; il est alors possible de calculer l'utilité d'un joueur i, $u_i(\sigma_1, \sigma_2, \ldots, \sigma_n)$, qui est la i-ième composante de la fonction d'utilité u. Étant donné que les joueurs sont des agents rationnels, leur objectif est de maximiser leur utilité en choisissant leur propre stratégie.

Une **solution** est un profil de stratégie qui décrit la dynamique et le résultat du jeu. Von Neumann et Morgenstern [243] ont approfondi le concept de rationalité en fournissant une hypothèse supplémentaire : l'**élimination des stratégies dominées**. Cette méthode suppose que, indépendamment du choix que font les autres joueurs, un joueur ne choisit pas l'alternative A, si B est à sa disposition lui permettant d'obtenir plus. En itérant le processus d'élimination des stratégies dominées, un jeu de forme normale peut être résolu. Si une stratégie strictement dominante existe pour un joueur dans un jeu, ce joueur jouera cette stratégie. Si les deux joueurs ont une stratégie strictement dominante, le jeu n'a qu'une seule solution équilibre, appelée *équilibre de Nash*. Les équilibres de Nash sont des concepts de solution raisonnables puisqu'ils représentent un scénario dans lequel personne n'est tenté de changer unilatéralement sa propre stratégie.

Definition 5.17 (Équilibre de Nash) *Un profil de stratégie* $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_i, \ldots, \sigma_n)\mathcal{S}$ *est un équilibre de Nash si pour chaque joueur* i *et pour chaque* $\tau_i \in S_i$ *nous avons que :*

$$u_i(\sigma_1, \sigma_2, \ldots, \sigma_i, \ldots, \sigma_n) \geq u_i(\sigma_1, \sigma_2, \ldots, \tau_i, \ldots, \sigma_n)$$

Jeux coopératifs

Afin de caractériser la théorie des jeux coopératifs, il est crucial d'identifier un ensemble de joueurs et de définir une fonction caractéristique pour les jeux identifiant comment les agents coopératifs partagent quelque chose au sein de leur coalition.

Definition 5.18 (Jeux coopératifs) *Un jeu coopératif est défini par la paire* (N, v) *où* N *est l'ensemble des joueurs et* $v : 2^N \to \mathbb{R}$ *est la fonction caractéristique qui attribue une valeur* $v(S)$ *à chaque coalition* $S \subset N$*.*

Chaque fois que le montant $v(S)$ peut être librement divisé entre les membres de S nous avons que (N, v) est un jeu à Utilité Transférable (TU). La question qui se pose maintenant est la suivante : *comment cette valeur est-elle partagée ?*, il est maintenant temps d'introduire l'idée de solution. Dans un jeu (N, v), une solution est un vecteur $(x_1, ..., x_n)$ où x_i représente le montant attribué au joueur i. Un concept de solution stable pour les jeux coopératifs est le suivant.

Definition 5.19 (Le noyau) *Soit $v : 2^N \to \mathbb{R}$ un jeu TU. Le noyau du jeu, désigné par $\mathcal{C}(v)$, est l'ensemble $\mathcal{C}(v) = \{x \in \mathbb{R}^n : \sum\limits_{i \in N} x_i = v(N) \wedge \sum\limits_{i \in S} x_i \geq v(S), \forall S \subset N\}$.*

Les solutions du noyau garantissent que les joueurs ne s'opposent pas puisqu'ils n'obtiennent pas moins que ce qu'ils peuvent obtenir seuls.

Jeux de banqueroute

Une classe particulière de jeu coopératif que nous adoptons dans le chapitre suivant est celle des jeux de banqueroute. Une situation de banqueroute se présente chaque fois que des agents revendiquent un certain montant d'une masse divisible et que la somme des revendications est supérieure à la masse . Formellement, une *situation de banqueroute* sur l'ensemble N des agents consiste en une paire $(\mathbf{c}, E) \in \mathbb{R}^N \times \mathbb{R}$ avec $c_i \geq 0$ pour tout $i \in N$ et $0 < E < \sum_{i \in N} c_i = C$ ou $\mathbf{c}$ représente les demandes des agents et E est la masse qui doit être divisé entre les agents. Trois solutions bien connues pour les jeux de banqueroute seront analysées lors de la modélisation des problèmes blockchain.

Definition 5.20 (Règles) *Pour chaque situation de banqueroute $(\mathbf{c}, E) \in \mathbb{B}^N$, la règle*

- ***Proportional (P)** donne le vecteur d'allocation $P(\mathbf{c}, E) = \pi \mathbf{c}$, où π est tel que $\sum_{i \in N} \pi c_i = E$.*
- ***Constrained equal awards (CEA)** est définie comme $CEA_i(\mathbf{c}, E) = \min\{c_i, \lambda\}$ où le paramètre λ est tel que $\sum_{i \in N} \min\{c_i, \lambda\} = E$.*
- ***Constrained equal losses (CEL)** est définie comme $CEL_i(\mathbf{c}, E) = \max\{c_i - \lambda', 0\}$ où le paramètre λ est tel que $\sum_{i \in N} \max\{c_i - \lambda', 0\} = E$.*

La règle CEA consiste à donner à chaque agent le même montant jusqu'à ce que sa demande ne soit pas entièrement satisfaite et que la masse ne soit pas terminée. En revanche, au lieu d'égaliser les revendications, la règle CEL égalise les pertes en privilégiant les grosses demandes par rapport aux petites. Les trois règles sont dans le noyau des jeux banqueroute.

Chapitre 3

Ce chapitre modélise et analyse les comportements d'un type particulier de nœuds de validation opérant sur une DLT spécifique, à savoir les mineurs de Bitcoin. Les mineurs qui effectuent des tâches de validation dans la blockchain Bitcoin sont modélisés comme des agents rationnels qui peuvent ou non adopter un comportement malveillant dans certaines situations. La situation que nous présentons dans ce chapitre est le processus de rémunération des mineurs pour leur travail de validation. Deux types de comportements malveillants sont analysés : le *pool-hopping* et le *block-withholding.* La première partie du chapitre est consacrée à la description et à l'évaluation du phénomène du "pool-hopping", tandis que dans la deuxième partie, les mineurs qui retiennent (i.e., withhold) les blocs sont modélisés comme des joueurs d'un jeu de banqueroute.

Comme présenté dans le chapitre 1, les transactions blockchain sont rassemblées en blocs, validées et publiées sur le registre distribué. L'objectif des *mineurs* est de trouver une valeur numérique i.e., *nonce* qui, ajoutée à une chaîne de données d'entrée et "hachée" (i.e., donnée comme input à la fonction SHA256), produit un résultat inférieur à une valeur de seuil. Un mineur qui trouve une *solution complète* (i.e., un nonce répondant à l'objectif de difficulté) et le diffuse sur le réseau obtient une récompense consistant en crypto-monnaies émises dans le réseau via de *transactions coinbase.*

Le minage est une procédure par laquelle les mineurs peuvent gagner une somme d'argent substantielle. Aujourd'hui, les petits mineurs se font concurrence dans cette nouvelle industrie en rejoignant des pools de mineurs. Un pool est une approche coopérative dans laquelle plusieurs mineurs partagent leurs efforts (i.e., leur puissance de calcul) afin de valider des blocs et d'obtenir des récompenses. Lorsqu'une solution complète est trouvée, la récompense du pool est répartie entre les mineurs.

La récompense des mineurs est basée sur leur contribution à la recherche d'une solution complète. Afin de donner une preuve de leur travail, les mineurs soumettent au pool des solutions partielles, c'est-à-dire des nonces qui n'atteignent pas le seuil initial, mais un seuil plus élevé. Les solutions de ce crypto-puzzle plus facile sont considérées comme des solutions "proches de la validité" et sont appelées *shares.* Les mineurs sont récompensés en fonction du nombre de shares qu'ils fournissent. Lorsqu'un share est également une solution complète, le bloc est validé et le pool gagne une récompense qui est divisée entre les participants du pool en fonction du nombre de shares qu'ils ont déclarés. Chaque pool adopte son propre système de récompense.

Une **attaque** contre un pool désigne le comportement d'un mineur qui diffère de la pratique par défaut (la pratique honnête) et qui met en péril le bien-être collectif du pool. Les mineurs attaquent un pool afin d'obtenir une récompense plus élevée i.e., les mineurs se comportent comme des agents rationnels dont les décisions sont motivées par le gain attendu. Les mineurs peuvent attaquer leur pool au moment de la déclaration de leur preuve de travail (share). Plus précisément, ils peuvent (i) retarder la déclaration du share (i.e., *block-withholding*) et/ou (ii) déclarer le share ailleurs (i.e., *pool-hopping*). Le premier attaque implique un retard de la validation d'un bloc et de la possession de la récompense associée. Le "pool-hopping" est une attaque où les mineurs "sautent" d'un pool à un autre en fonction de leur attractivité en terme de rémunération.

Pool-hopping et Stratégies de Rémunération

Les méthodes de récompense [256] paient généralement selon une division en *rounds*, où un round est le temps écoulé entre deux validations de blocs différentes effectuées par le même pool. Ainsi, le pool distribue la récompense du bloc entre les mineurs à la fin du round, en proportion de leur contribution au cours du round. Meni Rosenfeld montre dans [256] que les pools utilisant des systèmes proportionnels encouragent des pratique de pool-hopping dans le sens où le saut de pool est plus rentable que le minage en continu. Notre contribution [15] décrit et analyse le phénomène de pool-hopping en analysant les flux de transactions des pools et mineurs opérant dans la blockchain Bitcoin.

En partant des transactions de coinbase et de récompense, nous avons construit le **sous-réseau de transactions** [255] représentant le flux de bitcoins entre le pool et les mineurs au niveau de leur adresses. Dans un deuxième temps, nous avons mis en correspondance chaque acteur bitcoin avec les adresses qui lui sont associées afin de dériver un **sous-réseau d'utilisateurs** plausible où les utilisateurs d'intérêt sont les mineurs et le pool.

À partir du sous-réseau d'utilisateurs construit, il est possible de sélectionner les mineurs qui sont rémunérés par plusieurs pools. Dans le cas de deux pools, si le même mineur est présent dans deux transactions de récompense consécutives (ordonnées) appartenant l'une à un pool et l'autre à un autre, nous pouvons le déclarer comme un hopper. Il peut y avoir des situations douteuses lorsque le même mineur est trouvé dans deux transactions de récompense de différents pools qui ne sont pas directement consécutives ou encore certains pools n'associent pas exactement une transaction de rémunération à chaque round. Cette modélisation ne capture donc pas l'intégralité du phénomène.

Nous avons analysé le sous-réseau d'utilisateurs impliquant des mineurs interagissant avec les deux pools miniers les plus populaires en 2016 : Slush pool [258] et Kano pool [262]. Dans l'ensemble, 43 "pool-hoppers" ont été détectés au sein d'une population de 69 mineurs actifs dans les deux pools et d'une population globale approximative de 150 000 mineurs (Fig. 5.21). De plus, nous étions intéressés par la quantification de l'avantage du "hopping". La distribution des revenus des sauteurs sur la fenêtre temporelle est présentée dans la Fig. 5.22 avec des boxplots ; les deux derniers boxplots sur le côté droit couvrent tous les sauteurs ensemble et tous les mineurs statiques ; nous trouvons une récompense médiane de transaction 3 fois plus élevée pour les mineurs sauteurs que pour les statiques.

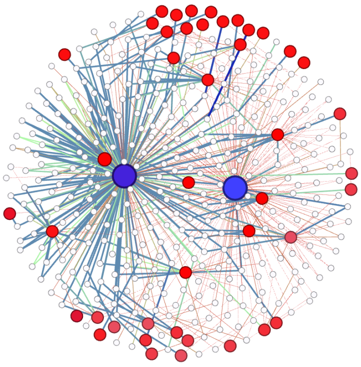


FIGURE 5.21 – Représentation graphique des pool-miners. Le nœud bleu de gauche est le pool Kano et le nœud bleu de droite est le pool Slush. Les hoppers détectés sont mis en évidence en rouge. L'épaisseur et la couleur des arcs dépendent de la fréquence des interactions des utilisateurs.

