POLITECNICO DI TORINO

Corso di Laurea
in Ingegneria Matematica

Tesi di Laurea

Dispute Resolution on Blockchain: from the Schelling Point to Aspera and Decentralised Mediation

Relatori
prof. Danilo Bazzanella
dott. Andrea Gangemi

Candidato
Giorgio Colantoni

Anno Accademico 2021-2022

† A mia sorella Eleonora

Summary

Blockchain is considered a revolutionary and potentially disruptive technology. Briefly, a blockchain is a distributed ledger among a network of computer nodes where data is stored in a transparent, immutable, and verifiable way to anyone through a consensus protocol. Decentralised consensus and security are achieved through the combination of cryptographic protocols with an economic incentive system.
Initially, applications for this technology were only done in financial field until the development of *Ethereum* platform, which is focused on *smart contracts.* A smart contract is a code that runs on top of a blockchain and executes itself when certain conditions are met. The use of smart contracts allows to stipulate deals between two or more anonymous parties, or to develop entire platforms or ecosystems within the blockchain.
The legal sector has shown interest in this technology: litigation through a national or international court requires long timelines not compatible with Internet reality, and high costs compared to the dispute's value. Besides, traditional systems are more vulnerable to corruption.
In contrast, the blockchain is potentially incorruptible because the several steps of litigation can be recorded transparently and are verifiable by parties involved. Smart contracts are efficient in managing the dispute resolution process and ensure the ruling enforcement. As a result, the total time and cost of litigation may be reduced.
Moreover, new classes of disputes have arisen; these latter cannot be resolved through traditional legal institutions due to the anonymity of the parties or the lack of clarity about the jurisdiction to be applied.
In this master thesis we have investigated the way in which some *Blockchain Dispute Resolution Platforms* handle the resolution process through the analysis of *Kleros*, *Aragon* and *Jur* protocols that adhere to arbitration models based on a *Schelling game* scheme, where disputes are settled by an anonymous group of jurors, who vote independently among a finite set of options, which are usually binary. A user must stake tokens from the specific platform in order to be drawn as juror. The plurality system establishes the outcome of a dispute. Thus, the option with most votes wins. Jurors who vote coherently with the majority win a reward, whereas others lose the stake.
According to these platforms, the economic incentive encourages users to vote for the fairest option by improving the quality of rulings compared to traditional methods. Arbitration may turn out not to be the best way to resolve a dispute.
Aspera is a project that aims to introduce *mediation* within the blockchain scenario. We propose two different protocols. The first one is based on the use of *Artificial Intelligence*

and *Machine Learning algorithms* in order to formulate mediation proposals.
The second one, defined as *Decentralised Mediation*, consists in reaching a solution to the dispute through a crowdsourcing mechanism, where platform users, known as *proposers*, submit at least one mediation proposal based on their idea of justice. Subsequently, the system draws a *chooser* who elects the mediation agreement perceived as the fairest one and sends it to the parties. Like arbitration protocols, a system of rewards and penalties has been designed in decentralised mediation to ensure the participants' honest behaviour in the resolution process. As a result, this mechanism should lead to multiple and high-quality solutions.

Contents

List of Tables

List of Figures

Chapter 1

Blockchain

1.1 The ancestor of blockchain technology: Bitcoin

Bitcoin is the first application of blockchain technology. The inventor Satoshi Nakamoto, whose identity is still unknown, published in 2008 the iconic whitepaper *"Bitcoin: a peer-to-peer electronic cash system"* after the failure of the Lehman Brothers Holding Inc., proposing a new paradigm that would replace the economic model based on trust in a central authority. The first block was created by Satoshi himself on January 3, 2009. This date represents a true revolution.
From a technical point of view, the term Bitcoin indicates the entire ecosystem including both the blockchain and the protocol that manages it. Since it is a payment system not bound to any institution, the protocol provides for the creation and circulation of a currency native to the blockchain. The latter is called bitcoin or BTC and it is a decentralised and anonymous cryptocurrency, whose value is exclusively determined by demand.

Satoshi succeeded in building an open, public and secure system, that is able to resolve the double-spending problem and where individuals all over the world can transfer money between them, without the presence of a central authority. The idea behind Bitcoin is to record transactions inside blocks sorted chronologically, that are connected to each other through the use of cryptographic functions. This ledger is also distributed among the users of a peer-to-peer network, referred as nodes, those who work to maintain and update it.
One of the key points of Bitcoin is the absence of a central authority which holds *"absolute truth"* about the state of transactions, so it is necessary to find a protocol that ensures distributed consensus between nodes when the chain is extended.

Satoshi suggests to use a *proof of work* system, close to the Adam Black's *Hashcash.* Nodes compete with each other to insert the new block through the resolution of a mathematical problem. The latter must be difficult to solve and simultaneously it must be easy to verify. The problem consists in the generation of pseudo-random values using a special

class of cryptographic functions, called hash functions, and this value must comply with certain conditions. For this reason, a miner is forced to make a lot of attempts in order to find this special value, and the one that is able to make more of them in less time, which means the one who has more computing power, has a higher probability of success. In addition, the protocol provides an economic incentive for those dealing with the insertion of new blocks.
In particular, each new block generates money through a special transaction, referred as *coinbase*, to the miner that attacked it. Another source of earning is the presence of transaction fees that users pay to have their transactions recorded on the blockchain. The higher the fee, the lower the waiting time for the formalization of the positive outcome of the transaction.
The success of Bitcoin and its revolutionary trustless structure inspired the creation of hundreds of new cryptocurrencies around the net.

1.2 The advent of blockchain 2.0: Ethereum

The development of *Ethereum* represents a second revolution that led to the birth of the *"Blockchain 2.0"* ecosystem. Bitcoin showed how to solve the problems of decentralised consensus and double-spending in a hostile environment without having to rely on a central authority, thus showing the potential of this technology, but it still remains a payment management tool. On the contrary, Ethereum presents itself as a general-purpose blockchain, where nodes are allowed to provide their computing power to run *smart contracts* and *decentralised applications* (*dApps*) in the *Ethereum Virtual Machine* (*EVM*). The idea of Ethereum's founder Vitalik Buterin is to build a global super-computer such that the entire network and all the changes in status are recorded within the blockchain in order to ensure security and immutability. Through this blockchain, it is possible to extend applications in sectors other than financial. An example is the development of *decentralised justice*, which is the main topic of this Master's thesis. Differently from Bitcoin, Ethereum adopts a Turing-complete scripting language to facilitate the writing of smart contracts.
A consequence of this choice is the risk of suffering attacks due to the creation of endless loops that can congestion the entire blockchain. The elegant solution proposed by Buterin is the introduction of gas. Each operation has its own precise computational cost in gas. The greater the complexity of the calculation, the greater the amount of gas required for its execution. The latter is paid through *Ether* (*ETH*), which is the native cryptocurrency of the Ethereum blockchain. Ethereum was designed to implement a proof of work consensus protocol which was replaced on 15 September 2022 with a *proof of stake* one through a process called *The Merge.*

1.3 Distributed Ledger Technology

The first step to understand blockchain is introducing the definition of Distributed Ledger Technology, or DLT. A DLT is a register of transactions that is shared, replicated, and synchronized in a distributed and decentralised manner in multiple places, called nodes. At the basis of this technology there is the idea of decentralisation. In fact, a DLT system wants to create a network where data are not stored in central data stores and managed by a few number of entities, but each node holds a copy of the data structure and periodically has to update it. Furthermore, nodes have equal rights and all decisions are made collectively. Another feature is data transparency, which allows each user to verify recorded transactions. This technology also brings the following benefits:

- a DLT system is more resistant to cyber attacks, because an attacker, or a group of them, has not a single point to violate, but a large number of nodes. Therefore, an attack requires enormous computing power: this point will be resumed later with the introduction of 51% attack.
- the creation of many copies and the transparency of the data between nodes guarantee the immutability of the latter. In fact, even in case of a successful attack on a node with the manipulation of its data, the others can compare and verify that the data of that node does not coincide with the whole network, identifying the attack occurred and correcting it.
- the presence of a large number of copies in the network prevents the loss of data due to malfunctions or structural damage of a central database.
- when data are managed by a single identity, there is always the risk of manipulation. Transparency and distribution guarantee data consistency.

A blockchain is a particular type of a DLT, so it owns all the properties listed above.

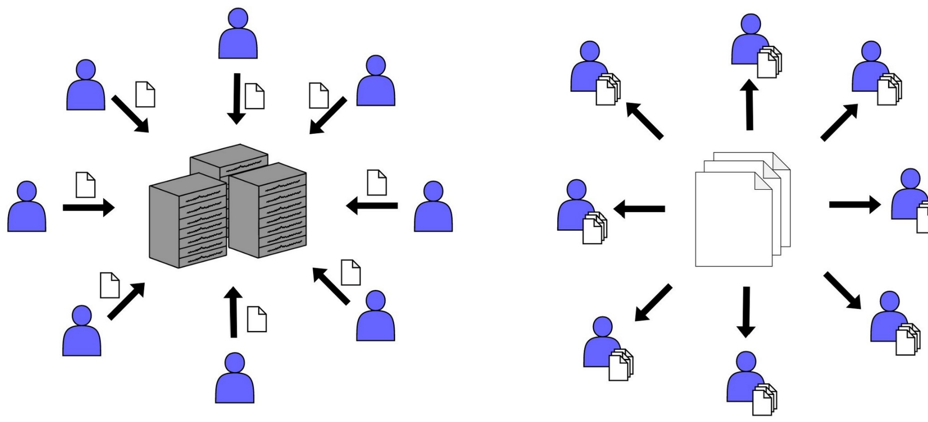


Figure 1.1. The figure represents the designs of a centralised (left) and decentralised (right) database. In the former, users collect their data in one location, while in the latter several copies are produced and distributed to different entities.

1.4 Cryptographic tools

The second step to better understand the protocols and the structure of a blockchain is to introduce cryptographic tools on which it is based.

1.4.1 Hash function

Definition 1 *Given an alphabet Σ and the set Σ^*, containing words of arbitrary length composed with elements of Σ, a hash function is a function*

$$h : \Sigma^* \longrightarrow \Sigma^n$$

where n is fixed.

A hash function receives as an argument a string of arbitrary length and returns a fixed-size string, that is called *digest* or *hash value*. In addition, these functions must be deterministic. Once we fix a hash function, the same input always generates the same hash value. Generally, for applications we use $\Sigma = \{0,1\}$ and $n = 160, 256, 384$ or 512. By definition, this class of functions can't be injective because the input set is much bigger then the output set. In cryptographic applications, an hash function should have the following properties:

- *pre-image resistance* or *one-way property*: a hash function h is one-way if given an input x, it is computationally efficient to calculate the output $h(x)$, but given an image y it is computationally inefficient to find the input x such that $h(x) = y$;

- *Collision resistance*: a cryptographic collision is a pair $(x_1, x_2) \in \Sigma^*$, with $x_1 \neq x_2$, such that $h(x_1) = h(x_2)$. A hash function h is collision resistant if it is computationally inefficient to find a collision. Notice that collisions always exist because hash functions are not injective.
- *Second pre-image resistance* or *weak collision*: given a pair $(x_1,\ h(x_1))$, it is computationally inefficient to find another input x_2 such that:
$$h(x_2) = h(x_1).$$
- *Avalanche effect*: a small change, even a single bit, in input must generate a drastic change in output such that it is not possible to find a correlation between the two outputs.