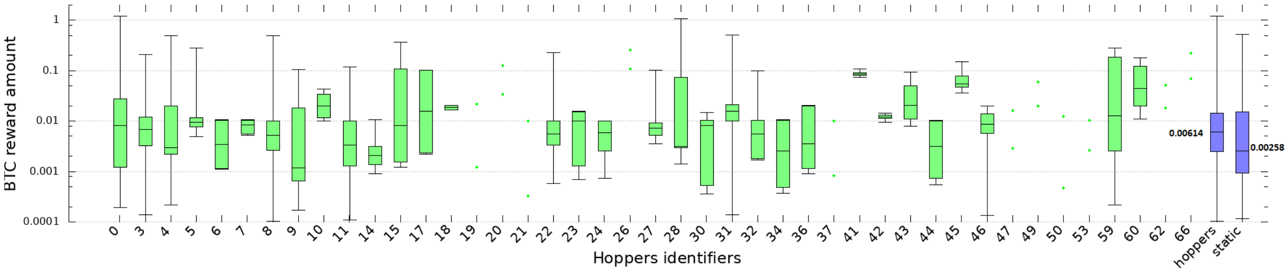


FIGURE 5.22 – Distributions par boxplot des valeurs bitcoin des transactions de récompense (échelle logarithmique).

Récompenser les mineurs rationnels : Situations de banqueroute

En suivant la logique introduite dans [253], nous proposons un mécanisme de récompense alternatif qui décourage les comportements de "pool-hopping" et de "block-withholding". En réinterprétant la fonction de récompense proposée dans [253] comme le résultat d'une situation de banqueroute, nous construisons, analysons et testons un nouveau mécanisme de récompense pouvant être adopté par les pools pour rémunérer les mineurs. Soit $N = \{1, \ldots, n\}$ un ensemble fini de mineurs. Le modèle que nous adoptons divise le temps en *rounds* et la récompense pour un bloc validé est fixée à $B = 1$. La situation est représentée par le vecteur $\mathbf{s} = (s_1, s_2, ..., s_n) \in \mathbb{N}^N$, défini comme *histoire des transcriptions*, qui contient le nombre de shares s_i déclarés par chaque mineur $i \in N$ dans un round. Une fonction de récompense $R : \mathbb{N}^N \to [0, 1]^n$, selon [253], est une fonction attribuant à chaque histoire des transcriptions $\mathbf{s}$ une allocation de la récompense $(R_1(\mathbf{s}), \ldots, R_n(\mathbf{s}))$, où R_i désigne la fraction de récompense obtenue par le mineur $i \in N$ et $\sum_{i \in N} R_i = B = 1$.

Dans [253], une fonction de récompense R est dite *incentive compatible* si la meilleure stratégie de chaque mineur est de déclarer immédiatement au pool un share ou une solution complète. Schrijvers et al. [253] introduisent un mécanisme de récompense qui satisfait la propriété de incentive compatibility en utilisant l'identité du découvreur de solution complète w. Étant donné un vecteur $\mathbf{e}^w = (e_1^w, \ldots, e_n^w) \in \{0, 1\}^N$ tel que $e_i^w = 0$ pour chaque $i \in N \setminus \{w\}$ et $e_w^w = 1$, la fonction de récompense incentive compatible R est la suivante :

$$R_i(\mathbf{s}; w) = \begin{cases} \frac{s_i}{D} + e_i^w \Big(1 - \frac{S}{D}\Big), & \text{if } S < D \\ \frac{s_i}{S}, & \text{if } S \geq D \end{cases} \qquad \forall i \in N, \tag{5.1}$$

où s_i est le nombre de shares déclarés par le mineur i, S est le nombre total de shares déclarés dans un round et D est la difficulté du puzzle cryptographique. On voit que la fonction de récompense R est la combinaison de deux méthodes d'allocation distinctes. Dans un *short round* (i.e., $S < D$), la fonction de récompense alloue un montant fixe par share à tous les agents, mais l'agent w qui trouve une solution est récompensé par un prix supplémentaire. Au contraire, dans un *long round* (i.e., $S \geq D$), la fonction de récompense alloue la récompense proportionnellement aux shares soumis. Dans les deux cas, $\sum_{i \in N} R_i(\mathbf{s}; w) = B = 1$. Rémunérer les mineurs par share, pour les longs rounds, conduirait à la banqueroute du pool puisque la récompense B résulte insuffisante à payer toutes les shares déclarés. Pour les longs rounds, le mécanisme de récompense proposé dans [253] n'est rien d'autre qu'une solution à une situation de banqueroute. Par conséquent, il est possible de créer de nouvelles fonctions de récompense en substituant simplement dans les long rounds différentes solutions de banqueroute. Nous pouvons donc construire une nouvelle fonction de récompense en substituant à l'allocation proportionnelle la règle CEL définie dans Def 2.13. Cette nouvelle règle devrait être robuste contre l'attaque Sybil et préserver la propriété de incentive compatibility. C'est pourquoi nous avons créé la nouvelle règle suivante :

Definition 5.21 *Étant donné l'identité du découvreur de la solution complète w, pour tous les $i \in N$ la fonction de récompense basée sur la règle CEL $\widehat{R}$ est définie comme suit :*

$$\widehat{R}_i(\mathbf{s}; w) = \begin{cases} \frac{s_i}{D} + e_i^w\left(1 - \frac{S}{D}\right), & \textit{if } S < D \\ \\ N\frac{e_i^w}{D} + \max\left(\frac{s_i}{D} - \lambda, 0\right),\ \lambda : \sum_i \max\left(\frac{s_i}{D} - \lambda, 0\right) = 1 - \frac{1}{D}, & \textit{if } S \geq D \end{cases},$$

où $\mathbf{e}^w = (e_1^w, \ldots, e_n^w) \in \{0,1\}^N$ est un vecteur tel que $e_w^w = 1$ et $e_i^w = 0\ \forall i \in N \setminus \{w\}$, s_i est le nombre de shares déclarés par le mineur i, $S = \sum_{i \in N} s_i$ est le nombre total de shares déclarés dans un round et D est la difficulté du crypto-puzzle.

Nous attribuons à l'agent w, qui trouve la solution pendant un long round, un prix supplémentaire de $\frac{1}{D}$ à ajouter à l'allocation établie par la règle CEL classique pour la situation de banqueroute $(\mathbf{c}, E) = (\frac{1}{D} \cdot \mathbf{s}, 1 - \frac{e_i^w}{D})$, la masse étant réduite de $\frac{1}{D}$.

Après avoir créé une règle d'allocation empêchant les comportements de withholding, voyons ce qui se passe dans un environnement multi-pool où les mineurs peuvent faire du "pool-hopping". Généralement, lorsqu'un mineur i est rémunéré avec un fonction R, son intérêt à réaliser du pool-hopping peut être mesuré comme la différence entre (i) la récompense moyenne lors d'un "hop" $\mathbb{E}[R_i^*]$ et (ii) la récompense moyenne $\mathbb{E}[R_i]$ lorsqu'il travaille pour le pool : $\delta_{hop} := \mathbb{E}[R_i^*] - \mathbb{E}[R_i]$.

Proposition 5.6 *La fonction de récompense R proposée par Schrijvers et. al. donne toujours aux mineurs un intérêt positif $\delta_{hop} > 0$ à effectuer des pool hopping.*

Nous pouvons analyser l'intérêt à effectuer du "pool-hopping" en adoptant la règle $\hat{R}$ basée sur CEL, puis comparer les résultats avec ceux fournis par R. À cette fin, nous définissons et estimons les éléments suivants : (i) la récompense moyenne de R et $\widehat{R}$ dans un environnement honnête (i.e., α_i et β_i, respectivement) et (ii) la récompense moyenne donnée par un pool laissé par un hopping mineur (i.e., α_i^* et β_i^*, respectivement). Il est possible de comparer les intérêts $\delta_{\text{hop}}, \hat{\delta}_{\text{hop}}$ donnés par $R, \widehat{R}$ à travers ces variables puisque :

$$\hat{\delta}_{\text{hop}} \leq \delta_{\text{hop}} \Leftrightarrow \beta_i - \beta_i^* \geq \alpha_i - \alpha_i^*.$$

Nous avons prouvé que l'intérêt hopping pour la fonction de récompense basée sur CEL est plus faible par rapport à l'intérêt donné par R (i.e., $\beta_i - \beta_i^* \geq \alpha_i - \alpha_i^*$) avec la Proposition 3.5.

Chapitre 4

Ce chapitre est consacré à l'analyse (i) du problème du cross-chain swap distribué dans le contexte de la blockchain et (ii) des comportements des *parties contractantes* visant à échanger des actifs via de cross-chain swaps. Les échanges atomiques inter-chaînes entrent dans la catégorie de recherche dites TAST (i.e., Token Atomic Swap Technology) [273] visant à rendre les blockchains interopérables. La toute première solution d'échange atomique a été proposée pour Bitcoin par *Nolan* [5] en utilisant des *hash-time locked contracts* permettant des transferts d'actifs conditionnels. La recherche se concentre maintenant sur des protocoles de swap *hybrides*, avec des hash-locks centralisés, résultant plus efficaces. Il faut noter que les différents protocoles d'échange se distinguent essentiellement par les parties concernées. L'ensemble des participants au swap peut être composé uniquement des propriétaires des actifs (comme dans le protocole [6]) ou des propriétaires accompagnés d'un tiers de confiance (comme dans le protocole AC3TW [7]).

Nous proposons, dans cette contribution, un cadre théorique (avec la théorie des jeux) qui formalise un problème de swap et caractérise les protocoles de swap blockchain en séparant clairement la phase de **publication des contrats** et la phase de **réalisation des contrats**. Un protocole général de swap définit l'ensemble des transferts d'actifs et l'ordre dans lequel ils doivent être exécutés. Étant donné un ensemble d'actifs $\mathcal{A}$ et les propriétaires correspondants $\mathcal{O}$, un *swap* consiste en un échange de propriété d'actifs au sein de l'ensemble des propriétaires. Un problème de swap est défini comme un tuple $\langle \mathcal{A}, \mathcal{O}, b_0, b_*, (u_i)_{i \in \mathcal{O}} \rangle$ où : $b_0, b_* : \mathcal{A} \to \mathcal{O}$ sont les plans de propriété *originaux* et *désirés* – les deux surjectives – telles que $\forall a \in \mathcal{A},\ b_0(a) \neq b_*(a)$ et u_i est la fonction d'utilité du propriétaire $i \in \mathcal{O}$ sur les actifs dans $2^{\mathcal{A}}$ telle que $u_i(b_0^{-1}(i)) < u_i(b_*^{-1}(i))$. pour chaque $i \in \mathcal{O}$. Afin de définir formellement un protocole de swap blockchain, nous devons d'abord introduire la structure d'un *protocole d'échange décentralisé*, consistant en une suite de transferts d'actifs à réaliser.