The combination of the preceding properties makes the use of hash functions important in many applications. An example is the verification of data integrity. Hash functions allow the computation of a value of finite length that uniquely identifies the input data. The digest provided from the latter can be viewed as a kind of digital fingerprint. Then, a user easily checks if an information has been manipulated by an attacker during a transmission as the modification of even a single bit totally changes the digest due to the avalanche effect.
The implementation of a hash function is also required in most digital signature protocols and in proof of work.
The last application of hash functions that we present concerns the design of a *bit commitment protocol*. The purpose of the protocol is to use a hash function to hide a prediction and reveal it only after a stipulated time by proving that the prediction sent has not been altered. The same mechanism is used by dispute resolution platforms to maintain the secrecy of the jurors' votes until the ruling is revealed. We identify the *commit phase* which consists in submitting the hidden prediction to the network and the *reveal phase* during which the user who made the prediction allows other users to verify it. The protocol must have two key features: *hiding* and *binding*. The first property requires that, at the end of the commit phase, nobody should be able to obtain any information on the prediction submitted. The binding property instead tells us that, during the reveal phase, there is only one possible value that returns as an ouput the commit sent during the first phase.
Let us introduce an example to describe the protocol step by step. We suppose that Alice wants to make a prediction and she does not intend to reveal it until it can be verified. Meanwhile, Bob must be sure that Alice cannot change her opinion by altering the outcome. Let b be the prediction made by Alice and h the common hash function chosen. The protocol is divided into the following steps:

1. Alice generates two random bit strings R_1 and R_2;
2. Alice calculates the value $y = h(R_1||R_2||b)$ and submits R_1 and y to Bob. We recall that b is Alice's prediction;

3. At the beginning of the reveal phase, Alice sends Bob her prediction b and the string R_2;

4. Bob calculates the value $h(R_1||R_2||b)$ and checks that it is equal to y.

Lastly, we cite the family of cryptographic functions *Secure Hash Algorithms* also known as SHA developed by the *National Security Agency* (NSA) and published by the *National Institute of Standards and Technology* (NIST) as a federal standard of the US government. In particular, we are interested in two hash functions *SHA-256* and *Keccak-256*, employed respectively in Bitcoin and Ethereum. Both functions return a digest with a length of 256 bits.

1.4.2 Public key cryptosystem

The second tool to analyse is public key cryptography or asymmetric encryption.
Cryptography is the study of techniques and protocols to keep communication between multiple users secure and private, preventing contents to be accessible by unauthorised third parties.
We are interested in asymmetric system or *public key cryptography*. Each user generates a pair (sk, pk), respectively called secret key and public key. As the name suggests, the secret key is known only by the owner and it is recommended to generate it randomly, while the public key is clearly visible to the network. Furthermore, the public key is derived mathematically from the secret one and the generation algorithm used must have the property that sk is computationally infeasible to find from pk.
The public key is used to generate the address of the recipient in a transaction. On the other hand, the private key allows the sender to prove that they are the owners of the amount of cryptocurrency to be transferred.
A huge improvement in public key systems was the introduction of elliptic curve cryptosystems (ECC), the third tool which is analysed.

1.4.3 Elliptic Curve Cryptosystem (ECC)

In 1985, Neal Kobitz and Victor S. Miller independently suggested the use of elliptic curve cryptography (or ECC). ECC is more complex and by consequence was adopted some years later, but on the other hand it's more efficient and stronger than classical public key systems. For example, ECC allows to create shorter keys, while maintaining high the security level.

Definition 2 *An elliptic curve in short Weierstrass form E, defined over a field $\mathbb{K}$, is the set of points (x, y) in the Cartesian plane that satisfies the equation*

$$y^2 = x^3 + ax + b \tag{1.1}$$

where a, b must fulfill the condition

$$4a^3 + 27b^2 \neq 0. \tag{1.2}$$

The condition imposed on a and b ensures that the curve is regular.
Given two points $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$, we can introduce a third sum point $S = P + Q$ such that $S \in E$. We define the sum operation between two points $P + Q$ on an elliptic curve as follows:

1. We draw the line passing through P and Q.
2. The line will intersect the elliptic curve E at a third point R with coordinates (x_R, y_R).
3. We reflect the point R with respect to the horizontal axis and the point found is

   $$P + Q = (x_R, -y_R)$$

   .

Given the points $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$, we obtain the coordinates of the point $S = P + Q$ as

$$x_R = \lambda^2 - x_P - x_Q \tag{1.3}$$
$$y_R = \lambda(x_P - x_R) - y_P, \tag{1.4}$$

where $\lambda = \frac{y_Q - y_P}{x_Q - x_P}$ is the angular coefficient of the line passing through P and Q.
We must note that the operation defined above is not complete because two cases are not considered.

- The line through P and Q does not intersect the curve E. This case occurs if the line passing through P and Q is parallel to the y-axis. Then we introduce the point at infinity ∞ which does not belong to the plane and such that $P + Q = \infty$.
- P and Q coincide, so there is a bundle of lines passing through them. In this case we choose the tangent line to P which is unique since by definition the elliptic curve is regular. If the intersection $R = (x_R, y_R)$ with the curve E exists, then the coordinates of $S = P + Q = 2P$ are calculated with the doubling formula

  $$x_S = \lambda^2 - 2x_P$$
  $$y_S = \lambda(x_P - x_S) - y_P$$

  with $\lambda = \frac{3x_P^2 + a}{2y_P}$.
  If there is no intersection with the curve E, then $2P = \infty$.

It is possible to prove that E is a commutative group with this sum operation.
For cryptographic applications, we are interested in defining the elliptic curve over a finite field.

Definition 3 *Given a finite field $\mathbb{F}_p$, with $p > 3$. We define an elliptic curve E defined over a finite field $\mathbb{F}_p$ as the set*

$$E(\mathbb{F}_p) := \{(x, y) \in E \mid x, y \in \mathbb{F}_p\} \cup \{\infty\} \tag{1.5}$$

Even in this case, one can prove that the elements of $E(\mathbb{F}_p)$ satisfy a group law with respect to the sum defined above. For cryptographic applications, it is useful to know

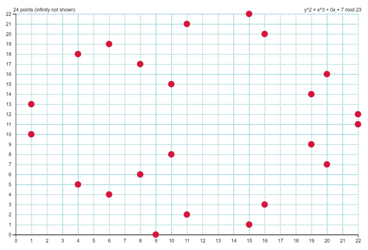


Figure 1.2. An example of ECC over a finite field $\mathbb{F}_p$. In this case $a = 0$, $b = 7$ and $p = 23$.

how to efficiently compute the value kP. The first way is to calculate $P + P + P + ... + P$ k times. A more efficient method is the *double-and-add*, which achieves the same result but with less steps. We explain how the algorithm works through an example. If $k = 13$, we may represent this value in powers of 2, so $k = 2^3 + 2^2 + 2^0 = 8 + 4 + 1$. Consequently, $13P$ can also be rewritten as

$$13P = 2^3P + 2^2P + P = 8P + 4P + P$$

The double-and-add method requires the following steps:

- Take the point P;
- Double P to obtain $2P$;
- $2P$ is not present in the reformulation of $13P$ so we do not add this term;
- Double $2P$ to obtain $2^2P = 4P$;
- $4P$ is present in the reformulation of $13P$ so we add this term to P and we get the partial result $4P + P = 5P$;
- Double 2^2P to obtain $2^3P = 8P$;
- $8P$ is present in the reformulation of $13P$ so we add this term to the partial result $5P$ and we get the final result $8P + 5P = 13P$.

To achieve our result, we need 3 doublings and 2 additions for a total of 5 operations.
Let us explain how to build a public key system on an elliptic curve. We introduce an elliptic curve E defined over a finite field $\mathbb{F}_p$ with p a large prime number in order to generate as many points on the curve. Next we fix a point G generator of $E(\mathbb{F}_p)$ such that the order of G is a large prime number equal to N. Each user chooses a number k and computes the point $P = kG \mod N$. The pair (k, P) represents a user's private and public key respectively. We observe that the choice of G of high order allows the construction of a larger number of public keys.
The security of ECC is guaranteed by the Elliptic Curve Discrete Logarithm Problem, or ECDLP. ECDLP consists in finding the value d, given a point P, that satisfies the equation $dG = P \mod N$. Nowadays, there is no efficient algorithm for solving this problem and as a result, the use of ECCs is considered safer than other public key systems.
One of the most famous curves is the *secp256k1* curve, used by both Bitcoin and Ethereum. The equation of this curve is obtained by placing $a = 0$ and $b = 7$, so

$$y^2 = x^3 + 7 \mod p.$$

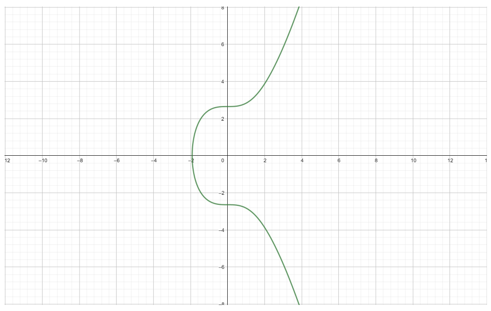


Figure 1.3. The representation of $y^2 = x^3 + 7$ on $\mathbb{R}$.

The prime number p used for this curve is $2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$, so private keys are represented with a length of 256 bits or 32 bytes. Consequently, the length of a public key P is 64 bytes (32 for each coordinate). The curve equation suggests a more efficient way to store the public key in a compressed form. In fact, the size can be reduced to 33 bytes such that 32 correspond to the x-coordinate of P and 1 byte is used to detect the sign of the y-coordinate. Tracking the sign is important because the equation is quadratic in y and therefore there are two acceptable values.

1.4.4 Digital signatures

The purpose of a digital signature is to prove the identity of the sender to a recipient. The idea is to combine hash functions and public key cryptography to reach this goal. A user creates a signature with his private key and the network checks it through the public

one. One of the most important protocol used is the Digital Signature Algorithm, or DSA, which also has a version on elliptic curves, called ECDSA. Most of the blockchains, including Bitcoin and Ethereum, adopt ECDSA.
Given the equation of an elliptic curve, the generator G, the total number of points equal to the prime N and a shared hash function h, if the user with the couple of keys $(k, P = kG)$, wants to sign a message M using the protocol ECDSA he has to follow these steps:

1. the sender calculates the digest $h = h(M)$;
2. the sender chooses randomly an integer $d \in \mathbb{Z}_N$;
3. the sender calculates the point $dG = (x, y)$;
4. the sender fixes $r = x \mod N$ and then computes $s = (h + rk)d^{-1} \mod N$;
5. the signature is composed by the couple (r, s) for a total size of 33+32 bytes.

The recipient can check the signature in this way:

1. he finds the inverse of s, which is $w = s^{-1} = d/(h + rk)$;
2. he calculates $u = wh$ and $v = wr$;
3. he computes the point $Q = uG + vP$, where $P = kG$ is the public key of the sender;
4. he accepts the signature if the coordinate $x_Q \mod N$ of Q is equal to the first element of the signature r.

This protocol works because:

$$Q = uG + vP = whG + wr(kG) = \left(\frac{dh}{h + rk} + \frac{drk}{h + rk}\right)G = dG = (x, y).$$

Let us make two observations. The first one is that the signature can be build only by the signer, because he is the only one who knows the value of his private key k.
The second observation is the dependence of the signature on the message via the hash $h = h(M)$. Consequently, the same signature cannot be used for two different messages.
In the blockchain scenario, the digital signature is used to prove the ownership of cryptocurrencies or digital assets, or to send them to another user.

1.5 Definition of blockchain and general overview

Definition 4 *A blockchain is a particular type of Distributed Ledger Technology (DLT) where data are organised in a growing list of ordered blocks.*

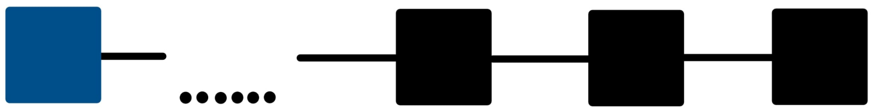


Figure 1.4. The chain structure of a generic blockchain. The blue block represents the genesis one.

So, a block can only be added and once inserted it is not possible to modify or delete it. This mechanism is managed directly by nodes of a peer-to-peer network, which cooperate to support the blockchain, following the dictates of a shared consensus protocol. The last-mentioned guarantees the presence of a unique and valid state of the data structure. Each block contains a reference to the previous one constituting the chain structure. The only block that does not have this link is the first of the chain, called *genesis block*.
The combination of cryptography and consensus protocol guarantees security and immutability. In practice, the main purpose of this structure is to store data, which can be of different nature, by simple money transactions to entire programs or applications. Each user has a pair of private and public keys. The private key allows the sender to sign a transaction. The signature proves to the other nodes that the sender is the owner and he is entitled to proceed with the transaction.
It is essential that every user keeps their private key hidden because anyone who holds of it would be able to spend the associated cryptocurrency. In most cases, key management is entrusted to applications called *wallets*. These can be software or even external hardware. A wallet does not properly contain cryptocurrencies, which are registered in the blockchain, but contains the private keys that allow you to unlock and send them.
There are different types of blockchains, depending on the level of decentralisation, but the most common are: public and private. In *public blockchains*, anyone can send transactions or become a validator and there are not access restrictions. For example, Bitcoin and Ethereum are both public. An economic incentive mechanism is designed to stimulate nodes to develop public blockchains.
The consensus protocols commonly used are *proof-of-work* or *proof-of-stake*.
As the name suggests, instead, a *private blockchain* is not accessible to anyone, but joining the network requires authorisation and an invite from the organisation that manages it. In this case, the consensus protocol relies on a few nodes that are considered trusted. Although in contrast to the ideal of decentralisation underlying this technology, several private blockchains have been developed in recent years as they allow companies to collaborate transparently even in the absence of mutual trust.

1.6 Blocks structure

A block is the data structure added sequentially one at a time, that represents the fundamental component of a blockchain. The structure and size depend on the specific protocol adopted, but there are some generic properties. Each block is identified by a hash value, called *hash block*, generated directly with the block data inside. We remark that this value is not recorded directly in the corresponding block because it is not known until the creation of the respective block.
A block is split into a header and a body. The former includes basic information about the block, such as the hash of the previous one and other management fields that we will explain later. Instead, the body collects transaction data.
Let us observe that the hash of the previous block is one of the inputs used for the hash block computing, creating in practice the link between them. This use of hash functions guarantees the immutability of data recorded, because a change in any block produces an avalanche effect that spreads over all subsequent hashes and every node would notice the attempt to commit fraud.

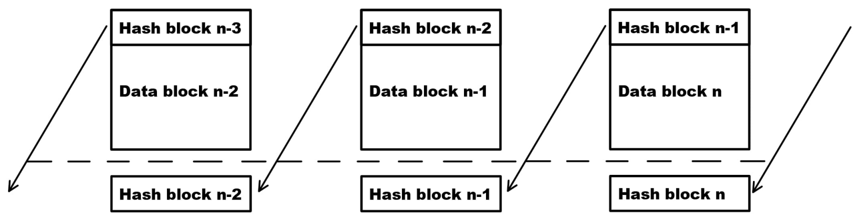


Figure 1.5. The link between blocks through hash functions.

The *timestamp* is stored among the management fields. The timestamp is a parameter that indicates the data and the time in which a block is generated. It is important to save this value because blockchain is a first-to-file system, so it is necessary to establish a chronological order.
The *block number* or *block height* represents another way to identify blocks and it is the relative position to the genesis block one, which is number zero for definition.
Another important field is the *Merkle tree root*, which resumes data transactions in a single string. There are different models of Merkle trees, for example Ethereum's blockchain uses a different version called Patricia Merkle tree. In general, a Merkle tree is a type of binary tree. In this case each leaf node is the hash value of a transaction in the block. The intermediate nodes are built by calculating the hash of the two child nodes, and this process is repeated until the root node is reached.

The benefit of using a Merkle tree is twofold. This data structure allows an efficient

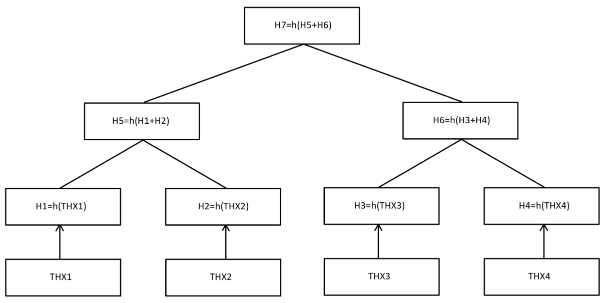


Figure 1.6. Structure of a Merkle tree.

verification of a transaction. A second benefit in compacting all transactions into a single string via a Merkle tree root is in disk saving.
For example, the hash of a block in Bitcoin is computed by feeding the header data as input to the hash function. In this way, if a node wants to check the correctness of the entire blockchain, it may save only the headers and avoid downloading bodies which represent most of the block in terms of storage. In fact, a header has a weight of 80 bytes, while the total size of a block is about 1 megabyte.

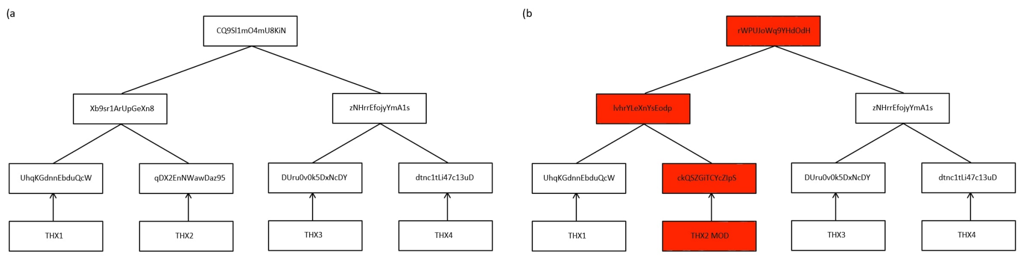


Figure 1.7. a): The correct state of the blockchain. b): The attempt of change the second transaction brings a different value of the Merkle tree root.

Other fields refer to proof of work related parameters: *nonce, difficulty* and *target*. The nonce is a casual string added with the purpose to solve the proof of work. The difficulty is a measure of how difficult is to mine the next block in relation to the genesis one. This value is adjusted by protocol to keep constant the *block time*, which refers to the average time it takes to mine a new block. When blocks are added faster, that is when there is more computational power over the network, the system tends to increase the difficulty. On the other hand, a higher time block leads to a decrease in difficulty, that prevents the accumulation of transactions. Every blockchain has its target time, for example in Bitcoin a miner adds a new block on average every 10 minutes, while in Ethereum it happens on

average every 12 seconds.
The last important concept concerns the size of the blocks, which is bounded. The protocol prevents the insertion of arbitrable size blocks, which would weigh and slow down too much the network and consequently there would be a loss of efficiency of the entire blockchain. Also, in this case each blockchain has its personal standards.
When we introduced Ethereum, we defined the notion of gas as the cost required to process a given operation. The protocol recommends that each block performs transactions consuming a total amount of about 15 million gas, but if there are particularly high transaction volumes this value doubles. It is difficult to estimate the size in terms of storage, but through the site Etherscan.io we can observe that rarely blocks exceed the size of 200 KB.

1.7 Scalability problem

We are interested in discussing scalability, which represents one of the major issues facing this technology.
Scalability is the capacity of a blockchain to manage an increasing number of transactions without the loss of performance. The block size restriction and stable insertion rate render the management of rapid growths in transaction volumes difficult.
The consequences are hours of waiting for a transaction confirmation and a higher cost for fees. This problem is also known as *blockchain trilemma* and scalability is the cost to be paid to ensure a high level of decentralisation and security, which are the foundation of this technology. An ideal blockchain holds all these three properties, but in practice is very hard to find a balance among them. The search for a possible solution to the trilemma is still today object of debate, and in some cases has also led to irremediable splits between communities. The most classic proposals are on the change of the physical parameters of blocks, such as increasing the size or reducing the block time. Another adopted solution consists in moving transactions off-chain in a second layer level and record them subsequently on the blockchain.

1.8 Nodes and miners

A node is any device that is able to interact with the blockchain. All the nodes together create the peer-to-peer network. Then, the data is transmitted from node to node until it spreads over the whole network. Two types of nodes can be distinguished: *full nodes* and *light nodes*.
Full nodes store the entire blockchain, starting from the genesis block. They have to update their version of the chain, whenever a block is inserted, and they verify all transactions proposed by the network. The set of full nodes ensures both decentralisation and immutability of the blockchain.

On the other hand, light nodes store only the block headers. They can send transactions and verify them through the Simple Payment Verification mechanism only if helped by a full node. An example of a light node is a wallet that runs inside a smartphone.
Furthermore, there are special actors that manage the addition of blocks in accordance with the consensus protocol. They are commonly referred as *miners* in proof-of-work-based blockchain. Miners compete with themselves in order to insert a block and receive a reward, that consists in new coins (that is the reason of the term miner) and transaction fees.

1.9 Consensus protocols

All blockchains must face the problem of reaching consensus. Since there is no central authority to manage data entry, it is necessary to build a mechanism able to create a general agreement between the nodes of the network about the current state of the chain and how it would evolve.
The introduction of a consensus mechanism prevents incorrect data entry, maintains the integrity of the transaction history and manages the block addition. Each node verifies the validity of the proposed transactions, which will be recorded on the blockchain if they are approved.
Without this kind of mechanism, a malicious user could build blocks containing fraudulent transactions, such as double-spending. As a result, trust in the system would decrease and users may tend to leave the blockchain.
Another feature that is required for consensus protocols is the resistance to Sybil attacks. A Sybil attack consists in the creation of several pseudo-identities to achieve majority.
There are different types of consensus protocol, but the most used ones are *proof of work*, based on computational power used to resolve mathematical problems, and *proof of stake*, based on the amount of cryptocurrency that a user is disposed to place at stake.
In both cases the following chain selection rule is applied : nodes consider as the only valid state, as a single source of truth, the longest chain.

1.9.1 Proof of work (PoW)

A proof-of-work protocol consists in proving to the network that a large amount of computational power has been invested in solving a complex mathematical puzzle. As aforementioned, miners compete with each other and the first one to reach a solution has the chance to extend the chain and receive the block reward. The cost of participating in this mechanism is expressed in terms of time-computing, hardware equipment and electricity.
The mathematical puzzle proposed to miners involves the search for a value called *nonce*. The latter must be entered as input to a hash function together with the data of the candidate block. The digest produced must begin with a specific number of zero bits. This condition is defined by a target value given in the last block of the chain. We observe that

the average work to be performed is exponential in the number of zero bits required.
The use of a hash function makes the process pseudo-random, so the best way to solve the proof of work is through trial and error. Consequently, a miner with higher computing power can perform a greater number of attempts, which represents an advantage in the search for the nonce.
The most important advantage of a proof-of-work consensus protocol is the robustness to different types of attacks. As we have mentioned, the PoW is resistant to Sybil attacks. Indeed, distributing computing power over multiple addresses is not convenient for the miner, which on the contrary risks lower performance.
If the network of miners is particularly numerous, the protocol is highly resistant against a possible 51% attack. This is because the total computing power is better distributed.
The disadvantages of proof of work are the excessive consumption of electricity, which causes serious environmental damage, and the foundation of mining pools where a massive amount of computing power is accumulated, representing a risk for the protocol's decentralisation.

1.9.2 Proof of stake (PoS)

Another way to reach distributed consensus is though proof of stake protocols, which represent an alternative to the proof of work described above. In proof of stake, the mechanism for adding and checking blocks is assigned to nodes referred to as *validators*. Users which intend to participate as validators temporarily deposit an amount of their capital, called *stake*, within a smart contract. In contrast to PoW, where miners compete among themselves to enter a block, in PoS validators are alternated through a random drawn based on the quantity of frozen cryptocurrency. In practice, the nodes with the largest stake are more likely to be drawn. Here is a summary of how a proof of stake protocol works:

1. the protocol chooses randomly a validator;
2. if the validator fails to participate, he loses an amount of his stake equal to the average reward generated by a block;
3. if the validator participates, he builds a block and proposes it to the network;
4. the protocol randomly selects a committee of validators. Again, they are penalized if they fail to participate. The committee receives the candidate block from the peers;
5. each member of the committee transmits to the network an *attestation* in favour of the block if it is correct;
6. if the block receives a minimum number of attestations, it is added to the chain and all validators are rewarded;

7. if the block contains errors, the validator who proposed it suffers a heavy penalty.