Definition 5.22 (protocole d'échange décentralisé) *Soit* $\sigma = \{(A^k, O^k, X^k) : |A^k| \geq |O^k|\}_k$, $k \in \{1, \dots, t\}$, $t \in \mathbb{N} : t \leq m$ *une suite d'échanges où (i)* $A^k \subseteq \mathcal{A}$ *et* $O^k \subseteq \mathcal{O}$ *spécifient respectivement les actifs et les propriétaires impliqués dans l'échange à l'étape k et (ii)* $X^k : A^k \to O^k$ *(surjectif) spécifie le propriétaire* $X^k(a) \in O^k$ *de tout actif* $a \in A^K$ *à l'étape k. La suite* σ *définit un protocole d'échange décentralisé qui engendre une suite de plans* $b_1^\sigma, b_2^\sigma, \dots, b_t^\sigma : \mathcal{A} \to \mathcal{O}$ *telle que pour tout* $k \in \{1, 2, \dots, t\}$ *:* $b_k^\sigma(z) = b_{k-1}^\sigma(z),\ \forall z \in \mathcal{A} \setminus A^k$ *et* $b_k^\sigma(z) = X^k(z),\ \forall z \in A^k$ *(avec* $b_0^\sigma = b_0$*).*

La suite σ définit, pour chaque étape du protocole k, les actifs A^k dont la propriété doit être transférée et les nouveaux propriétaires O^k pour chaque actif. Cependant, une étape supplémentaire est nécessaire lorsque l'on envisage des parties contractantes qui ne se font pas confiance. Le protocole doit être construit de telle sorte que l'échange soit réalisé dans son intégralité ou qu'aucun transfert d'actifs ne soit engagé (i.e., tout ou rien). Afin d'empêcher les participants d'échanger en externe les actifs impliqués dans un swap, le protocole exige que les actifs soient verrouillés dans des transactions spécifiques avec des *hash-time locked contracts* (HTLC). Ces derniers doivent être correctement mis en œuvre, publiés et validés sur le réseau avant d'être réalisés. Une fois publié, la réalisation des transferts permet à chaque participant de racheter le ou les nouveaux actifs échangés. Un protocole d'échange de blockchain doit spécifier les phases de publication et d'engagement ainsi que les horaires dans lesquels les agents effectuent les opérations. Par conséquent, étant donné une suite d'opérations σ, un ensemble de propriétaires d'actifs $\mathcal{O}$ et une autorité de confiance centralisée τ, nous définissons une *fonction de décision* $F : \{1, \dots, t\} \longrightarrow \mathcal{O} \cup \{\tau\}$ comme un plan qui spécifie le ou les agents $F(k)$ responsables de la publication (ou de la réalisation) du transfert (A^k, O^k). Notez que le ou les agents appelés à décider du transfert peuvent être soit des propriétaires d'actifs (i.e., $F(k) \in \mathcal{O}$) soit des acteurs de confiance externes (i.e., $F(k) = \tau$). Une fonction de décision F_T est *effective* sur σ_T si et seulement si $F_T(k) = O^k$ pour tout $k \in \{1, \dots, t_T\}, t_T \in \mathbb{N} : t_T \leq m$ i.e., les joueurs ont le pouvoir d'accepter ou de refuser les actifs désirés.

Definition 5.23 (protocole de swap blockchain) *Un protocole de swap blockchain est défini par la paire* (σ_P, σ_T) *représentant un protocole de publication suivi par un protocole de réalisation où :*

- $\sigma_P = \{(A^j, O^j)\}_{j \in \{1,\dots,t_P\}}$, $t_P \in \mathbb{N} : t_P \leq m$, $A^j \subseteq \mathcal{A}$ *et* $O^j \subseteq \mathcal{O}$ *est une suite telle que* $\forall j \in \{1, \dots, t_P\}$, $O^j = \{o \in \mathcal{O} : o \in b_*(A^j) \vee o \in b_0(A^j)\}$ *et,*
- $\sigma_T = \{(A^k, O^k, X^k)\}_{k \in \{1,\dots,t_T\}}$ *est un protocole d'échange engendrant la suite de plans* $b_1^{\sigma_T}, \dots, b_{t_T}^{\sigma_T} : \mathcal{A} \to \mathcal{O}$ *selon la définition précèdente.*

Nous associons à σ_P, σ_T *la fonction de décision correspondante* F_P, F_T *définie ci-dessus.*

La formalisation fournie permet de capturer les protocoles de swap blockchain avec *publication et réalisation séquentiels* où les transferts de propriété sont publiés et réalisés selon un ordre temporel précis et les protocoles de swap avec *publication simultanée* où les transferts sont créés et propagés simultanément au réseau blockchain. Nous avons prouvé que (i) suivre un protocole de swap séquentiel caractérisé par une fonction de décision *effective* est un *équilibre parfait de sous-jeu en stratégies dominantes* et, que (ii) suivre un protocole de swap caractérisé par une publication concurrente et une réalisation *snap* (immédiate) est un équilibre de Nash.

Notre cadre théorique nous permet de caractériser les équilibres de deux protocoles représentatifs présentés respectivement dans [5] et [7]. Dans le cas du protocole proposé dans [5] et généralisé dans [6], suivre le protocole est l'unique équilibre de sous-jeu parfait (en stratégies dominantes i.e., toutes les autres stratégies offrent une utilité inférieure), alors que dans le cas du protocole proposé dans [7], suivre le protocole est un équilibre de Nash. Le jeu de forme extensive associé au premier protocole est représenté en Fig. 4.3. Les équilibres sont identifiés en adoptant le processus de *raisonnement rétrograde* (i.e., babackward induction) pour chaque étape de décision. En revanche, le jeu de forme normale associé au protocole AC3TW est représenté en Fig. 4.5 où les équilibres de Nash sont calculés en éliminant les stratégies dominées.

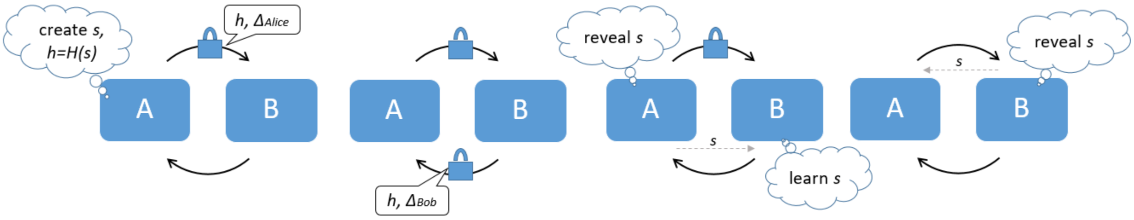


FIGURE 5.23 – Le protocole de swap proposé dans [5; 6] caractérisé par une publication et une réalisation séquentielles et formalisé comme suit peut être modélisé comme un jeu de forme extensive (i.e., un arbre). $\sigma_P = \{(x, B), (y, A)\}$ et $\sigma_T = \{(y, A), (x, B)\}$.

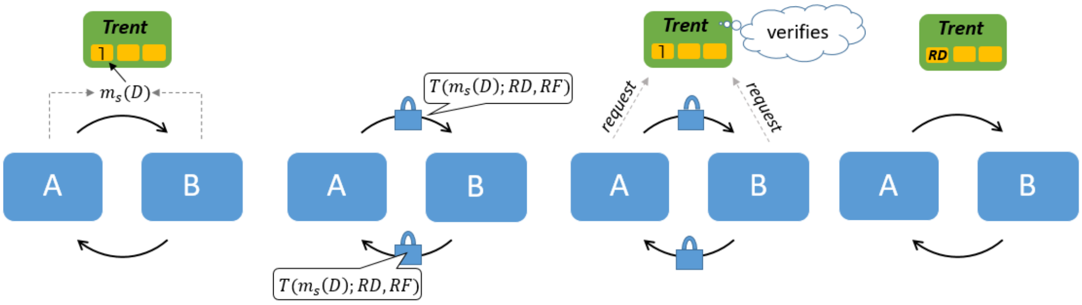


FIGURE 5.24 – Le protocole de swap AC3TW proposé dans [7] caractérisé par une publication simultanée et une réalisation instantanée et formalisé comme suit peut être modélisé comme un jeu de forme normale. $\sigma_P = \{(\{x, y\}, \{A, B\})\}$ and $\sigma_T = \{(\{x, y\}, \{A, B\}, \{X^1(x) = B, X^1(y) = A\})\}$.

Chapitre 5

Dans ce chapitre, nous proposons un cadre théorique, avec la théorie des jeux, qui caractérise formellement la **robustesse** des systèmes blockchain généraux en termes de *résilience aux déviations rationnelles* et d'*immunité aux comportements byzantins.* Les utilisateurs blockchain rationnels confrontés à un protocole blockchain à suivre peuvent agir de manière altruiste (en suivant le protocole) ou malveillante (en s'écartant du protocole) en fonction de leur utilité personnelle. En revanche, les utilisateurs byzantins s'écartent du protocole en toute situation (même en agissant de manière irrationnelle).

Modélisation de la Robustesse des Blockchains par la Théorie des Jeux

Le premier cadre générique d'analyse de la *robustesse des protocoles distribués* par rapport au comportement des joueurs rationnels et byzantins a été proposé par les auteurs de [309] qui ont introduit le concept de *mécanisme* (i.e., une paire jeu et stratégie prescrite).

Definition 5.24 (mécanisme [309]) *Un mécanisme est une paire* (Γ, σ) *dans laquelle* $\Gamma = \langle N, \mathcal{S}, u \rangle$ *est un jeu et* $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_i, \ldots, \sigma_n) \in \mathcal{S}$ *est un profil de stratégie.*

Il est conseillé à chaque joueur de jouer la stratégie $\sigma_i \in \mathcal{S}_i$. Le jeu Γ présente toutes les stratégies possibles à la disposition des joueurs. De plus, dans [309], les auteurs ont introduit les notions de (i) k**-résilience**, (ii) **praticité** et (iii) t**-immunité**. Un profil de stratégie est défini comme k*-résilient* s'il n'existe aucune coalition avec au plus k joueurs ayant une incitation à dévier du protocole prescrit. La catégorie des profils stratégiques *pratique* est définie lorsque les équilibres avec des stratégies faiblement dominées sont exclus. Dans ce chapitre, nous avons suivi la ligne de travail ouverte dans [309] et présenté un cadre théorique caractérisent la *robustesse des protocoles blockchain* aux comportements rationnels et byzantins des utilisateurs. Nos contributions en terme de modélisation de la robustesse peuvent être résumées comme suit :

a) Nous avons prouvé que la propriété d'immunité de t définie dans [309] n'est pas vérifiée par une grande classe de protocoles blockchain ;

b) Nous avons introduit le nouveau concept de t*-weak-immunity* ; un mécanisme est t-weak-immune si tout joueur altruiste ne reçoit pas de gain pire que l'état initial, quelle que soit la façon dont un ensemble de t joueurs s'écarte du protocole prescrit. Ce nouveau concept est suffisamment fort pour capturer la robustesse d'une grande classe de protocoles blockchain ;

Definition 5.25 (t-weak-immunity) *Un profil de stratégie* $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_i, \ldots, \sigma_n) \in \mathcal{S}$ *est t-weak-immune si pour tout* $T \subseteq N : |T| \leq t$*, tous les* τ_T *dans*$\mathcal{S}_T$ *et tous les* $i \in N \setminus T$*, on a* $u_i(\sigma_{-T}, \tau_T) \geq 0$*. Un mécanisme* (Γ, σ) *est t-weak-immune si* σ *est t-weak-immune dans le jeu* Γ*.*

c) Nous avons identifié et prouvé les conditions nécessaires et suffisantes pour qu'un mécanisme soit k-résilient et t-immunisé ;

d) Nous avons défini un nouvel opérateur pour la composition de jeux et prouvé qu'il préserve les propriétés de robustesse des jeux individuels.

Robustesse aux Comportements des Utilisateurs dans les protocoles *layer-1* et *layer-2*

À l'aide de notre cadre théorique et de l'opérateur de composition, nous avons étudié la robustesse aux comportements des deux principaux types d'utilisateurs blockchain : *parties contractantes* et *nœuds de validation*, les deux modélisés comme des agents rationnels et byzantins. Plus précisément, la robustesse aux comportements des validateurs est analysée pour Tendermint [1] et Bitcoin [2] (i.e., les protocoles layer-1). Pour les protocoles layer-2, (Lightning Network [3], le protocole side-chain [4] et le protocole cross-chain [5]), la robustesse est analysée en tenant compte des comportements des parties contractantes. Nos résultats sont présentés dans le tableau 5.2. Notre analyse nous a permis de repérer la faiblesse du protocole Lightning Network [3] face au comportement byzantin. Par conséquent, nous avons proposé et analysé plus en détail une version alternative du protocole.

TABLE 5.2 – Propriétés d'immunité et de résilience par rapport au nombre d'agents déviants rationnels (k) et au nombre d'agents déviants byzantins (t) où n est le nombre total de joueurs dans le jeu.

Protocol	k-résilience	t-immunité	t-weak immunity	Résultats
Tendermint	**Oui, k** $< n/3$	**Non**	**Oui, t** $< n/3$	Thm. 5.5
Bitcoin	**Oui, k** $< 3n/20$	**Non**	**Non**	Thm. 5.7
Lightning Network	**Oui, k** $< 3n/20$	**Non**	**Non**	Thm. 5.9
Closing module	Oui	Non	Non	Thm. 5.12
(Alternative closing module)	(Oui)	(Non)	(Oui)	Thm. 5.13
Other modules	Oui	Non	Oui	Thm. 5.10, 5.11, 5.15, 5.18, 5.19
Side-chain (Platypus)	**Oui, k** $< n/3$	**Non**	**Oui, t** $< n/3$	Thm. 5.20
Cross-chain Swap	**Oui**	**Non**	**Oui**	Thm. 5.23

Conclusion

Pour qu'une nouvelle technologie réalise tout son potentiel, il faut que de nombreuses circonstances coexistent avant que les effets de réseau puissent se concrétiser. En tant que technologie d'infrastructure, tous les principaux acteurs du marché doivent collaborer pour définir des normes de manière démocratique. Étant donné le coût du passage d'une technologie d'infrastructure à une autre, une approche collaborative est la voie la plus prometteuse. C'est la direction sur laquelle nous avons insisté dans la première partie de cette thèse et au cours de mon expérience de travail de trois ans au sein du "Programme Blockchain et Cryptoactifs" de Groupe Caisse des Dépôts [316].

Les contributions présentées dans la thèse analysent les différents comportements des utilisateurs blockchain en les considérant comme des agents rationnels, pleinement conscients de toutes les actions à leur disposition et capables de choisir celle qu'ils estiment être la meilleure pour eux-mêmes. Le manuscrit prend en considération différents utilisateurs ainsi que différentes blockchains.

Adoptant une approche agnostique, ce travail analyse les caractéristiques et attributs de la blockchain qui rendent les cas d'usage basés sur la blockchain plus performants ou plus innovants. Une revue de l'utilisation de la blockchain et du type de blockchain à utiliser est fournie dans le chapitre 1. Après cet aperçu, une analyse formelle est effectuée en combinant la théorie des systèmes distribués et la théorie des jeux (chapitre 2) afin d'analyser les comportements des utilisateurs opérant sur des systèmes de blockchain et de DLT. Ce mode de fonctionnement diffère sensiblement selon le type d'utilisateurs. Les contributions présentées dans les chapitres 3-5 analysent à la fois les utilisateurs qui adoptent simplement la technologie comme moyen de paiement/transfert (i.e., les parties qui effectuent des transactions) et les utilisateurs qui maintiennent le réseau en participant au processus de consensus (i.e., les nœuds de validation également appelés mineurs). Les limites de la rationalité sont prises en compte en analysant la robustesse de la blockchain aux comportements rationnels et byzantins (Chapitre 5).

Les questions de recherche ouvertes par les contributions présentées dans ce manuscrit concernent à la fois la modélisation avec la théorie de jeux (comportements rationnels et concepts de solution) et les types de blockchain analysés (comportements de blockchain et protocoles de consensus).

References

[1] "Tendermint Byzantine-fault tolerant state machine replication," accessed : 2021-10-10. [online] : http://tendermint.com.

[2] S. Nakamoto, "A peer-to-peer electronic cash system," 2008.

[3] J. Poon and T. Dryja, "The bitcoin lightning network : Scalable off-chain instant payments," 2016.

[4] A. R. Pedrosa and V. Gramoli, "Platypus : Offchain protocol without synchrony," in *18th IEEE International Symposium on Network Computing and Applications, NCA 2019, Cambridge, MA, USA, September 26-28, 2019*, A. Gkoulalas-Divanis, M. Marchetti, and D. R. Avresky, Eds. IEEE, 2019, pp. 1–8. [Online]. Available : https://doi.org/10.1109/NCA.2019.8935037

[5] T. Nolan, "Re : Alt chains and atomic transfers." accessed : 2021-10-10. [online] : https://bitcointalk.org/index.php?topic=193281.msg2224949#msg2224949.

[6] M. Herlihy, "Atomic cross-chain swaps," in *Proceedings of the 2018 ACM symposium on principles of distributed computing*, 2018, pp. 245–254.

[7] V. Zakhary, D. Agrawal, and A. Abbadi, "Atomic commitment across blockchains," *arXiv preprint arXiv :1905.02847*, 2019.

[8] Y. Amoussou-Guenou *et al.*, "Dissecting tendermint," in *Networked Systems*, M. F. Atig and A. A. Schwarzmann, Eds. Cham : Springer International Publishing, 2019, pp. 166–182.

[9] S. Popov, "The Tangle," iOTA White paper. Accessed : 2021-10-10. [online] : https://www.iota.org/research/academic-papers.

[10] “BigchainDB : a scalable blockchain database (White paper),” 2016, accessed : 2018-12-02. [online] : https://www.bigchaindb.com/whitepaper.

[11] E. Codd, “A relational model of data for large shared data banks. 1970,” *MD Comput*, vol. 15, pp. 162–166, 1998.

[12] R. Schollmeier, “A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications,” in *P2P’01*, 2001.

[13] A. S. Aiyer *et al.*, “Bar fault tolerance for cooperative services,” in *SOSP ’05*, 2005.

[14] M. Belotti *et al.*, “A vademecum on blockchain technologies : When, which, and how,” *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3796–3838, 2019.

[15] M. Belotti, S. Kirati, and S. Secci, “Bitcoin pool-hopping detection,” in *2018 IEEE 4th International Forum on Research and Technology for Society and Industry (RTSI).* IEEE, 2018, pp. 1–6.

[16] M. Belotti, S. Moretti, and P. Zappalà, “Rewarding miners : bankruptcy situations and pooling strategies,” in *Multi-Agent Systems and Agreement Technologies.* Springer, 2020, pp. 85–99.

[17] M. Belotti *et al.*, “Game theoretical analysis of atomic cross-chain swaps,” in *40th IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2020.

[18] P. Zappalà *et al.*, “Game theoretical framework for analyzing blockchains robustness,” in *International Symposium on Distributed Computing (DISC)*, 2021.

[19] R. Davenport, “Distributed database technology—a survey,” *Computer Networks*, vol. 2, no. 3, pp. 155–167, 1978.

[20] S. Nakamoto, “Bitcoin : A peer-to-peer electronic cash system,” Oct. 2008, accessed : 2021-10-10. [online] : https://bitcoin.org/bitcoin.pdf.

[21] M. Conti *et al.*, “A Survey on Security and Privacy Issues of Bitcoin,” *IEEE Communications Surveys Tutorials*, pp. 1–1, 2018.

[22] D. Bradbury, “The problem with Bitcoin,” *Computer Fraud & Security*, vol. 2013, no. 11, pp. 5–8, 2013.

[23] I. Eyal *et al.*, "Bitcoin-NG : A Scalable Blockchain Protocol." in *USENIX NSDI 2016.*

[24] "What to Mine," accessed : 2021-10-10. [online] : https://whattomine.com/coins.

[25] A. Wisniewska, "Altcoins," institute of Economic Research, Working Paper. Accessed : 2021-10-10. [online] : https://ideas.repec.org/p/pes/wpaper/2016no14.html.

[26] U. Chohan, "Are stable coins stable ?" *Notes on the 21st Century (CBRi)*, 2019.

[27] M. Khalilov and A. Levi, "A Survey on Anonymity and Privacy in Bitcoin-like Digital Cash Systems," *IEEE Communications Surveys & Tutorials*, 2018.

[28] M. Walport, "Distributed ledger technology : beyond blockchain," UK Government Office for Science, Tech. Rep., 2016.

[29] F. Tschorsch and B. Scheuermann, "Bitcoin and Be-yond : A Technical Survey on Decentralized Digital Currencies," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2084–2123, 2016.

[30] T. Neudecker and H. Hartenstein, "Network layer aspects of permissionless blockchains," *IEEE Communications Surveys & Tutorials*, 2018.

[31] U. Mukhopadhyay *et al.*, "A brief survey of Cryptocurrency systems," in *PST 2016.*

[32] L. Sankar, M. Sindhu, and M. Sethumadhavan, "Survey of consensus protocols on blockchain applications," in *ICACCS 2017.*

[33] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol : Analysis and applications," in *EUROCRYPT 2015.*

[34] Z. Zheng *et al.*, "Blockchain challenges and opportunities : A survey," *Working Paper*, 2016.

[35] ——, "An Overview of Blockchain Technology : Architecture, Consensus, and Future Trends," in *IEEE BigData Congress*, 2017.

[36] C. Cachin and M. Vukolić, "Blockchains Consensus Protocols in the Wild," *arXiv preprint arXiv :1707.01873*, 2017.

[37] S. Bano *et al.*, "Consensus in the Age of Blockchains," *arXiv preprint arXiv :1711.03936*, 2017.

[38] W. Wang *et al.*, "A survey on consensus mechanisms and mining management in blockchain networks," *arXiv preprint arXiv :1805.02707*, 2018.

[39] X. Li *et al.*, "A survey on the security of blockchain systems," *Future Generation ComputerSystems*, 2017.

[40] I. Lin and T. Liao, "A Survey of Blockchain Security Issues and Challenges." *IJ Network Security*, vol. 19, no. 5, pp. 653–659, 2017.

[41] "Survey on Blockchain Technologies and Related Services," FY2015 Technical report – Nomura Research Institute, accessed : 2021-10-10. [online] : http://www.meti.go.jp/english/press/2016/pdf/0531_01f.pdf.

[42] M. A. Ferrag *et al.*, "Blockchain Technologies for the Internet of Things : Research Issues and Challenges," *arXiv preprint arXiv :1806.09099*, 2018.

[43] N. Bozic, G. Pujolle, and S. Secci, "A tutorial on blockchain and applications to secure network control-planes," in *SCNS 2016*, pp. 1–8.

[44] W. Stallings, "A Blockchain Tutorial," *The Internet Protocol Journal*, vol. 20, no. 3, pp. 2–24, 2017.

[45] A. M. Antonopoulos, *Mastering Bitcoin, unlocking digital cryptocurrencies.* O'reilly Media, 2014.

[46] S. Haber and W. Stornetta, "How to time-stamp a digital document," in *Conference on the Theory and Application of Cryptography.* Springer, 1990, pp. 437–455.

[47] K. Saur *et al.*, "Technology for secure partitioning and updating of a distributed digital ledger," uS Patent Application Publication, Intel Corporation, CA (USA), 2016-11-18, US20180145836A1.

[48] V. Buterin, "On public and private blockchains," accessed : 2021-10-10. [online] : https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains.

[49] K. Wüst and A. Gervais, "Do you need a blockchain ?" *IACR Cryptology ePrint Archive*, vol. 2017, p. 375, 2017.

[50] R. Brown *et al.*, “Corda : An Introduction, White paper,” accessed : 2021-10-10. [online] : https://docs.corda.net/head/_static/corda-introductory-whitepaper.pdf.

[51] “Hyperledger Architecture Vol.1, Introduction to Hyp-erledger Business Blockchain Design Philosophy and Consensus,” accessed : 2021-10-10. [online] : https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger_Arch_WG_Paper_1_Consensus.pdf.

[52] R. Merkle, “Digital signature system and method based on a conventional encryption function,” Nov. 14, 1989, uS Patent 4,881,264.

[53] D. Johnson and A. Menezes, “Elliptic curve DSA (ECDSA) : an enhanced DSA,” in *USENIX Security Symposium*, vol. 7, 1998.

[54] D. Johnson, A. Menezes, and S. Vanstone, “The elliptic curve digital signature algorithm (ECDSA),” *Int. Journal of Information Security*, vol. 1, no. 1, pp. 36–63, 2001.