The existence of a compensation and penalty system ensures the security of such protocols. A validator receives a reward for both proposing the block and making an attestation. Sanctions, on the other hand, discourage inappropriate behaviour by validators.
There are two different types of penalties. The first one affects lazy behaviours. To ensure an adequate level of scalability and quality of the blockchain, the majority of nodes must always be active. If one of them is elected, but it fails to participate, the system burns from its stake a quantity of cryptocurrency equal to the average value of a block reward. The second type of sanctions punishes validators who have a dishonest behaviour. In this case, the penalties are much more severe and can burn much part or all the stake and lead to the ejection of the validator from the network. The main dishonest behaviours are the proposition of several blocks at once and the attempt to insert wrong or fraudulent transactions. This staking mechanism is also used in mostly dispute resolution applications in order to select jurors.
We detect several advantages in using PoS over PoW. The first one is the greater sustainability. The mining process of Pow requires a huge quantity of electricity in order to keep on specific hardware that solves complex mathematical problems. On the other hand, in PoS energy consumption is equal to the electricity used daily to power a pc.
A second benefit is the greater resistance to 51% attack. In this case, it is much expensive to get more than half of the cryptocurrency present in the network, because from its nature the value is defined only by demand. If an attacker wanted to buy a huge amount of that cryptocurrency, the value would rise exponentially in a very rapid time, making impossible to complete the attack. If the attack succeeds anyway, the set of honest nodes may decide to create a new chain through a hard fork. In this new version of the blockchain, they burn the stake of the attacker, causing him a huge damage.
Another advantage is that proof-of-stake-based blockchain results more decentralised. Theoretically, to participate in the network it is enough to have a positive cryptocurrency balance. On the other hand, to solve the PoW efficiently nodes need to equip themselves with specialized hardware, i.e. ASICs, which are very expensive.
It is appropriate to specify that also in the PoS there may be limits to accessibility. Some protocols require a minimum amount of cryptocurrency for staking and this value can be high. For example, Ethereum validators must stake a minimum of 32 ETH which correspond to about 50'000 dollars (on date 28/05/2022).

1.10 Fork

The term *fork* in a blockchain refers to the generic phenomenon that generates two or more branches in the chain. There is not a single definition because causes and consequences could be very different. It can be a temporal situation, as in the case of a regular fork, or it can even lead to a chain-split with the creation of a new cryptocurrency. According to the context, three different categories of fork are identified: regular, soft and hard.

1.10.1 Regular fork

A *regular* or *accidental* fork occurs when two or more valid blocks are proposed simultaneously to the network, so they have theoretically the same height. The cause of this phenomenon lies in the asynchronous data transmission due to the peer-to-peer scheme because information only spreads between the nearest nodes. As a result, nodes do not receive the same candidate block and temporarily different branches of the blockchain arise. All versions are correct because there is not a central authority that resolves the ambiguity. By consequence, a miner starts working on the branch he receives first. The fork resolves itself when the next block is added because it identifies the longest chain, which represents the only valid state. Miners then move on the latter. Blocks of the other branches, that are defined *orphans*, are abandoned and transactions in them are annulled. The possibility that a transaction may be contained in an orphan block explains why it is necessary to wait for multiple confirmations of a block to consider a transaction definitive.

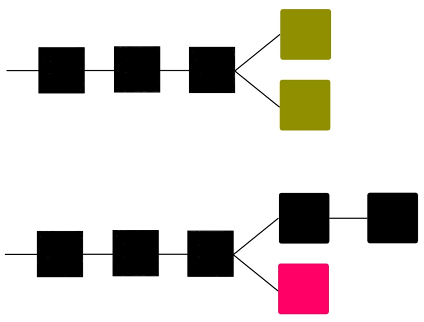


Figure 1.8. Upper: an example of regular fork. Under: the fork is solved in favour of the higher branch. In pink is represented the orphan block.

1.10.2 Soft fork

The expression *soft fork* indicates a retro compatible update of the software that manages the blockchain and it is typically an optimization of the current protocol. A node may decide not to download the update and still interact with the blockchain.

1.10.3 Hard fork

On the contrary, a *hard fork* is a variation of the protocol that is not backward compatible with the previous one. Since there is no central authority, it is not possible to impose protocol changes to the blockchain community, so a proposal must be submitted to the network. This concept also applies to soft forks. The main difference is that nodes are forced to update all their systems following a hard fork in order to interact with the blockchain. In this case, two different scenarios may be developed. In the first one, all nodes adhere to the new protocol and the chain continues its undisturbed growth. In the second one, the community is split in two on the proposal and a part of the nodes decides to break away from the main blockchain, creating a new one. This phenomenon is referred as chain-split. The two chains share the same block history until the moment the hard fork took place and from that time they develop independently. By consequence, when a chain-split occurs a node holds the same balance of cryptocurrency in both blockchains. There are different reasons that lead to a hard fork. The purpose is generally to make valid blocks that are not in the current version. Another reason could be the cancellation of some transactions following an attack as in the case of Ethereum. An organisation called *theDAO* was hacked in 2016 and an amount of 3.6 billion ETH was stolen from it. One side of the community wanted to cancel the attack, while the other side considered that action permissible since attackers had exploited a bug in the contract code. As a result, the blockchain was split in Ethereum, where the state was returned prior to the attack, and Ethereum Classic.

1.11 Smart contracts

A *smart contract* is a computer code that runs automatically on the top of a blockchain when certain conditions are met. This concept was introduced in the early 1990s by Nick Szabo. Smart contracts allow to make agreements between two untrustworthy parties without the presence of a trusted third part. In fact, a smart contract eliminates the counterparty risk that occurs when a party tries to not comply with the conditions. The code is stored on multiple nodes of the blockchain ensuring immutability and transparency. As we said, Ethereum represents the first and still today the most important platform for developing smart contracts and the most used programming language is Solidity. The activation of one of these takes place through the receipt of a transaction. The cost of a smart contract's activation depends on its complexity. On Ethereum, the higher is the number of computational steps, the higher will be the amount of gas that will be used. Let us remember that the introduction of this cost is a defence mechanism that prevents the execution of smart contracts that never halt. In fact, since Ethereum has a Turing-complete scripting language, an attacker could program a smart contract for it which executes an infinite loop clogging the network. The cost of gas to carry out the attack would have to be unlimited, so it is impossible to sustain.
By a technical point of view, the structure of a smart contract is defined by states and functions. The first ones are data stored within, that could be defined as constants or variables. Functions collect the set of commands to run. They are classified in *read-only functions* and *write functions*. The former group does not require any cost, on the other

hand the second one is responsible to store changes in states on blocks and requires the payment of a transaction fee. Among the most common uses, we remind:

- automatic payment due to the achievement of specific results or due to triggering events;
- impose penalties, such as in PoS;
- creation of tokens. In Ethereum, the first standard for tokens' creation is ERC-20. The introduction of the latter has led to a huge development of this kind of applications;
- crowdfunding through a process referred as *Initial Coin Offering* or *ICO*. This mechanism allows to raise directly cryptocurrency from nodes in order to support a project.
- foundation and management of *Decentralised Autonomous* organisations or *DAOs*.

Other applications of interest in this paper are referred to dispute resolution mechanisms. A smart contract can be used as a tool to automatically enforce a ruling or to redistribute rewards among jurors.

1.11.1 Oracles

Oracles are third-party services that provide a bridge between a blockchain and the off-chain world. They are responsible for taking information and data outside the blockchain and put them into smart contracts.
Thereby, an oracle is not directly the source of that data but has the duty to recover, verify and authenticate them. To some extent, oracles represents an elegant solution to accessing off-chain resources. On the other hand, they may cause a incorrect performance of the smart contract. The latter, once activated, is impossible to arres, hence the code runs all its predefined steps. Furthermore, some oracles have a centralized structure, so they must be reliable and this fact reduces the level of decentralisation by depending on a trusted third party.

1.12 Token

A token is a digital representation of a value or a right and it is in form of a coin. There are various classifications of tokens, but the one that is considered classic is according to their use.
A high level classification divides them into three different classes: *utility tokens, security tokens* and *payment tokens*. Cryptocurrencies, such as BTC, are also tokens. They belong to the category of payment tokens and differ from others in that they are native to a blockchain.

Utilities and security tokens are generated by smart contracts so they can be designed with even more complex features, making them 'programmable'. An utility token guarantees the right to access a service offered by the platform that generated it. The set of applications is very wide. As seen, a token can be used to fund projects through an ICO. Another application is to confer a voting right to its holder. Usually the vote does not concern real governance issues. A very popular application is the creation and trading of *Non Fungible Tokens* or NFTs, that has led to the remarkable development of cryptoart. Ethereum enables the creation of them through the standard ERC-721. Each NFT is unique and its value is defined exclusively by the request. In this case, the token proves the ownership of an asset. Some utility tokens are used as rewards when certain tasks are completed. Sometimes they are multi-purpose. For example, tokens in Kleros, which is a platform that deals with the resolution of disputes that we will investigate later, allow the owner to participate in the voting mechanism and at the same time is used as a reward for a honest behaviour.
Security tokens guarantee the participation in the profit of the platform that issued them through dividends. Most applications involve the process of digitizing a company's assets, called *tokenization*.

1.13 Decentralised Autonomous Organisation (DAO)

A *Decentralised Autonomous Organisation* or DAO is an organisation where activities and executive power are managed entirely by smart contracts. It is decentralised in terms of both the infrastructure and the governance.
A DAO relies on a blockchain that records transactions and the digital properties of its assets. Even in this case, the choice of this technology falls on the extreme difficulty of manipulating data thanks to the consensus protocol and decentralisation.
Governance is not based on a traditional hierarchical system. Decisions within the DAO are made collectively through a voting system. Each member has the opportunity to propose actions aimed at the management or development of the organisation. The right to vote on proposals is conferred to holders of DAO tokens. This system is such that a greater number of tokens corresponds to a greater influence in the decision-making process. In the following chapters, we will give the example of a dispute resolution platform that offers support to users of these organisations in the management of governance issues.
A DAO is defined as autonomous since the executive power is entirely entrusted to smart contracts. The combination of these organisations and a system of oracles can lead to the definition of a structure completely independent of the human factor.
Finally, when a DAO generates profit, it divides the latter among its members in the form of cryptocurrency.

Chapter 2

Game Theory

Game theory is a branch of mathematics that studies models of interaction between rational agents, referred also as *players.* In each game is defined a well-established set of rules that provides the set of possible actions. In addition, each player is assigned a numeric value defined as *utility,* which measures the preference in performing a certain action to achieve a desired outcome. Thereby, an agent is defined as rational in the sense that its choices aim to maximize the personal expected utility. A series of possible moves is called pure strategy. Another typical hypothesis of game theory is that each player has a common knowledge of the game, so rules, strategies and payoffs are specified.
Our focus is on non-cooperative games. In this class, it is assumed that players choose strategies simultaneously , or they can consider themselves independent from each other, without knowing what the other players have chosen. This assumption is without loss of generality and describes the typical scenario of blockchain dispute resolution protocols, where judges do not know each other and the mechanism governing votes is a game based on the concept of *Schelling* or *focal point.*
In this chapter, first we will define a game in its normal or strategic form. Then, we will introduce the concept of Nash equilibrium, which is the basis of the Schelling point and we will see applications related to two-players games. Finally, we will define the games that mainly characterise the protocols in Chapter 3.

2.1 Games in normal (or strategic) form

Our focus on game theory begins with the definition of games in strategic form. Let $\mathcal{V}$ be a finite set of players. In general, each player is assigned a finite set of actions $\mathcal{A}_i$, where the index represents the player $i \in \mathcal{V}$. In our applications, we can relax this hypothesis and define a single set of actions $\mathcal{A}$. Next, we define the configuration space:

$$\mathcal{X} = \mathcal{A}^{|V|},$$

whose elements are vectors $x \in \mathcal{X}$ referred as *action profiles* or *configurations*. The entries of x represents the actions played by each player in that time.
As mentioned above, players choose the moves to be made rationally, so as to influence the evolution of the game in their favor and maximize utility (or minimize it). In the first case, we talk about a *profit game*, while in the second one about *cost game*. To formalize the concept of utility, we equip each player with a *utility* or *payoff function*

$$u_i : \mathcal{X} \longrightarrow \mathbb{R}.$$

This function takes as argument a configuration and returns the value of the expected utility payoff of player i. Note that the utility of a player also depends on the actions $x_j \in \mathcal{A}$ chosen by others players j.