[55] D. Conte de Leon *et al.*, “Blockchain : properties and misconceptions,” *Asia Pacific Journal of Innovation and Entrepreneurship*, vol. 11, no. 3, pp. 286–300, 2017.

[56] Q. DuPont, “Experiments in algorithmic governance : A history and ethnography of “the dao”, a failed decentralized autonomous organization,” in *Bitcoin and Beyond.* Routledge, 2017, pp. 157–177.

[57] “Bitcoin Wiki. Transaction,” accessed : 2021-10-10. [online] : https://en.bitcoin.it/wiki/Transaction.

[58] V. Buterin, “Ethereum White-paper,” accessed : 2021-10-10. [online] : https:github.com/ethereum/wiki/wiki/White-Paper.

[59] J. Willet, “The Second Bitcoin Whitepaper, V. 0.5,” accessed : 2021-10-10. [online] : https://bravenewcoin.com/insights/the-second-bitcoin-whitepaper-vs--0-5.

[60] J. Bonneau *et al.*, “Research perspectives on bitcoin and second-generation cryptocurrencies,” in *IEEE Symposium on Security and Privacy. IEEE*, 2015.

[61] D. MacGregor, D. Mothersole, and J. Zolnowsky, “Method and apparatus for a compare and swap instruction,” Apr. 22, 1986, uS Patent number 4,584,640, NXP USA Inc.

[62] N. Atzei *et al.*, “SoK : unraveling Bitcoin smart con-tracts,” in *POST 2018*. Springer.

[63] G. Andresen, “BIP 16 : Pay to script hash,” 2012, accessed : 2021-10-10. [online] : https://github.com/bitcoin/bips/blob/master/bip-0016.mediawiki.

[64] “Bitcoin Wiki. Nonce,” accessed : 2021-10-10. [online] : https://en.bitcoin.it/wiki/Nonce.

[65] “Bitcoin Wiki. Script,” accessed : 2021-10-10. [online] : https://en.bitcoin.it/wiki/Script.

[66] N. Szabo, “The Idea of Smart Contracts.” accessed : 2021-10-10. [online] : https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/idea.html.

[67] “Chain Protocol - White Paper,” accessed : 2021-10-10. [online] : https://chain.com/docs/1.2/protocol/papers/whitepaper.

[68] P. Dai *et al.*, “Smart-Contract Value-Transfer Protocols on a Distributed Mobile Application Platform - White paper,” 2017.

[69] E. Androulaki *et al.*, “Hyperledger fabric : A distributed operating system for permissioned blockchains,” *arXiv preprint arXiv :1801.10228*, 2018.

[70] S. Popejoy, “The Pact Smart-Contract Language. White paper,” accessed : 2021-10-10. [online] : http://kadena.io/docs/Kadena-PactWhitepaper.pdf.

[71] C. Lee, “Litecoin-open source p2p digital currency,” accessed : 2021-10-10. [online] : https://github.com/coblee.

[72] D. Kuhnert, “The Dogecoin survival guide,” accessed : 2021-10-10. [online] : https://imgur.com/a/Sgyox.

[73] M. Green and I. Miers, “Bolt : Anonymous payment channels for decentralized currencies,” in *ACM CCS 2017*.

[74] G. Greenspan, ““MultiChain” Private Blockchain – White Paper,” accessed : 2021-10-10. [online] : https://www.multichain.com/download/MultiChain-White-Paper.pdf.

[75] L. Goodman, “Tezos – A self-amending crypto-ledger, White paper,” accessed : 2021-10-10. [online] : https://tezos.com/static/papers/white_paper.pdf.

[76] D. Schwartz *et al.*, “The Ripple protocol consensus algorithm – White Paper,” accessed : 2021-10-10. [online] : https://ripple.com/files/ripple_consensus_whitepaper.pdf.

[77] D. Mazieres, “The stellar consensus protocol : A fede-rated model for internet-level consensus - White paper,” accessed : 2021-10-10. [online] : https://www.stellar.org/papers/stellar-consensus-protocol.pdf.

[78] K. Olson *et al.*, “Sawtooth : An Introduction – White paper,” accessed : 2021-10-10. [online] : https://www.hyperledger.org/wp-content/uploads/2018/01/Hyperledger_Sawtooth_WhitePaper.pdf.

[79] A. Demers *et al.*, “Epidemic algorithms for replicated database maintenance,” in *ACM PODC 1987*, pp. 1–12.

[80] B. Wiki, “Weaknesses-Denial of Service (DoS),” accessed : 2021-10-10. [online] : https://en.bitcoin.it/wiki/Weaknesses.

[81] E. Heilman *et al.*, “Eclipse Attacks on Bitcoin’s Peer-to-Peer Network,” in *USENIX Security Symposium*, 2015.

[82] G. Fanti and P. Viswanath, “Deanonymization in the Bitcoin P2P Network,” in *NIPS 2017*, pp. 1364–1373. [Online]. Available : http://papers.nips.cc/paper/6735-deanonymization-in-the-bitcoin-p2p-network.pdf

[83] M. Babaioff *et al.*, “On bitcoin and red balloons,” in *ACM EC 2012*.

[84] J. Göbel *et al.*, “Bitcoin blockchain dynamics : The selfish-mine strategy in the presence of propagation delay,” *Performance Evaluation*, vol. 104, pp. 23–41, 2016.

[85] A. Gervais *et al.*, “On the security and performance of proof of work blockchains,” in *ACM CCS 2016*, pp. 3–16.

[86] L. Lamport, R. Shostak, and M. Pease, “The byzantine generals problem,” *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, pp. 382–401, 1982.

[87] T. Swanson, “Consensus-as-a-service : a brief report on the emergence of permissioned, distributed ledger systems,” R3 Technical Report, Apr. 2015, accessed : 2021-10-10. [online] : https://allquantor.at/blockchainbib/pdf/swanson2015consensus.pdf.

[88] A. Ellervee, R. Matulevicius, and N. Mayer, “A Comprehensive Reference Model for Blockchain-based Distributed Ledger Technology,” in *ER Forum 2017*.

[89] T. Dinh *et al.*, “Untangling Blockchain : A Data Processing View of Blockchain Systems,” *arXiv preprint arXiv :1708.05665*, 2017.

[90] P. Seijas, S. Thompson, and D. McAdams, “Scripting smart contracts for distributed ledger technology,” *IACR Cryptology ePrint Archive*, vol. 2016, p. 1156, 2016.

[91] T. Neudecker, P. Andelfinger, and H. Hartenstein, “Timing analysis for inferring the topology of the bitcoin peer-to-peer network,” in *Intl IEEE Conferences UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld*, 2016, pp. 358–367.

[92] M. Vukolić, “The quest for scalable blockchain fabric : Proof-of-work vs. bft replication,” in *Int. Workshop on Open Problems in Network Security*, 2015.

[93] A. Baliga, “Understanding Blockchain Consensus Models,” Persistent Systems – White paper, 2017, accessed : 2021-10-10. [online] : https://pdfs.semanticscholar.org/da8a/37b10bc1521a4d3de925d7ebc44bb606d740.pdf.

[94] S. Kiyomoto, M. Rahman, and A. Basu, “On blockchain-based anonymized dataset distribution platform,” in *IEEE SERA 2017*, June, pp. 85–92.

[95] “A Next-Generation Smart Contract and Decentralized Application Platform, Ethereum white paper,” 2014, [online] https://github.com/ethereum/wiki/wiki/White-Paper.

[96] C. Cachin, “Architecture of the Hyperledger blockch– ain Fabric,” in *DCCL 2016*.

[97] M. Hearn, “Corda : A distributed ledger,” white Paper, Accessed : 2021-10-10. [online] : https://docs.corda.net/head/_static/corda-technical-whitepaper.pdf.

[98] “Quorum White paper,” accessed : 2021-10-10. [online] : https://github.com/jpmorganchase/quorum-docs/blob/master/Quorum%20Whitepaper%20v0.1.pdf.

[99] S. Luan and V. Gligor, “A fault-tolerant protocol for atomic broadcast,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 1, no. 3, pp. 271–285, 1990.

[100] V. Hadzilacos and S. Toueg, “Fault-tolerant Broadcasts and Related Problems,” in *Distributed Systems (2Nd Ed.).* New York, NY, USA : ACM Press/Addison-Wesley Publishing Co., 1993, pp. 97–145. [Online]. Available : http://dl.acm.org/citation.cfm?id=302430.302435

[101] T. Chandra and S. Toueg, “Unreliable Failure Detectors for Reliable Distributed Systems,” *J. ACM*, vol. 43, no. 2, pp. 225–267, 1996.

[102] G. Coulouris, J. Dollimore, and T. Kindberg, *Distributed systems : concepts and design.* Pearson education, 2005.

[103] X. Défago, A. Schiper, and P. Urbán, “Total Order Broadcast and Multicast Algorithms : Taxonomy and Survey,” *ACM Comput. Surv.*, vol. 36, no. 4, pp. 372–421, 2004.

[104] I. Abraham *et al.*, “The Blockchain Consensus Layer and BFT,” *Bulletin of EATCS*, vol. 3, no. 123, 2017.

[105] M. Vukolić, “Rethinking Permissioned Blockchains,” in *ACM Workshop on Blockchain, Cryptocurrencies and Contracts*, 2017, pp. 3–7.

[106] A. Singh *et al.*, “BFT Protocols Under Fire,” in *USENIX NSDI 2008*, vol. 8, pp. 189–204.

[107] K. Christidis and M. Devetsikiotis, “Blockchains and Smart Contracts for the Internet of Things,” *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[108] D. Kreutz *et al.*, “Software-Defined Networking : A Comprehensive Survey,” *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[109] F. Cristian, “Understanding Fault-tolerant Distributed Systems,” *Communications of the ACM*, vol. 34, no. 2, pp. 56–78, 1991.

[110] C. Dwork, N. Lynch, and L. Stockmeyer, “Consensus in the presence of partial synchrony (preliminary version),” in *ACM PODC 1984*.

[111] ——, “Consensus in the presence of partial synchrony,” *Journal of the ACM (JACM)*, vol. 35, no. 2, pp. 288–323, 1988.

[112] M. Fischer, N. Lynch, and M. Paterson, "Impossibility of Distributed Consensus with One Faulty Process," *J. ACM*, vol. 32, no. 2, pp. 374–382, 1985.

[113] J. Aspnes, "Randomized protocols for asynchronous consensus," *Distributed Computing*, vol. 16, no. 2-3, pp. 165–175, 2003.

[114] C. Cachin, K. Kursawe, and V. Shoup, "Random Oracles in Constantipole : Practical Asynchronous Byzantine Agreement Using Cryptography," *Journal of Cry-ptology*, vol. 18, no. 3, pp. 219–246, 2005.

[115] E. Brewer, "Towards robust distributed systems," in *ACM PODC*, vol. 7, 2000.

[116] S. Gilbert and N. Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services," *SIGACT News*, vol. 33, no. 2, pp. 51–59, 2002.

[117] N. Lynch, M. Fischer, and R. Fowler, "A Simple and Efficient Byzantine Generals Algorithm," Georgia Inst of Tech school od information and computer science, Tech. Rep., 1982.

[118] M. Fischer and N. Lynch, "A lower bound for the time to assure interactive consistency," *Information processing letters*, vol. 14, no. 4, pp. 183–186, 1982.

[119] D. Dolev *et al.*, "An efficient algorithm for byzantine agreement without authentication," *Information and Control*, vol. 52, no. 3, pp. 257–274, 1982.

[120] I. Askoxylakis *et al.*, *Computer Security - ESORICS.* Springer, 2016.

[121] V. Buterin, "Ethereum News : On Stake," accessed : 2021-10-10. [online] : https://blog.ethereum.org/2014/07/05/stake.

[122] I. Abraham *et al.*, "Distributed computing meets game theory : robust mechanisms for rational secret sharing and multiparty computation," in *ACM PODC 2006*, pp. 53–62.