Definition 5 *We define a game in strategic form as the triple* $\Gamma = (\mathcal{V}, \mathcal{A}, \{u_i\}_{i \in \mathcal{V}})$.

2.2 Nash equilibrium

The definition of *Nash equilibrium* is fundamental in game theory.

Briefly, a Nash equilibrium is a configuration where no player has an incentive to change the current move individually, as it would decrease his own utility. This notion is named after the American mathematician John Nash, who in 1951 published the article *Non-cooperative Games*, in which he proves the existence of at least one equilibrium in terms of mixed-strategy in any game with finite sets of players and actions.
Let us define a *mixed-strategy* as a probability distribution over a set of pure strategies. The probability of each pure strategy is assigned in terms of *expected payoff*. Indeed, a player must evaluate which mixed strategy to adopt based on the payoffs of a given scenario. In particular, we are interested in defining the Nash equilibrium in a pure strategy game.
First, we introduce a notation that is often used and indicates the configuration vector without the i-th entry

$$x_{-i} = x_{\mathcal{V} \setminus \{i\}}.$$

This formulation allows us to rewrite the utility function of a player i at configuration x as

$$u_i(x) = u_i(x_i, x_{-i}).$$

At each step of the game, the i-th player must try to maximize $u_i(x_i, x_{-i})$. Let us now introduce the *best response function*

$$\mathcal{BR}_i(x_{-i}) = \arg\max_{x_i \in \mathcal{A}} u_i(x_i, x_{-i}).$$

The above function returns the move of the player i $x_i^* \in \mathcal{A}$ that guarantees his highest payoff for that particular action profile. The assumption is that the player can know in

advance the moves of others and then can choose his best strategy. As already mentioned, a rational agent aims to maximize his utility.
Otherwise, we may introduce the dual problem, which consists of searching for the action $x_i \in A$ that minimises the player's cost function u_i^*, defined ad $u_i^* = -u_i$.
The definition of the best response of a player $i \in \mathcal{V}$ for the dual problem becomes

$$\mathcal{BR}_i(x_{-i}) = \arg\min_{x_i \in \mathcal{A}} u_i^*(x_i, x_{-i}).$$

Definition 6 *A pure strategy Nash equilibrium is an action profile $x^e \in \mathcal{X}$ such that*

$$x_i^e \in \mathcal{BR}_i(x_{-i}^e) \quad \forall\, i \in \mathcal{V}. \tag{2.1}$$

There is also a definition in terms of utility function. A Nash equilibrium must meet the following relationship

$$u_i(x_i^e, x_{-i}^e) \geq u_i(x_i, x_{-i}^e) \quad \forall i \in \mathcal{V},\ \forall x_i \in \mathcal{A}. \tag{2.2}$$

There are four more general observations.
The first is that there are games in which a Nash equilibrium in pure strategy does not exist. An example is the rock-scissors-paper where players can always improve their utility changing move.
The second observation is that a game can have more Nash equilibria. A Nash equilibrium is referred as *strict* if in the configuration x^e each player has a unique best response, hence no player can unilaterally change strategy without reducing his own utility. Besides, a Nash equilibrium is defined as *weak* if there is at least one player who has the possibility of changing his strategy while maintaining exactly the same payout.
The third observation is that a Nash equilibrium can be inefficient in the Pareto sense. In general, an outcome is defined as *Pareto efficient* if there are no outcomes that can lead to an increase in payoff for each player. So there could be a equilibrium that displeases everyone, because there is a configuration that is not a equilibrium, but that would make them earn more.
The last observation is that in a game there can be different Nash equilibria, that are nonequivalent and noninterchangeable. Two axioms, proposed by Harsanyi and Selten, intervene to choose between the different balances. The first is called *payoff-Dominance.* Given two different equilibria x^{e1} and x^{e2}, x^{e1} payoff-dominates x^{e2} if and only if the inequality

$$u_i(x^{e1}) > u_i(x^{e2}) \ \ \forall i \in \mathcal{V}$$

holds, i.e. x^{e1} is a Pareto efficient configuration among the Nash equilibria. The second axiom is the *risk-dominance principle.* The idea is that given two Nash equilibria x^{e1} and x^{e2}, the former risk-dominates the latter if and only if

$$\min_{i \in \mathcal{V}} u_i(x^{e1}) > \min_{i \in \mathcal{V}} u_i(x^{e2}).$$

Therefore, players who follow the risk-dominance approach choose the equilibrium point that minimises their loss and, at the same time, ensure them a minimum reward.

2.3 Two-players game

Let us consider the simplest case: games with just two players $\mathcal{V} = \{1,2\}$ to whom we respectively associate the utility functions $u_i(x, y)$, for $i = 1,2$. The first entry x indicates the action played by the player i, while the second y is the action played by the opponent. If the action set is finite and if we assume that $|\mathcal{A}| = n$, the utilities of two-player games can be represented within a table, defined as *payoff matrix* in $\mathbb{R}^{n\times n}$. The rows correspond to the moves chosen by player 1, while the columns correspond to the moves chosen by player 2. The (x, y)-th entry of the matrix is the pair of utility values $(u_1(x, y), u_2(x, y))$ where player 1 and player 2 choose the actions x and y respectively. Games with two players in which the set of actions is binary, i.e., $|\mathcal{A}| = 2$, are of particular importance in game theory. This class of games is referred to as 2×2-*games*. An example of a payoff matrix for a 2×2-game is shown in the table 2.1.

	0	1
0	$(u_1(0,0),\ u_2(0,0))$	$(u_1(0,1),\ u_2(0,1))$
1	$(u_1(1,0),\ u_2(1,0))$	$(u_1(1,1),\ u_2(1,1))$

Table 2.1. Payoff matrix of a two-player game with an action set $\mathcal{A} = \{0,1\}$.

Let's add the symmetry hypothesis to games with two players. The utility functions $u_i(x, y)$ for $i = 1,2$ must respect the following relationship:

$$u_1(x, y) = u_2(x, y) = \phi(x, y) \quad x, y \in \mathcal{A}.$$

$\phi(x, y)$ is the unique utility function that characterises the 2x2-symmetric game.

	0	1
0	(a, a)	(d, c)
1	(c, d)	(b, b)

Table 2.2. Payoff matrix of a symmetric 2x2-games with an action set $\mathcal{A} = \{0,1\}$ and a, b, c, d represent the reward values.

2.3.1 Prisoner's dilemma

The *Prisoner Dilemma* is one of the most famous games in literature and offers an example of inefficient Nash equilibrium. In the classic description of the game, provided by Luce and Raiffa, the two players are two criminals arrested on charges of committing a serious crime. The two are locked in separate rooms for questioning and have no opportunity to communicate. Both players have two possible actions: they can confess betraying the partner, or not confess. So, the action set is $\mathcal{A} = \{C, NC\}$. The possible outcomes are three:

1. the prisoners confess that they committed the crime. In this case, both receive a severe penalty;
2. both decide not to confess and are sentenced to a light penalty;
3. one of the two decides to confess, while the other one does not confess. In this case, the first prisoner (the betrayer) is released, while the second one receives the most severe penalty.

The payoff-matrix is represented in 2.3 and the utility function is described in terms of years in prison. The aim of both prisoners is to minimize the penalty. So, prisoner's dilemma could be interpreted as a cost game.

	C	NC
C	(6,6)	(0,7)
NC	(7,0)	(1,1)

Table 2.3. Payoff matrix of Prisoner's dilemma.

In this case, it seems obvious that for both the best solution would be to not confess, i.e. the configuration (NC, NC), because both would spend just a single year in prison. Instead, it is possible to prove that the unique Nash equilibrium is given by the scenario in which both confess, i.e. the configuration (C, C), which implies both spend 6 years in prison. In fact, in terms of best response of the player 1,

$$\mathcal{BR}_1(NC) = C,$$

i.e. if prisoner 2 decides to not confess then prisoner 1 should confess because $u_1(C, NC) < u_1(NC, NC)$. At the same time,

$$\mathcal{BR}_1(C) = C,$$

because $u_1(C, C) < u_1(NC, C)$. Since the game is symmetrical, the same results apply to $\mathcal{BR}_2(NC)$ and $\mathcal{BR}_2(C)$. So, the choice to confess is a dominant strategy in the game for both players and the system converges on the strict Nash equilibrium (C, C). This example shows that if players make rational choices for their own gain, it does not imply that they will ultimately arrive to an optimal configuration. In fact, as already observed, configuration (NC, NC) is a better solution than (C, C).

2.3.2 Coordination game

A *coordination game* game is a class of 2×2 games in which players get a higher payout if they perform the same move. Therefore, playing two different strategies is disadvantageous to both. The 2×2 coordination game is characterised by the existence of two Nash equilibria that coincide with configurations in which the action of player 1 is the same of player 2. The payoff matrix is illustrated on Table 2.4 and entries must satisfy the following conditions

$$a_i > c_i, \quad b_i > d_i \text{ for } i = 1,2. \tag{2.3}$$

	A	B
A	(a_1, a_2)	(d_1, c_2)
B	(c_1, d_2)	(b_1, b_2)

Table 2.4. Payoff matrix of a generic coordination game.

Let us now calculate the best response functions for each player:

$$\mathcal{BR}_i(A) = A \text{ because } a_i > c_i \text{ for } i = 1,2; \tag{2.4}$$
$$\mathcal{BR}_i(B) = B \text{ because } b_i > d_i \text{ for } i = 1,2. \tag{2.5}$$

So, the two Nash equilibria in a 2×2 coordination game are the configurations (A, A) and (B, B).
If the hyphotesis $a_1 = a_2 = b_1 = b_2$ is added to conditions in (2.3), the game is called *pure coordination game* and both equilibria are Pareto efficient.
Let us consider another classical example of coordination game in literature referred to as the *Battle of sexes*. The scenario involves a couple deciding how to spend a night during the weekend. Both offer an activity to do together and decide to be directly at the appointment place. The man wants to go to the stadium to watch a game, while the woman prefers to go to the theatre to see a ballet. The rules of the game state that:

- if the couple goes in different places, then their expected utility is 0;

- if the couple spends the evening together both will have a positive benefit, but depending on the place the profit will be greater for those who had proposed it.

The payoff matrix of this game is shown in 2.5.

	A	B
A	(7,5)	(0,0)
B	(0,0)	(5,7)

Table 2.5. Payoff matrix of *Battle of sexes* game with actions $A = stadium$ and $B = theatre$.

As aforementioned, both equilibria are Pareto efficient, but each player has a preference. In this case there is not a rational criterion to establish which point will reach the game.

2.4 Schelling point

The concept of Nash equilibrium loses efficiency when in a game there are more equilibria in pure strategy and the two axioms of payoff-Dominance and risk-Dominance are not able to identify the possible equilibrium to which the system will converge. In 1960, Thomas Schelling proposed the *focal-point theory* in his manual *"The strategy of conflict"*. The theory is based on the assumptions that players have a common interest to pursue and are in total absence of communication.
Let us consider the pure coordination game that we explained in the previous section. In this game, there are two pure strategy Nash equilibria which guarantee the same payoff for both. By hypothesis, players choose the move simultaneously and they do not have the opportunity to agree before. Both should predict the expectations that the other would have on themselves. According to Schelling, if there is an equilibrium point with any element that distinguishes it from the others, then all players could notice it and therefore that point could be chosen automatically.

Definition 7 *The Schelling or focal point of a game with multiple equilibria is an equilibrium point that possesses some property that distinguishes it from others.*

	A	B
A	**(10,10) $**$	(0,0)
B	(0,0)	(10,10)

Table 2.6. Payoff matrix of a different version of the pure coordination game.