[123] W. Dai, "B-money (Blockchain)," [online :] : http://www.weidai.com/bmoney.txt .

[124] W. Ren, R. Beard, and E. Atkins, "A survey of consensus problems in multi-agent coordination," in *ACC 2005*, vol. 3, June, pp. 1859–1864.

[125] N. Lynch, *Distributed Algorithms.* San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1996.

[126] J. Gray, *Notes on data base operating systems.* Berlin, Heidelberg : Springer Berlin Heidelberg, 1978, pp. 393–481.

[127] F. Schneider, "Implementing Fault-tolerant Services Using the State Machine Approach : A Tutorial," *ACM Comput. Surv.*, vol. 22, no. 4, pp. 299–319, 1990.

[128] L. Lamport *et al.*, "Paxos made simple," *ACM Sigact News*, vol. 32, no. 4, pp. 18–25, 2001.

[129] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *USENIX Annual Technical Conference.*

[130] M. Pease, R. Shostak, and L. Lamport, "Reaching Agreement in the Presence of Faults," *J. ACM*, vol. 27, no. 2, pp. 228–234, 1980.

[131] R. Baldoni *et al.*, "Unconscious Eventual Consistency with Gossips," in *Symposium on Self-Stabilizing Systems*, 2006, pp. 65–81.

[132] B. Wiki, "Hashcash," accessed : 2021-10-10. [online] : https://en.bitcoin.it/wiki/Hashcash.

[133] J. Aspnes, C. Jackson, and A. Krishnamurthy, "Exposing computationally-challenged Byzantine impostors," TYALEU/DCS/TR-1332, Yale University, Tech. Rep., 2005.

[134] C. Dwork and M. Naor, "Pricing via processing or combatting junk mail," in *CRYPTO 1992*, pp. 139–147.

[135] A. Back, "Hashcash - a denial of service counter-measure," 2002.

[136] D. Eastlake and P. Jones, "US secure hash algorithm 1 (SHA1)," *RFC 3174, DOI 10.17487/RFC3174*, 2001.

[137] N. Szabo, "Bit Gold, 2008," accessed : 2021-10-10. [online] : http://unenumerated.blogspot.de/2005/12/bit-gold.html.

[138] H. Finney, "RPOW - Reusable PoW," accessed : 2021-10-10. [online] : http://cryptome.org/rpow.htm.

[139] I. Eyal, “The miner’s dilemma,” in *IEEE SP 2015*, pp. 89–103.

[140] L. Luu *et al.*, “Demystifying incentives in the consensus computer,” in *ACM CCS 2015*, pp. 706–719.

[141] B. Wiki, “Testnet,” accessed : 2021-10-10. [online] : https://en.bitcoin.it/wiki/Testnet.

[142] Y. Sompolinsky and A. Zohar, “Accelerating Bitcoin’s Transaction Processing Fast Money Grows on Trees, Not Chains,” accessed : 2021-10-10. [online] : https://pdfs.semanticscholar.org/4016/80ef12c04c247c50737b9114c169c660aab9.pdf.

[143] H. Okada, S. Yamasaki, and V. Bracamonte, “Proposed classification of blockchains based on authority and incentive dimensions,” in *ICACT 2017*, Feb, pp. 593–597.

[144] A. Gervais *et al.*, “Is Bitcoin a Decentralized Currency ?” *IEEE Security Privacy*, vol. 12, no. 3, pp. 54–60, 2014.

[145] B. Wiki, “Scrypt proof of work,” accessed : 2021-10-10. [online] : https://en.bitcoin.it/wiki/Scrypt_proof_of_work.

[146] C. Percival, “Stronger key derivation via sequential memory-hard functions,” *Self-published*, 2009, [Onli–ne] : http://www.bsdcan.org/2009/schedule/attachments/87_scrypt.pdf.

[147] J. Zhou, K. Yu, and B. Wu, “Parallel frequent patterns mining algorithm on GPU,” in *IEEE ICSMC 2010*, pp. 435–440.

[148] G. Pinto, F. Castor, and Y. Liu, “Mining questions ab-out software energy consumption,” in *ACM MSR 2014*, pp. 22–31.

[149] M. B. Taylor, “The Evolution of Bitcoin Hardware,” *Computer*, vol. 50, no. 9, pp. 58–66, 2017.

[150] S. King, “Primecoin : Cryptocurrency with prime number proof-of-work,” *Working paper*, 2013, accessed : 2021-10-10. [online] : http://primecoin.io/bin/primecoin-paper.pdf.

[151] J. Andersen and E. Weisstein, “Cunningham chain. from mathworld–a wolfram web resource,” 2005.

[152] A. Coventry, “NooShare : A decentralized ledger of shared computational resources,” Technical report, Apr. 2012, accessed : 2021-10-10. [online] : http://web.mit.edu/alex_c/www/noosharepdf.

[153] A. Shoker, “Sustainable blockchain through proof of exercise,” in *IEEE NCA 2017*, pp. 1–9.

[154] B. Marshall *et al.*, “Proofs of Work from Worst-Case Assumptions,” Cryptology ePrint Archive, Report 2018/559, 2018, accessed : 2021-10-10. [online] : https://eprint.iacr.org/2018/559.

[155] I. Bentov *et al.*, “Proof of Activity : Extending Bitcoin’s Proof of Work via Proof of Stake,” Cryptology ePrint Archive, Report 2014/452, 2014, accessed : 2021-10-10. [online] : https://eprint.iacr.org/2014/452.

[156] I. Bentov, R. Pass, and E. Shi, “Snow white : Provably secure proofs of stake.” *IACR Cryptology ePrint Archive*, vol. 2016, p. 919, 2016.

[157] A. Kiayias *et al.*, “Ouroboros : A provably secure proof-of-stake blockchain protocol,” in *CRYPTO 2017*, pp. 357–388.

[158] P. Singh *et al.*, “Performance Comparison of Executing Fast Transactions in Bitcoin Network Using Verifiable Code Execution,” in *ADCONS 2013*, Dec.

[159] Y. Amoussou-Guenou *et al.*, “Correctness and Fairness of Tendermint-core Blockchains,” *arXiv preprint arXiv :1805.08429*, 2018.

[160] S. King and S. Nadal, “Peercoin–secure & sustainable cryptocoin,” accessed : 2021-10-10. [online] : https://peercoin.net/whitepaper.

[161] A. Penzl *et al.*, “SNAPSHOT-Nxt unsurpassable blockchain solutions,” accessed : 2021-10-10. [online] : https://www.nxter.org/snapshot-nxt-unsurpassable-blockchain-solutions.

[162] L. Ren, “Proof of stake velocity : Building the social currency of the digital age,” Technical report, 2014, accessed : 2021-10-10. [online] : https://www.reddcoin.com/papers/PoSV.pdf.

[163] P. Vasin, “Blackcoin’s proof-of-stake protocol v2,” accessed : 2021-10-10. [online] : https ://blackcoin.co/blackcoin-pos-protocolv2-whitepaper.pdf,.

[164] “Novacoin,” accessed : 2021-10-10. [online] : https//:altcoinwiki.org/en/Novacoin.

[165] V. Buterin and V. Griffith, "Casper the friendly finality gadget," *arXiv preprint arXiv :1710.09437*, 2017.

[166] V. Buterin, "Understanding Serenity, part I : Abstraction," accessed : 2021-10-10. [online] : https://blog.ethereum.org/2015/12/24/understanding-serenity-part-i-abstraction.

[167] W. Li *et al.*, "Securing proof-of-stake blockchain protocols," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology.* Springer, 2017, pp. 297–315.

[168] A. Miller *et al.*, "Permacoin : Repurposing Bitcoin Work for Data Preservation," in *IEEE SP 2014*, pp. 475–490.

[169] S. P. *et al.*, "SpaceMint : A Cryptocurrency Based on Proofs of Space," Cryptology ePrint Archive – Report, 2015/528, accessed : 2021-10-10. [online] : https://eprint.iacr.org/2015/528.

[170] A. Haleem *et al.*, "Helium : A Decentralized Machine Network," accessed : 2021-10-10. [online] : http://whitepaper.helium.com/.

[171] "NEM - Technical Reference, NEM, Version 1.2.1," Tech. Rep., Feb 2018.

[172] D. Larimer, "Delegated proof-of-stake white paper," accessed : 2021-10-10. [online] : https://steemit.com/dpos/@dantheman/dpos-consensus-algorithm-this-missing-white-paper.

[173] C. Cachin, "Yet another visit to Paxos," *IBM Research, Zurich, Switzerland, Tech. Rep. RZ3754*, 2009.

[174] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *OSDI 1999.*

[175] C. Cachin, S. Schubert, and M. Vukolić, "Non-determinism in byzantine fault-tolerant replication," *arXiv preprint arXiv :1603.07351*, 2016.

[176] S. Liu *et al.*, "XFT : Practical Fault Tolerance beyond Crashes," in *OSDI 2016.*

[177] "NEO White Paper," accessed : 2021-10-10. [online] : http://docs.neo.org/en-us.

[178] "Proof of Authority," accessed : 2021-10-10. [online] : https://wiki.parity.io/Proof-of-Authority-Chains.

[179] "Aura-Authority Round," accessed : 2021-10-10. [online] : https://wiki.parity.io/Aura.html.

[180] "Clique PoA protocol," accessed : 2021-10-10. [online] : https://github.com/ethereum/EIPs/issues/225.

[181] K. Cong, "A Blockchain Consensus Protocol With Horizontal Scalability," *Master Thesis, Delft University of Technology*, 2017, accessed : 2021-10-10. [online] : https://infoscience.epfl.ch/record/232895.

[182] E. Kogias *et al.*, "Enhancing bitcoin security and performance with strong consistency via collective signing," in *USENIX Security 2016*, pp. 279–296.

[183] C. Decker, J. Seidel, and R. Wattenhofer, "Bitcoin meets strong consistency," in *ACM ICDCN 2016*, p. 13.

[184] E. Buchman, "Tendermint : Byzantine fault tolerance in the age of blockchains," Ph.D. dissertation, University of Guelph, 2016.

[185] I. Abraham *et al.*, "Solidus : An incentive-compatible cryptocurrency based on permissionless byzantine consensus," *arXiv preprint arXiv :1612.02916*, 2016.

[186] E. Kokoris *et al.*, "OmniLedger : A Secure, Scale-Out, Decentralized Ledger via Sharding," Cryptology ePrint Archive, Technical Report 2017/406, accessed : 2021-10-10. [online] : //https: eprint.iacr.//org/2017/406.

[187] Y. Gilad *et al.*, "Algorand : Scaling byzantine agreements for cryptocurrencies," in *ACM SOSP 2017*, pp. 51–68.

[188] "Slimcoin : A Peer-to-Peer Crypto-Currency with Proof-of-Burn," Technical report, 2014, accessed : 2021-10-10. [online] : http://www.doc.ic.ac.uk/˜ids/realdotdot/crypto_papers_etc_worth_reading/proof_of_burn/slimcoin_whitepaper.pdf.

[189] L. Bach, B. Mihaljevic, and M. Zagar, "Comparative analysis of blockchain consensus algorithms," in *IEEE MIPRO 2018.*

[190] T. Koens and E. Poll, "What blockchain alternative do you need ?" in *Data Privacy Management, Cryptocurrencies and Blockchain Technology.* Springer International Publishing, 2018, pp. 113–129.

[191] D. Yaga *et al.*, “Blockchain technology overview,” *Draft NISTIR*, vol. 8202, 2018.

[192] K. Wüst and A. Gervais, “Do you need a blockchain ?” in *2018 Crypto Valley Conference on Blockchain Technology (CVCBT).* IEEE, 2018, pp. 45–54.

[193] S. Pongnumkul *et al.*, “Performance Analysis of Private Blockchain Platforms in Varying Workloads,” in *ICCCN 2017*, July, pp. 1–6.