As an example, let us consider the pure coordination game with the payoff matrix 2.6. Then, it is reasonable to assume that both players will choose the move A.
We know that the equilibria of the system are the configurations (A, A) and (B, B) , but the presence of the star patterns around the first configuration catches more attention. Thus, each player, in choosing the strategy, will assume that the other had the same feeling from looking at the payoff matrix and will consequently play move A with higher probability.
From a mathematical point of view, this new representation of utilities does not change anything, but makes one of the two equilibria *recognizable*.
The problem of recognizing a Schelling point is not trivial. Generally, when the number of equilibria is very high, multi-disciplinary factors also come into play, such as the psychology of the players or the cultural norms to which they are subject.

Finally, we observe that a Schelling point must also be a Nash equilibrium of the game.

In the prisoner's dilemma, the Nash equilibrium of the system is the configuration in which both players confess. Looking at the payoff matrix 2.3 someone might think that it is most convenient not to confess for both, and therefore (NC,NC) is a Schelling point. The hypothesis of rational agents prevents this scenario, because if one of the players is able to predict that the opponent will not confess then he is inclined to confess since it would guarantee a better outcome. Consequently, the Schelling point coincides with the Nash equilibrium.

2.5 Schelling game in blockchain applications

Schelling's theory has inspired the development of several voting mechanisms in blockchain-related applications. The existence of a focal point allows players to find a common solution in the absence of communication. This problem is analogous to coordination of multiple users within a trustless environment, which is typical of blockchain.
Let us define a class of games based on the Schelling point referred to as *Schelling games*. A Schelling game is a pure coordination game with at least two players and a finite set of actions or options to choose from. The goal of each player is to vote for the option that is predicted to be voted by the majority of the other players in order to receive a reward. If we consider a set of binary options X and Y and a reward p for the option

voted by the majority, then a player's payoff matrix is in the form of Table 2.7. As we have

	X wins	Y wins
Player votes X	p	0
Player votes Y	0	p

Table 2.7. Payoff matrix of a player in a simple Schelling game.

already mentioned, this game has two Nash equilibria, and both can occur with the same probability. Instead, according to Schelling's theory, an option may have some property that is able to coordinate players. In this case, players focus on the option they believe to be true and fair. The Schelling point of the game is detected by the truthfulness of the option itself.
This mechanism is considered reliable because in the absence of communication each player tends to vote honestly because they expect that others will also act in the same way. Moreover, the economic incentive reinforces their believe by inhibiting them from malicious behaviour.
Most dispute resolution protocols on blockchain are built on a Schelling game. Jurors or arbitrators must vote on a ruling from a finite set of options. The outcome of the dispute is determined by the so-called plurality voting mechanism whereby the option with the highest number of votes wins. Plurality is particularly effective when there are two choices, so all dispute resolution platforms involve the presence of a binary choice.
Let us now examine the issues that can arise in a voting mechanism built on a Schelling game. The first defect is the vulnerability to Sybil attacks. In fact, a malicious user may create a large number of fake accounts in order to vote several times and influence the final outcome. This problem can be easily solved by introducing a PoS-type mechanism to access the vote.
The second weakness is that a malicious user might pre-announce his vote to other players. In this case, the other voters are conditioned to vote for the attacker's option for fear of missing the reward. A possible solution could be the introduction of a bit commitment protocol to hide votes and penalise those who reveal theirs before the established time.

2.6 p+ϵ attack

The $p + \epsilon$ *attack* is a bribery attack that modifies the economic incentive structure of a Schelling game. The attacker's purpose is to manipulate the focal point of the game by coordinating other players toward a known malicious option. In order to do this, the attacker agrees to pay a bribe of higher value than the reward to the players who voted

for the malicious option if the fair option wins at the end of the game.
We assume that, in the Schelling game represented in Table 2.7, X is the correct option that everyone would vote for, while Y is the malicious one. Therefore, if an attacker has an interest in making Y win, then he can reliably commit to paying a bribe of value $p+\epsilon$ to players who vote Y when the outcome of the game is "X wins". The payoff matrix for this attack is the following:

	X wins	Y wins
Player votes X	p	0
Player votes Y	$p+\epsilon$	p

Table 2.8. Payoff matrix in the case of a $p+\epsilon$ attack.

Voting Y becomes the dominant strategy of the game. It is convenient for a player to vote Y regardless of what the majority will vote for. Indeed, if the majority votes X the player who accepts the bribe receives a higher reward, while if Y wins everyone receives the reward stipulated by the game. The peculiarity of the $p+\epsilon$ attack is that when it succeeds, the cost is zero because the attacker pays the bribe only if the attack fails.
As we mentioned earlier, the attacker must be able to guarantee the payment. The most reliable method is to program a smart contract where the bribe is deposited inside, that manages the bribe's transfer to ensure that the player who proposed the attack cannot extricate himself from the agreement. Consequently, the $p+\epsilon$ attack may require a large budget.
Some possible solutions to limit this attack are:

1. introduction of *deposits*: the game requires the player to deposit a sum d in order to participate in the voting phase. The deposit adds an in-game penalty system which provides an additional incentive to vote fairly. In fact, the player loses d if the vote is inconsistent with the final outcome. In the scenario of a $p+\epsilon$ attack, the reward of a player who accepts the bribe must take into account the loss of d. Consequently, the attacker will have to offer a bribe that is at least equal to $p+d+\epsilon$. If the required deposit is high, the attacker must also commit to a very high capital making it harder and riskier to perform the attack.

2. *appeals mechanism*: appeals allow the outcome of a vote to be postponed. The winning option of the game is the one receives the most votes in the final appeal. Therefore, this mechanism assumes that it is not enough for an attacker to corrupt one round, but that he or she must also be able to corrupt subsequent rounds. If in addition each new appeal increases the number of voters, as in Kleros where the number of arbitrators is doubled, then the attack becomes very costly if not

impractical.
The use of an appeal system also has weaknesses. The first one is that if the attack occurs in the last round then the appeal system is meaningless. The second one is that if players think that in later appeals there may be a $p + \epsilon$ attack, then it is also convenient for them to sustain the attacker in order to be rewarded.

3. *dispute against attackers*: a player may decide to initiate a dispute against the attacker who proposed the bribe. The winner of the dispute gets the other player's deposit. The aim is not only to discourage a possible attacker, but it is also to encourage players to expose the briber.

In conclusion, an effective solution that can prevent these attacks does not exist. The $p+\epsilon$ attack is a real issue that every platform with a voting mechanism on blockchain has to deal with. In the next chapter we will see how the specific protocols of dispute resolution platforms try to prevent this attack.

Chapter 3

Blockchain Dispute Resolution Platforms (BDRPs)

The following chapter is focused on *Blockchain Dispute Resolution Platforms*, also referred to as *BDRPs*. The purpose of these platforms is to provide solutions based on blockchain technology for dispute resolution. In particular, we have identified three platforms that we consider the most relevant in this sector: *Kleros, Aragon* and *Jur*.
The first part of this chapter provides an overview of the dynamics that led to the development of BDRPs. In advance, two main factors have been identified: the recent growing interest in *Online Dispute Resolution* (*ODR*) methods and the need to build tools to address new classes of disputes associated with blockchain technology and not solvable through traditional dispute resolution mechanisms.
In the second part of the chapter, we discuss in detail how the aforementioned platforms settle disputes.

3.1 Dispute resolution question

3.1.1 Overview on classical dispute resolution approaches

The resolution of commercial disputes has been historically relied on by *national* and *international courts* or methods known as *Alternative Dispute Resolution* or *ADR*.
Courts provide a professional treatment of the case. Judges are entities that are recognised as competent and able to maintain impartiality by a State. Furthermore, procedures must follow strict rules defined by the law. The main advantage of resolving a dispute through a court is that the quality and enforcement of the ruling is guaranteed by a government. Moreover, the parties have the right to appeal a judge's decision. We also have to highlight several problems related to the slow speed of proceedings and the high costs required as court fees. Indeed, the timeframe for a ruling is usually very long and the excessive bureaucracy required by a state may generate further delays. The continuous postponement of a judgement leads to an inevitable increase in the costs of the entire process, which must be added to the court fees. In addition, litigants do not have the possibility to organise the meetings according to their own availability since the timing is imposed by the court. A final issue is related to the privacy of litigants. Indeed, a court deals with cases in public without preserving the confidentiality of those involved.
As an alternative to courts, parties may decide to rely on *Alternative Dispute Resolution* or *ADR procedures*. This approach allows disputants to settle their dispute through an out-of-court process involving an expert who acts as a trusted third party. A first difference of ADR methods compared to courts is that they offer parties a privacy-friendly confidential service. All procedures are held in private. Therefore, the management of the dispute is out of public domain. A second difference is that these methods offer greater flexibility in finding more creative solutions that cannot be achieved through a litigation where judges must strictly follow procedural rules. An ADR mechanism makes the parties in the dispute more involved. Disputants have the possibility to manage appointments according to their availability in order to avoid inconveniences due to busy schedules. A second advantage is the parties' freedom to choose who will manage the process. In fact, litigants may evaluate on the basis of reputation and expertise the best profile for their dispute.
Next, we point out that ADR procedures are bounded in time, which means that disputes are resolved on average in less time. Finally, the costs of an ADR service are generally pre-defined, so the parties, before the dispute begins, generally have an expectation of the final fees to be paid. All these factors together make ADR protocols cheaper and allow more disputes to be resolved than through the courts. The ADR modalities we are focusing on are *arbitration* and *mediation*.
Arbitration allows parties to resolve the dispute through the figure of the arbitrator, which provides a binding ruling that is enforced at the end of the litigation process. The enforcement of the verdict is guaranteed by the competent authorities of the jurisdiction where the arbitration body operates. The solution through arbitration is closer to court,

but disputants may not necessarily appeal the final judgement.
Mediation, on the other hand, is a process involving a third party, referred to as a mediator, who communicates with both litigants with the aim of reaching a mutual agreement. Once the mediation proposal has been generated, the disputants can accept or reject it. The proposal becomes legally binding if both sides mutually accept the agreement generated through the mediation service. It is clear that the main difference between the two methods is that arbitration guarantees an award, whereas mediation may fail. Parties who rely on a mediator generally want to maintain a positive working relationship with the other party.
The ADR approach has historically supported the judiciary's work because it reduces the pressure on the courts. On the other hand, the digital revolution of recent years has introduced new problems and challenges in the legal sector.

3.1.2 The growth of Online Dispute Resolution (ODR) methods

The last two decades are characterised by an increasingly widespread use of the Internet, which has led to a significant growth in the volume of online interactions. In particular, we observe the emergence and expansion of the e-commerce sector, which frequently involves *cross-border small trade transactions* between users. As a result, the number of disputes has increased, and from the beginning traditional methods that we described above have proven obsolete and not suitable for this new context.
The first cause is related to the time required to resolve a dispute. The Internet is a dynamic and evolving environment that requires fast solutions. As we mentioned, courts provide a service based on very long timeframes which are often unsuitable. Next, the cost of settling a dispute through a tribunal or an ADR method is high compared to the amount of money involved in a transaction, which is generally small. Even if courts specialised in this type of dispute were established, commission fees in most cases turned out to be about half of the dispute's value.
A further limitation of traditional methods is geographical distance since many transactions are cross-border and both courts and arbitration bodies require the physical presence of the parties concerned during the litigation.
E-commerce companies, in order to increase trust in their services and therefore encourage a widespread use, needed to develop new dispute resolution solutions introducing *Online Dispute Resolution* or *ODR protocols*. We define an *ODR method* as the array of fully online extra-judicial procedures and technological tools designed to resolve conflicts between consumers, merchants and companies concerning agreements stipulated via the Internet. These protocols proved to be more efficient and cost-effective than courts or ADRs.
The first advantage is that the physical presence of the parties is not required. In cross-border disputes in particular, organising meetings between the parties is much easier and cuts out the costs related to transportation from one place to another. ODR platforms are designed to be easy for users to understand. Typically, it is sufficient to compile an online

form through a guided process in order to open a dispute. A further advantage of ODR protocols is that through the technology several procedures are automated, accelerating the time required to handle a dispute. As a result, these mechanisms are able to process and resolve more disputes than traditional methods. Online procedures are also cheaper than in-person ones and allow the redress of disputes involving small amounts which are unprofitable in courts or via ADR methods. The combination of these factors has made justice more accessible. One of the most popular and widely used e-commerce-related ODR platforms is the *eBay Resolution Centre*, which resolves millions of disputes worldwide per year.