[194] G. Greenspan, “Blockchains vs centralized databases,” accessed : 2021-10-10. [online] : http://www.multichain.com/blog/2016/03/blockchains-vs-centralized-databases.

[195] ——, “Private blockchains are more than just shared databases,” accessed : 2021-10-10. [online] : https://www.multichain.com/blog/2015/10/private-blockchains-shared-databases.

[196] S. Ray, “Blockchains versus traditional databases,” accessed : 2021-10-10. [online] : https://hackernoon.com/blockchains-versus-traditional-databases-c1a728159f79.

[197] S. Goyal, “Centralized vs Decentralized vs Distributed,” accessed : 2021-10-10. [online] : https://medium.com.

[198] A. Narayanan, “Private blockchain is just a confusing name for a shared database,” accessed : 2021-10-10. [online] : https://freedom-to-tinker.com/2015/09/18/private-blockchain-is-just-a-confusing-name-for-a-shared-database.

[199] Y. Guo and C. Liang, “Blockchain application and outlook in the banking industry,” *Financial innovation*, vol. 2, no. 1, pp. 1–12, 2016.

[200] K. Fanning and D. Centers, “Blockchain and its coming impact on financial services,” *Journal of Corporate Accounting & Finance*, vol. 27, no. 5, pp. 53–57, 2016.

[201] B. Maurer, “Re-risking in realtime : on possible futures for finance after the blockchain,” *BEHEMOTH – A Journal on Civilisation*, vol. 9, no. 2, pp. 82–96, 2016.

[202] O. Bussmann, “The future of finance : Fintech, tech disruption, and orchestrating innovation,” in *Equity Markets in Transition.* Springer, 2017, pp. 473–486.

[203] A. Spielman, “Blockchain : digitally rebuilding the real estate industry,” Ph.D. dissertation, Massachusetts Institute of Technology, 2016.

[204] G. Zyskind *et al.*, "Decentralizing privacy : Using blockchain to protect personal data," in *2015 IEEE – Security and Privacy Workshops*, pp. 180–184.

[205] J. Mattila *et al.*, "Industrial Blockchain Platforms : An Exercise in Use Case Development in the Energy Industry," *ETLA Working Papers*, no. 43, 2016.

[206] F. Imbault *et al.*, "The green blockchain : Managing decentralized energy production and consumption," in *IEEE EEEIC/ICPS 2017*, June, pp. 1–5.

[207] E. Münsing, J. Mather, and S. Moura, "Blockchains for decentralized optimization of energy resources in microgrid networks," in *IEEE CCTA 2017*, Aug, pp. 2164–2171.

[208] N. Witchey, "Healthcare transaction validation via blockchain proof-of-work, systems and methods," May 13, 2015, uS Patent App. 14/711,740.

[209] X. Yue *et al.*, "Healthcare Data Gateways : Found Healthcare Intelligence on Blockchain with Novel Privacy Risk Control," *Journal of Medical Systems*, vol. 40, no. 10, p. 218, Aug. 2016.

[210] L. Linn and M. Koo, "Blockchain for health data and its potential use in health it and health care related research," in *ONC/NIST Use of Blockchain for Healthcare and Research Workshop*, 2016.

[211] A. Ekblaw *et al.*, "A Case Study for Blockchain in Healthcare : "MedRec" prototype for electronic health records and medical research data," in *IEEE Open & Big Data Conference*, 2016.

[212] M. Mettler, "Blockchain technology in healthcare : The revolution starts here," in *IEEE Healthcom 2016*, pp. 1–3.

[213] C. Broderson *et al.*, "Blockchain : Securing a New Health Interoperability Experience," *Accenture, Working paper*, 2016.

[214] K. Peterson *et al.*, "A Blockchain-Based Approach to Health Information Exchange Networks," *Working paper*, 2016.

[215] U. Sharma, "Blockchain in healthcare : Patient benefits and more," *IBM Blockchain Blog*, 2017.

[216] M. Orcutt, "Who will build the health-care blockchain," *Technology Review*, 2017.

[217] S. Huh, S. Cho, and S. Kim, “Managing iot devices using blockchain platform,” in *2017 19th international conference on advanced communication technology (ICACT).* IEEE, 2017, pp. 464–467.

[218] M. Samaniego, U. Jamsrandorj, and R. Deters, “Blockchain as a service for iot,” in *2016 IEEE international conference on internet of things (iThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData).* IEEE, 2016, pp. 433–436.

[219] D. Kravitz and J. Cooper, “Securing user identity and transactions symbiotically : IoT meets blockchain,” in *GIoTS 2017*, June, pp. 1–6.

[220] K. R. Özyılmaz and A. Yurdakul, “Work-in-progress : Integrating low-power iot devices to a blockchain-based infrastructure,” in *2017 International Conference on Embedded Software (EMSOFT).* IEEE, 2017, pp. 1–2.

[221] A. Hari and T. Lakshman, “The Internet Blockchain : A Distributed, Tamper-Resistant Transaction Framework for the Internet,” in *ACM HotNets*, 2016, pp. 204–210.

[222] N. Bozic, G. Pujolle, and S. Secci, “Securing virtual machine orchestration with blockchains,” in *CSNet 2017*, Oct, pp. 1–8.

[223] I. D. Alvarenga, G. A. F. Rebello, and O. C. M. B. Duarte, “Securing configuration management and migration of virtual network functions using blockchain,” in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, April 2018, pp. 1–9.

[224] D. Tse *et al.*, “Blockchain application in food supply information security,” in *2017 IEEE international conference on industrial engineering and engineering management (IEEM).* IEEE, 2017, pp. 1357–1361.

[225] F. Tian, “An agri-food supply chain traceability system for china based on rfid blockchain technology,” *ICSSSM*, 2016.

[226] M. Caro *et al.*, “Blockchain-based traceability in agri-food supply chain management : A practical implementation,” in *IoT Vertical and Topical Summit on Agriculture-Tuscany.* IEEE, 2018.

[227] Y. Yuan and F.-Y. Wang, “Towards blockchain-based intelligent transportation systems,” in *2016 IEEE 19th international conference on intelligent transportation systems (ITSC).* IEEE, 2016, pp. 2663–2668.

[228] L. Li *et al.*, “Creditcoin : A privacy-preserving blockchain-based incentive announcement network for communications of smart vehicles,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 7, pp. 2204–2220, 2018.

[229] C. Clack, V. Bakshi, and L. Braine, “Smart contract templates : foundations, design landscape and research directions,” *arXiv preprint arXiv :1608.00771*, 2016.

[230] X. Xu *et al.*, “A taxonomy of blockchain-based systems for architecture design,” in *IEEE ICSA 2017*, pp. 243–252.

[231] “MultiChain : open platform for building blockchains,” accessed : 2021-10-10. [online] : https://www.multichain.com.

[232] “Swarm,” accessed : 2021-10-10. [online] : https://swa rm-guide.readthedocs.io/en/latest/introduction.html.

[233] P. Labs, “Filecoin : A Decentralized Storage Network (White paper),” 2016, accessed : 2019-01-22. [online] : https://filecoin.io/filecoin.pdf.

[234] “Microsoft BaaS,” accessed : 2021-10-10. [online] : https://azure.microsoft.com/en/solutions/blockchain.

[235] “Types of tokens : the four mistakes beginner crypto-investors make,” accessed : 2021-10-10. [online] : https://medium.com/swlh/types-of-tokens-the-four-mistakes-beginner-crypto-investors-make-a76b53be5406.

[236] X. Xu *et al.*, “The blockchain as a software connector,” in *IEEE/IFIP WICSA 2016.*

[237] A. Back *et al.*, “Enabling blockchain innovations with pegged sidechains,” accessed : 2021-10-10. [online] : http://www.opensciencereview.com/papers/123/enablingblockchain-innovations-with-pegged-sidechains.

[238] R. Lucchetti, *A primer in game theory.* Società Editrice Esculapio, 2011.

[239] M. J. Osborne and A. Rubinstein, *A course in game theory.* MIT press, 1994.

[240] H. Peters, *Game theory : A Multi-leveled approach.* Springer, 2015.

[241] W. Thomson, “Axiomatic and game-theoretic analysis of bankruptcy and taxation problems : an update,” *Mathematical Social Sciences*, vol. 74, pp. 41–59, 2015.

[242] C. Herrero and A. Villar, “The three musketeers : four classical solutions to bankruptcy problems,” *Mathematical Social Sciences*, vol. 42, no. 3, pp. 307–328, 2001.

[243] J. Von Neumann and O. Morgenstern, “Theory of games and economic behavior, 2nd rev,” 1947.

[244] J. Nash, “Non-cooperative games,” *Annals of mathematics*, pp. 286–295, 1951.

[245] H. W. Kuhn and A. W. Tucker, *Contributions to the Theory of Games.* Princeton University Press, 1953, vol. 2.

[246] J. F. Nash *et al.*, “Equilibrium points in n-person games,” *Proceedings of the national academy of sciences*, vol. 36, no. 1, pp. 48–49, 1950.

[247] B. Bernheim, B. Peleg, and M. D. Whinston, “Coalition-proof nash equilibria i. concepts,” *Journal of Economic Theory*, vol. 42, no. 1, pp. 1 – 12, 1987. [Online]. Available : http://www.sciencedirect.com/science/article/pii/0022053187900998

[248] J. Hillas, “On the definition of the strategic stability of equilibria,” *Econometrica*, vol. 58, no. 6, pp. 1365–1390, 1990. [Online]. Available : http://www.jstor.org/stable/2938320

[249] E. Kohlberg and J.-F. Mertens, “On the strategic stability of equilibria,” *Econometrica : Journal of the Econometric Society*, pp. 1003–1037, 1986.

[250] H. Peters, *Game theory : A Multi-leveled approach.* Springer, 2015.

[251] I. J. Curiel, M. Maschler, and S. H. Tijs, “Bankruptcy games,” *Zeitschrift für operations research*, vol. 31, no. 5, pp. A143–A159, 1987.

[252] M. Rosenfeld, “Analysis of bitcoin pooled mining reward systems,” *arXiv preprint arXiv :1112.4980*, 2011.

[253] O. Schrijvers *et al.*, “Incentive compatibility of bitcoin mining pool reward functions,” in *International Conference on Financial Cryptography and Data Security.* Springer, 2016, pp. 477–498.

[254] Y. Lewenberg *et al.*, “Bitcoin mining pools : A cooperative game theoretic analysis,” in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems.* Citeseer, 2015, pp. 919–927.

[255] P. Koshy, D. Koshy, and P. McDaniel, “An analysis of anonymity in bitcoin using p2p network traffic,” in *International Conference on Financial Cryptography and Data Security.* Springer, 2014, pp. 469–485.

[256] M. Rosenfeld, “Analysis of bitcoin pooled mining reward systems,” *arXiv preprint arXiv :1112.4980*, 2011.

[257] ——, “Analysis of hashrate-based double spending,” *arXiv preprint arXiv :1402.2009*, 2014.

[258] “Reward system specification,” accessed : 2021-10-10. [online] : https://help.slushpool.com/en/support/solutions/articles/77000426280-reward-system-specification.

[259] “Optimal pool abuse strategy. Proofs and countermeasures,” accessed : 2021-10-10. [online] : https://bitcointalk.org/index.php?topic=3165.0.

[260] “Bitcoin Core,” accessed : 2021-10-10. [online] : https://bitcoin.org/en/bitcoin-core/.

[261] F. Reid and M. Harrigan, “An analysis of anonymity in the bitcoin system,” in *Security and privacy in social networks.* Springer, 2013, pp. 197–223.

[262] “KanoPool,” accessed : 2021-10-10. [online] : https://www.cryptocompare.com/mining/pools/kanopool/.

[263] “Pool Statistics,” accessed : 2021-10-10. [online] : https://btc.com/stats/pool/.

[264] E. Cortesi, “A new approach for bitcoin pool-hopping detection,” *Master Thesis*, 2021.

[265] M. Romiti *et al.*, “A deep dive into bitcoin mining pools : An empirical analysis of mining shares,” *arXiv preprint arXiv :1905.05999*, 2019.