The success of these new systems has subsequently caught the attention of institutions such as states and international communities which started to develop their own platforms in order to support courts. We cite as an example the *European Dispute Resolution Platform*, which in total has resolved about 130,000 disputes since its foundation. Another example of an ODR platform is the *Civil Resolution Tribunal* or *CRT* which is an American body specialising in civil disputes. Within the site, we can access the fees for this service, which in total vary from a minimum of $125 to a maximum of $250. The disputes resolved by this platform are mostly claims worth less than $3,000, but the range is expanding in last years. The CRT, similarly to its European equivalent, resolved approximately 160,000 disputes.

Technological progress has provided an opportunity to improve dispute resolution methods. Systems we mentioned above are centralised, meaning that they are managed by a single central authority, which can be a state or a commercial online company. From this point of view, the development and increasing adoption of blockchain has the potential to contribute towards the innovation process of the legal sector through the development of new decentralised protocols.

3.1.3 The benefits of blockchain technology in dispute resolution

More recently, various *Blockchain Dispute Resolution Platforms* have emerged. The purpose of BDRPs is to create an international legal system built on technology-based solutions that makes justice more accessible by reducing costs and delays, that settles disputes in a fairer and more transparent manner and, lastly, that acts without the presence of central authorities. Blockchain and smart contracts represent the tools needed to begin this revolution in the legal sector.

The main advantages that blockchain technology can bring to the legal sector are the following:

- Blockchain is a tool that facilitates data management. In dispute resolution, transparency and immutability properties of the transactions recorded inside blocks allow the construction of a history of the entire legal process. As a result, the parties are able to independently verify that each transaction has been carried out correctly.

- One of the focal points is related to the enforcement of the ruling. We have previously shown that traditional systems have a central authority that guarantees the parties the execution of the verdict. In BDRPs, this task is entrusted to smart contracts, which automate the enforcement of the ruling when a solution to the dispute is declared final and the process is irreversible.

- Smart contracts not only ensure the enforcement of the ruling, but also manage the whole dispute. The automation of processes through this technology reduces costs and increases speed as it bypasses bureaucracy. In crowdsourced protocols, smart contracts optimise the efficiency of jury selection, vote counting and electronic payments including the management of fees and the final distribution of rewards to judges.

- Public key cryptography guarantees the privacy of all parties involved in the conflict. We showed that blockchain users are pseudo-anonymous. Thus, BDRPs platforms allow disputes to be settled openly by publishing conflict-related data such as evidence or the status of the litigation while keeping the parties' identity hidden.
  Moreover, the identity of the jurors is also hidden, which is one of the main advantages of blockchain-based dispute resolution protocols. The result is that jurors receive more protection from intimidation attempts and they are free to express their concept of fairness unaffected by outside influences.

We identify two different kinds of services provided for blockchain dispute resolution.
The first one consists in assigning the dispute to professional arbitrators. The most representative platform offering this conflict resolution mechanism is *OpenLaw*. It is a project which facilitates the establishment of legal agreements on Ethereum through the drafting and implementation of smart contracts.
The second class of services for blockchain dispute resolution refers to the aforementioned *crowdsourced arbitration protocols* which are the ones of interest in our discussion. These models consist in addressing the dispute to a randomly drawn group of untrained volunteer jurors who prove to find a fairer solution in less time according to the theory of the *wisdom of the crowd*. The theory states that a large group of independent agents with different opinions holds a collective knowledge that achieves better results in decision-making and problem-solving than individual experts. As we mentioned in the previous chapter, protocols include jurors who independently vote among a finite set of options. Unlike professional arbitration which guarantees quality judgments through the reputation of arbitrators, crowdsourced protocols are based on a combination of game theory and crypto-economic system of rewards and penalties that incentivise jurors to vote fairly. By consequence, the whole legal process may be modelled as a Schelling game having a Schelling point corresponding to the ruling considered by the majority to be the fairest. These models are inspired by the legal tradition of ancient Greece where conflicts were settled by jurors drawn randomly from the population and who were remunerated if their

contribution to dispute resolution proved significant.
We observe that, in both scenarios, protocols are designed to provide arbitration services. In fact, the developers of these platforms consider arbitration to be the dispute resolution mechanism that best enhances the self-executing features of a smart contract. On the other hand, in the next chapter we will propose two mediation protocols, justifying how, from our point of view, mediation is more suitable with the original concept of blockchain technology.

3.1.4 The need for On-Chain tools in blockchain-related dispute resolution

Until now, we have presented BDRPs as an alternative or an improvement to existing ODR mechanisms. In this section, we approach the discussion from a different point of view. Blockchain is a relatively novel technology which has not been massively adopted yet. In the same way as all innovations, the emergence of the blockchain ecosystem has led to the rise of a new class of disputes which cannot be resolved through traditional methods. Accordingly, we will investigate the reasons that led to the development of dispute resolution mechanisms on-chain.
Users of a blockchain network are able to enter into commercial agreements through a smart contract that binds the parties to their obligations through code automatic execution. A problem arises when the smart contract is triggered improperly or else does not run when it should. The causes may be various, such as a bug generated by an error in the programmer's code or the malfunction of an oracle that reports a wrong value. The consequence is that contract participants risk to incur a large financial loss.
In these cases, the parties are not safeguarded because they cannot entrust a dispute to a court. The main reason is that there is no regulation or clear position on smart contracts by legal authorities which consider such agreements to be outside the law.
In addition, other issues must be taken into account. The actors involved in a smart contract are anonymous and may be spread around the world. Courts require identification of the parties, but in an anonymity context this is not always possible. Moreover, establishing which jurisdiction should be applied to the case is also difficult.
A second problem is the coding nature of smart contracts that would require dedicated programmers in order to translate the underlying agreement into understandable legal language for judges.
Moreover, once the code is executed, transactions performed are unrecoverable. As a result, even in the event that a court should approve the offended party's request, we do not know how it could restore the balance or more generally enforce a judgment.
All the earlier considerations led some blockchain users to design dispute resolution mechanisms specifically for smart contracts. The most consistent solution is to try preventing the code enforcement. Therefore, BDRPs allow parties to draft smart contracts with clauses that can temporarily block execution and set up an out-of-court dispute.

A further defect of smart contracts is the excessive objectivity required in the terms that trigger them. In fact, a margin of subjectivity is either an advantage or inevitable in many business agreements. A dispute resolution mechanism built into the smart contract can introduce the concept of subjectivity, allowing the parties to review the previous agreement and terms.
A further application of BDRPs is the governance of DAOs. Indeed, decisions within these decentralised organisations are reached collectively. All users do not always agree and disputes may arise. BDRPs offer voting tools to DAOs in order to manage these conflicts. The most representative platform in resolving this class of disputes is the AragonCourt, which provides its service to DAOs developed in Aragon and will be discussed in more detail in Section 3.3.

3.2 Kleros

Figure 3.1. Kleros logo.

Kleros is a fully decentralised Ethereum-based dispute resolution platform. Kleros' project stems from the desire to create a new form of decentralised justice that is able to achieve fairer rulings in a shorter time and lower cost than traditional methods.
The dispute resolution protocol involves a crowdsourcing system to assemble a group of anonymous jurors who will produce a ruling through a Schelling game scheme. Therefore, jurors have a finite set of options and must vote for the winning one in their view. The voting system implemented is the plurality system. The latter establishes that the winning ruling coincides with the option that receives the most votes. Jurors who voted with the majority outcome receive a reward, while others are penalized.
The choice for pseudo-anonymous judges provides greater protection against corruption attempts like bribes and collusion. The protocol provides that the ruling can be appealed. The number of jurors involved in the dispute increases with each new appeal. The final ruling coincides with the outcome of the last appeal, invalidating the results of previous votes. In addition, the protocol does not exclude that a juror may be selected several times for voting within the same dispute.

3.2.1 Use cases

The services offered by Kleros are several:

- **Arbitration**: this service is the basis of the platform and represents the application of greatest interest for our study. Two parties can open and settle a dispute in the *Kleros Court*.

- **Oracle**: The Oracle service allows to use the dispute resolver provided by Kleros as a decentralised oracle. Thus, users can verify real-world phenomena on-chain through a dispute. This application is mostly used by Reality.eth platform. Through the latter, a user can ask any question and in return for the correct answer it provides a reward. Anyone can propose an answer by depositing a bond. If a user believes the answer is incorrect, then they can overwrite a new one by depositing double the bond linked to the current answer. The process continues until no more changes occur.
  If the requester has inserted an arbitration clause, any participant can activate it at any time during the process. The outcome of the dispute will report the final answer.

- **Curated List**: Kleros arbitration is also used to compile lists having specific properties. At any time, a user can enter temporarily a new item in a list by depositing an amount of cryptocurrency within a smart contract. The proposer receives a reward and the deposit is returned when the item is approved by the community. If another user considers that the new entry does not meet the requirements to be added, then they can challenge the proposer.
  As a result, a dispute arises in Kleros Court between the challenger and the proposer. The outcome of the dispute determines whether the item will be inserted or not. Among the most important lists we mention the List of Humans, which we will return to later for a description of the Proof of Humanity protocol.

- **Escrow**: *Kleros Escrow* is a dApp that allows users to securely exchange multiple assets of various natures in Ethereum ecosystem. The protocol requires that a deposit should be locked in a smart contract and that payment will be released when certain conditions are met. Whenever a dispute arises, the funds deposited are frozen. A jury decides how funds will be redistributed between the parties.

- **Governance**: *KlerosGovernor* is a smart contract that can be inserted into a DAO to manage the governance. Any operational decision can be translated into a set of transactions that are submitted to the Kleros Governor. If any member of the DAO questioned the legitimacy of the operations, a dispute may occur. Next, a Kleros jury intervenes and orders the execution or not of the contested transactions.

- **Linguo**: another Kleros application is *Linguo*, which is a decentralised translation tool. The correctness of a translation can be a matter of dispute. Again, the dispute is transferred to the Kleros Court.

Kleros' platform has been active since 2018 and currently counts more than 1200 resolved disputes and more than 700 active jurors.

3.2.2 Pinakion (PNK), Kleros native token

At the heart of Kleros ecosystem workings there is the *Pinakion* or *PNK*. The PNK is the platform's native token. The total supply is 764,626,704 PNK and this value can only be changed through a vote by the Kleros community. Currently PNKs in circulation are 632,380,856.11. The introduction of a native token has a threefold function.
First of all, ownership of PNK tokens guarantees voting rights in the governance of Kleros. Users are invited to vote for new proposals for the development of the platform, called *Kleros Improvement Proposals* (*KIPs*), or for the management of certain parameters in the Kleros Court. *KlerosGovernor* is the smart contract that manages these governance processes.
Secondly, PNK allows users to participate as jurors through a staking mechanism. The protocol randomly elects judges for disputes. The chances of being selected increases as the number of tokens in stake grows. Whenever a judge is drawn, a portion of the tokens in stake is locked within the Kleros contract. The locked PNKs are returned if the juror votes for the outcome of the dispute. Otherwise, the locked amount is slashed.
In the third place, PNKs increase protection from Sybil Attacks. A malicious juror who intends to manipulate the outcome of a dispute needs enough votes to secure the majority. As a result, the attacker needs to buy a large amount of tokens for staking in order to be drawn several times in the same dispute, but this scenario is equivalent to the possibility of being able to perform a 51 % attack against the Kleros platform.
Kleros developers provide several explanations regarding why is better using a native token instead of an external one, such as Ether.