[266] M. Saad *et al.*, “Exploring the attack surface of blockchain : A comprehensive survey,” *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1977–2008, 2020.

[267] Forbes, “Bitcoin Fork Suffers ‘Massive’ 51% Attack In Attempt To ‘Destroy’ The Cryptocurrency, Sending Its Price Sharply Lower,” accessed : 2021-10-10. [online] : https://www.forbes.com/sites/billybambrough/2021/08/04/bitcoin-fork-suffers-massive-51-attack-in-attempt-to-destroy-the-cryptocurrency-sending-its-price-sharply-lower/?sh=d0ef85d248f7.

[268] “Kyber : An On-Chain Liquidity Protocol,” accessed : 2021-10-10. [online] : https://files.kyber.network/Kyber_Protocol_22_April_v0.1.pdf.

[269] “Open Application Network,” accessed : 2021-10-10. [online] : https://github.com/aionnetwork.

[270] “Cosmos : A Network of Distributed Ledgers,” accessed : 2021-10-10. [online] : https://cosmos.network/cosmos-whitepaper.pdf.

[271] “Polkadot : Vision for a Heterogeneous Multi-Chain Framework,” accessed : 2021-10-10. [online] : https://polkadot.network/PolkaDotPaper.pdf.

[272] B. A. Lewis, P.M. and M. Kifer, *Databases and transaction processing : an application-oriented approach.* Addison-wesley Reading, 2002.

[273] M. Borkowski *et al.*, “Towards atomic cross-chain token transfers : State of the art and open questions within tast,” *Distributed Systems Group TU Wien (Technische Universit at Wien), Report*, 2018.

[274] “Etherdelta,” accessed : 2021-10-10. [online] : https://etherdelta.com/.

[275] W. Warren, “Front-running, griefing and the perils of virtual settlement,” accessed : 2021-10-10. [online] : https://blog.0xproject.com/front-running-griefing-and-theperils-of-virtual-settlement-part-1-8554ab283e97.

[276] “Decred cross-chain atomic swapping,” accessed : 2021-10-10. [online] : https://github.com/decred/atomicswap.

[277] “Bitcoin abc,” accessed : 2021-10-10. [online] : https://github.com/Bitcoin-ABC/bitcoin-abc.

[278] "Bitcoin unlimited," accessed : 2021-10-10. [online] : https://github.com/BitcoinUnlimited/BitcoinUnlimited.

[279] "Bitcoin xt," accessed : 2021-10-10. [online] : https://github.com/bitcoinxt/bitcoinxt.

[280] "Litecoin core integration/staging tree," accessed : 2021-10-10. [online] : https://github.com/litecoin-project/litecoin.

[281] "Qtum project," accessed : 2021-10-10. [online] : https://github.com/qtumproject/qtum.

[282] "Komodo barterdex," accessed : 2021-10-10. [online] : https://github.com/KomodoPlatform/BarterDEX.

[283] "Komodo (advanced blockchain technology, focused on freedom)," accessed : 2021-10-10. [online] : https://docs.komodoplatform.com/whitepaper/introduction.html.

[284] "Blockchain.io (your gateway to the internet of value)," accessed : 2021-10-10. [online] : https://blockchain.io/.

[285] E. Heilman, S. Lipmann, and S. Goldberg, "The arwen trading protocols," 2019.

[286] A. Zamyatin *et al.*, "Xclaim : Trustless, interoperable, cryptocurrency-backed assets," *IEEE Security and Privacy. IEEE*, 2019.

[287] M. Miraz and D. Donald, "Atomic cross-chain swaps : Development, trajectory and potential of non-monetary digital token swap facilities," *AETiC*, vol. 3, 2019.

[288] J. Zie *et al.*, "Extending atomic cross-chain swaps," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology.* Springer, 2019, pp. 219–229.

[289] B. Wiki, "Hashlock," accessed : 2021-10-10. [online] : https:/en.bitcoin.it/wiki/Hashlock.

[290] ——, "Timelock," accessed : 2021-10-10. [online] : https://en.bitcoin.it/wiki/Timelock.

[291] Coincer, "Atomic protocol n.1," accessed : 2021-10-10. [online] : https://www.coincer.org/2015/01/27/atomic-protocol-1/.

[292] ——, "Atomic protocol n.2," 2015, accessed : 2021-10-10. [online] : https://www.coincer.org/2015/02/03/atomic-protocol-2/.

[293] L. Harn, "Digital multisignature with distinguished signing authorities," *Electronics Letters*, vol. 35, no. 4, pp. 294–295, 1999.

[294] B. Wiki, "Distributed contract," accessed : 2021-10-10. [online] : https://en.bitcoin.it/wiki/Contract.

[295] T. Dickerson *et al.*, "Adding concurrency to smart contracts," in *Proceedings of the ACM Symposium on Principles of Distributed Computing.* ACM, 2017, pp. 303–312.

[296] L. Cong and Z. He, "Blockchain disruption and smart contracts," *The Review of Financial Studies*, vol. 32, no. 5, pp. 1754–1797, 2019.

[297] A. Kiayias and A.-P. Stouka, "Coalition-safe equilibria with virtual payoffs," *arXiv preprint arXiv :2001.00047*, 2019.

[298] W. Wang *et al.*, "A survey on consensus mechanisms and mining management in blockchain networks," *arXiv preprint arXiv :1805.02707*, pp. 1–33, 2018.

[299] I. Bentov *et al.*, "Tortoise and hares consensus : the meshcash framework for incentive-compatible, scalable cryptocurrencies." *IACR Cryptology ePrint Archive*, vol. 2017, p. 300, 2017.

[300] I. Eyal and E. G. Sirer, "Majority is not enough : Bitcoin mining is vulnerable," in *International conference on financial cryptography and data security.* Springer, 2014, pp. 436–454.

[301] I. Tsabary and I. Eyal, "The gap game," in *Proceedings of the 2018 ACM SIGSAC conference on Computer and Communications Security*, 2018, pp. 713–728.

[302] G. Avarikioti, E. K. Kogias, and R. Wattenhofer, "Brick : Asynchronous state channels," *arXiv preprint arXiv :1905.11360*, 2019.

[303] J. Y. Halpern and X. Vilaca, "Rational consensus," 2020.

[304] Z. Liu *et al.*, "A survey on blockchain : A game theoretical perspective," *IEEE Access*, vol. 7, pp. 47 615–47 643, 2019.

[305] T. Moscibroda, S. Schmid, and R. Wattenhofer, "When selfish meets evil : Byzantine players in a virus inoculation game," vol. 2006, 01 2006, pp. 35–44.

[306] E. Koutsoupias and C. Papadimitriou, "Worst-case equilibria," in *STACS 99*, C. Meinel and S. Tison, Eds. Berlin, Heidelberg : Springer Berlin Heidelberg, 1999, pp. 404–413.

[307] R. Pass and E. Shi, "Fruitchains : A fair blockchain," in *Proceedings of the ACM Symposium on Principles of Distributed Computing*, 2017, pp. 315–324.

[308] Y. Amoussou-Guenou *et al.*, "Rationals vs byzantines in consensus-based blockchains," *to appear AAMAS 2020*, vol. abs/1902.07895, 2019. [Online]. Available : http://arxiv.org/abs/1902.07895

[309] I. Abraham *et al.*, "Distributed computing meets game theory : Robust mechanisms for rational secret sharing and multiparty computation," in *Proceedings of the Twenty-Fifth Annual ACM Symposium on Principles of Distributed Computing*, ser. PODC '06. New York, NY, USA : Association for Computing Machinery, 2006, pp. 53–62. [Online]. Available : https://doi.org/10.1145/1146381.1146393

[310] R. Brenguier, "Robust equilibria in mean-payoff games," in *Foundations of Software Science and Computation Structures*, B. Jacobs and C. Löding, Eds. Berlin, Heidelberg : Springer Berlin Heidelberg, 2016, pp. 217–233.

[311] A. Chinchuluun *et al.*, *Pareto Optimality, Game Theory And Equilibria*, 01 2008, vol. 17.

[312] J. Von Neumann, O. Morgenstern, and H. W. Kuhn, *Theory of games and economic behavior (commemorative edition).* Princeton university press, 2007.

[313] P. Hammond, *Utility Invariance in Non-Cooperative Games*, 06 2006, vol. 38, pp. 31–50.

[314] R. Bellman, "Dynamic programming and lagrange multipliers," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 42, no. 10, p. 767, 1956.

[315] O. F. Inc. The on-line encyclopedia of integer sequences, year = 2021, url = https ://oeis.org/A178682, urldate = 2021-01-29.

[316] "Caisse des Dépôts Groupe," https://www.caissedesdepots.fr/.

[317] "Association pour le Développement des Actifs NumDériques," https://adan.eu/.

[318] "International Association for Trusted Blockchain Applications," https://inatba.org/.

[319] J. Chen and S. Micali, “Algorand : A secure and efficient distributed ledger,” *Theor. Comput. Sci.*, vol. 777, pp. 155–183, 2019. [Online]. Available : https://doi.org/10.1016/j.tcs.2019.02.001

[320] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, “Spectre : A fast and scalable cryptocurrency protocol.” *IACR Cryptol. ePrint Arch.*, vol. 2016, p. 1159, 2016.

[321] Y. Sompolinsky and A. Zohar, “Phantom,” *IACR Cryptology ePrint Archive, Report 2018/104*, 2018.

[322] S. Popov, O. Saa, and P. Finardi, “Equilibria in the tangle,” *Comput. Ind. Eng.*, vol. 136, pp. 160–172, 2019. [Online]. Available : https://doi.org/10.1016/j.cie.2019.07.025

Marianna Belotti

Game Theoretical Analysis of Blockchain Users' Behaviours

Abstract : The disruptive technology born in 2008 with Bitcoin and known as blockchain represents a significant quality leap from the distributed database technology. Distributed systems theory provides then models and techniques to analyze some protocols characterizing the technology, however in order to analyze a blockchain system additional considerations on its users need to be done. This thesis aims at analyzing the different behaviors of the users operating in blockchains or more in general in DLTs (i.e., Distributed Ledger Technologies). The latter are considered as rational agents, fully aware of all actions available to them and capable of choosing the one they feel is the best for themselves. Game theory is then used to model situations where users are called to choose and perform certain actions within the DLT environment. This thesis analyzes different users as well as different blockchains with the scope of providing a general overview on the topic and formal results on their behaviors; users may indeed be honest *vis-à-vis* of other users or they may behave maliciously (as Byzantine nodes) attacking the blockchain system.

Keywords : Blockchain, Consensus, Game Theory, Crypto-assets

Résumé : La technologie disruptive née en 2008 avec Bitcoin et connue sous le nom de blockchain représente un saut qualitatif important par rapport à la technologie des bases de données distribuées. La théorie des systèmes distribués fournit alors des modèles et des techniques pour analyser certains protocoles caractérisant la technologie, cependant afin d'analyser un système blockchain des considérations supplémentaires sur ses utilisateurs doivent être faites. Cette thèse vise à analyser les différents comportements des utilisateurs opérant dans les blockchains ou plus largement dans les DLTs (i.e., Distributed Ledger Technologies). Ces derniers sont considérés comme des agents rationnels, pleinement conscients de toutes les actions à leur disposition et capables de choisir celle qui leur semble la meilleure pour eux-mêmes. La théorie des jeux est alors utilisée pour modéliser des situations où les utilisateurs sont appelés à choisir et à effectuer certaines actions dans l'environnement DLT. Cette thèse analyse différents utilisateurs ainsi que différentes blockchains dans le but de fournir une vue d'ensemble sur le sujet et des résultats formels sur leurs comportements; les utilisateurs peuvent en effet être honnêtes vis-à-vis des autres utilisateurs ou se comporter de manière malveillante (comme des nœuds byzantins) en attaquant le système blockchain.

Mots-clés : Blockchain, Consensus, Théorie des Jeux, Crypto-actifs