- PNKs make the attack more difficult to sustain from an economic point of view. A user who buys a large number of PNKs increases their scarcity, and consequently the cost of obtaining additional PNKs rises sharply. By contrast, ETHs are available in large quantities. Kleros represents only a fragment of the Ethereum universe, so a user who intends to buy enough ETH to control the entire court would not produce any fluctuation in its value. As a result, the attack would be cheaper relating to the case in which PNKs are involved.
  In addition, 51% of PNK may not be for sale, complicating the attack even more.

- Using PNK in the staking mechanism makes the attack more risky leading to a large loss for the attacker. The Kleros Court must be considered reliable and fair by the parties to a dispute. If an attacker managed to obtain at least 51% of PNKs in order to manipulate disputes, the platform would lose credibility. As a result, Kleros would be abandoned by both judges and users and the PNK token value would collapse. Hence, the depreciation generates a loss to the attacker, who previously bought the PNKs at a very high price. By contrast, the use of an external token would only damage Kleros if the attack succeeded.

- The Kleros protocol stipulates that whether a user is in possession of more than a half of the network's PNKs, the other users may reprogram the token into a new version of PNK. The aim is to exclude the attacker's balance. The problem with this extreme choice implies that every previously written contract is inherent in the old version of the PNK.

The same considerations may be applied to the other decentralised applications that we will examine later. In fact, the use of a native token for staking management is very common.

3.2.3 Kleros Court structure

Next, we proceed to a description of the court structure. The Kleros Court is organised in a tree structure referred to as the *Court Tree*. Each node presents a court specialized in a different topic. At the head of all courts stands the *General Court*, which also represents the root of the tree. The courts represented by intermediate nodes have a reference to the parent court and may have multiple child courts. Each court is identified by the *subcourtID* number and has a specific configuration. The parameters to be configured are the following:

- *parent* is a value indicating the ID of the parent court;
- *children* is a vector containing the IDs of the child sub-courts;
- *hiddenVotes* is a Boolean value that indicates whether in a court the vote should remain secret at an early stage or not;
- *minStake* is the minimum number of PNKs a user must deposit to be selected as a juror in the court. In addition, if a user stakes tokens in a sub-court, then it is automatically eligible for the parent court. This mechanism defines the paths within the Court Tree. Each path ends in the General Court. Thus, the protocol requires *minStake* of a court to be smaller than the *minStake* of the parent one.
- α is a coefficient used to determine the amount of the stake to be locked;
- *feeForJuror* is the arbitration fee to be paid for a single court judge;

- *jurorForCourtJump* is the maximum number of judges that can be selected by the court. When the parties decide to appeal, the protocol increases the number of judges by double plus one. If there are not enough jurors in a court to appeal, then the selection is expanded to the parent court. *jurorForCourtJump* defines this threshold.

- *timesPerPeriod* is a vector containing the duration of each dispute's phase.

The creation of new courts or changing the parameters of existing ones relies on the governance of Kleros. At present, Kleros Court provides the parties with 23 courts.

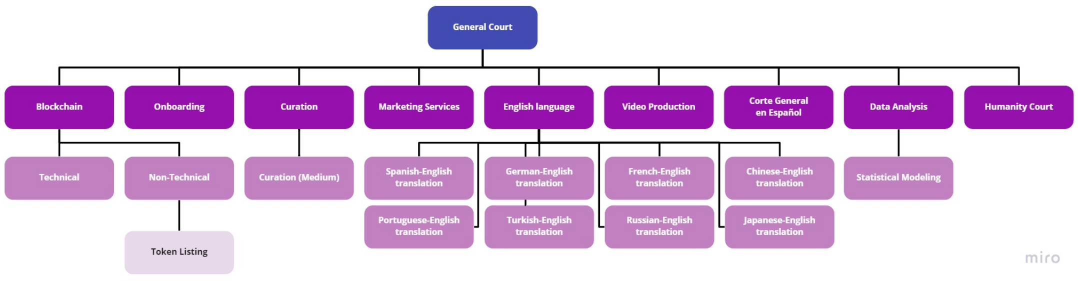

General Court
Blockchain
Onboarding
Curation
Marketing Services
English language
Video Production
Corte General en Español
Data Analysis
Humanity Court
Technical
Non-Technical
Curation (Medium)
Spanish-English translation
German-English translation
French-English translation
Chinese-English translation
Statistical Modeling
Token Listing
Portuguese-English translation
Turkish-English translation
Russian-English translation
Japanese-English translation
miro

Figure 3.2. Representation of the Court Tree. Image taken from Kleros.

Users who intend to act as jurors must at first choose a sub-court, and then stake in it an amount of PNK tokens greater than or equal to the *minStake*, creating a path. In the current Kleros setup, a juror may have a maximum of four stake paths.
Before going into the details of a dispute structure, we present an overview of the *Human Court*. Kleros is developing a new defence mechanism against Sybil attacks referred to as Proof of Humanity. The project is to build a curated list where users who are identified as real humans are registered. The purpose is to exclude fake or duplicate accounts and bots from applications where resistance to a Sybil attack is required. In the case of Kleros, Proof of Humanity is particularly useful because it would allow a single vote to be associated with a juror, whereas at present the protocol allows for the possibility of one judge having more influence than the others. In addition, a user's registration uses the Kleros Court as an arbitration service in case of a challenge, effectively creating a use case for the platform. These disputes are resolved by the Humanity Court where jurors are selected through the Proof of Humanity. The Kleros team suggests additional external applications where users on the list may be involved, such as the governance of a DAO.

3.2.4 Cycle of a Dispute

The Kleros Court offers an arbitration service for those who wish to open a dispute and resolve subjective issues that cannot be resolved independently by a smart contract. The protocol requires an *Arbitrable Side* to send a dispute to an *Arbitrator Side*, which must produce a ruling. For this reason, the Kleros protocol involves two different interacting smart contracts referred to as *Arbitrable* and *Arbitrator*.

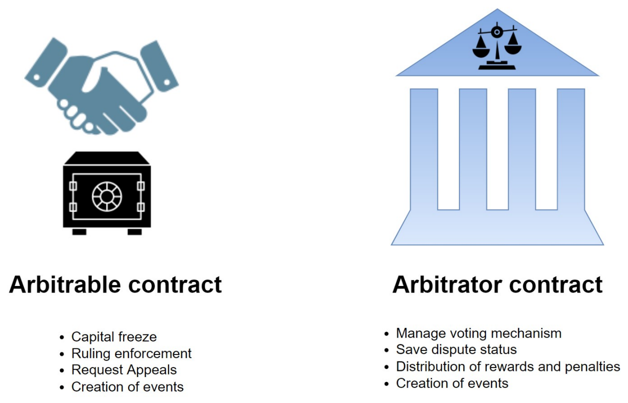


Figure 3.3. Arbitrable and Arbitrator contracts.

The *Arbitrable* contract allows two parties to begin a dispute, request appeals, and execute the ruling. The *Arbitrator* contract manages the voting mechanism that produces the ruling that will be submitted to *Arbitrable*. The advantage of keeping the ruling separate from its enforcement is that it allows the parties not to bind themselves to a specific dispute resolution provider. Then, litigants may decide which dispute resolver is most congenial to their case. Kleros developers introduced the *ERC-792* arbitration standard to simplify the interaction between this kind of contracts. Any *Arbitrable* contract may be arbitrated by any *Arbitrator* contract if they are written with the same standard. The Kleros Court is an *Arbitrator* contract that resolves disputes to users who designate Kleros as their arbitration service.
A dispute in Kleros Court faces five phases, also called periods: *Evidence, Commit, Vote, Appeal, Execution*. The timing of periods is different for each court.

3.2.5 Dispute creation

Two parties open a dispute through the *Arbitrable* contract. The latter invokes the *createDispute* function in the *Arbitrator* contract via a transaction containing arbitration fees. Inside the transaction parties also pass other parameters: the number of voting options, the sub-court, and the number of votes. The initial cost of creating a dispute depends on the sub-court, but in general it is calculated as follows:

$$ArbitrationFee = feeForJuror \times number\_of\_Votes.$$

Kleros platform requires that the minimum number of votes in a dispute is 3 and the protocol requires them to be an odd number. Moreover, the protocol is such that arbitration fees must be paid by the party who loses the dispute. Therefore, both parties deposit in *Arbitrable* an amount of ETH equal to $ArbitrationFee$. After the outcome, the winning party is reimbursed the same amount.

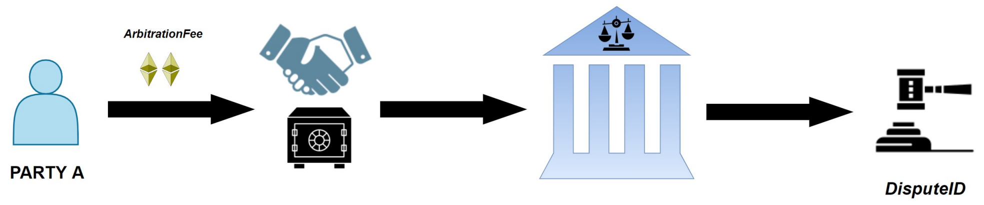


Figure 3.4. Each dispute is identified by a *dispute ID*.

3.2.6 Evidence period

The *Evidence period* begins after the dispute creation. Jurors are randomly drawn, and parties are invited to provide evidence. Let us recall that a user can only be drawn as a juror if they have a stake greater than *minStake* in the path where the dispute began, and that the probability of being selected depends on the amount.
To make the draw efficient, Kleros developers implemented a sorting protocol in a *k-ary Sum tree*. A *k-ary Sum tree* is a tree where each non-leaf can have a maximum of k child nodes, and the content of the nodes is calculated as the sum of the values in their children. In our case, the leaf nodes of the tree represent the jurors of a court and contain the respective staked PNKs. The root of the tree contains the totality of PNKs in stakes present in the court.
The protocol for extracting a juror starts at the root and traverses the entire tree until it lands on a juror. In the first step a pseudo-random number rn is generated by the system. From the root we move to the outermost child node on the left. Two scenarios are available at this stage. If p, which is the value contained in the node, is greater than rn then we move to the tree level below. Otherwise, we compute $rn - p$ and move rightward to the nearest node. The process is iterated until a leaf node is reached.

Figure 3.5 shows the situation of a court consisting of 4 judges A, B, C, D who have in stake $70, 50, 20, 60$ PNK, respectively. In the example, a tree was constructed with $k = 2$ and the pseudo-random number is $rn = 132$. The selected juror in the example is C.
This process is executed a number of times equal to $number\_of\_Votes$. Each time a juror is drawn a portion of PNK in stake is locked within the dispute. Since a juror can be drawn multiple times, ensuring greater weight in voting, the number of blocked tokens

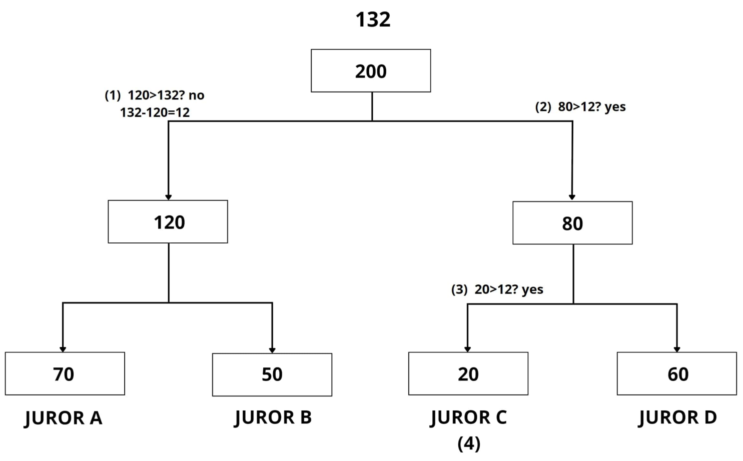


Figure 3.5. Example of a draw round through a 2-ary sum tree.

is multiplied by the number of selections. Thus, the protocol takes from a juror a value D equal to:

$$D = \alpha \times minStake \times number\_of\_selections$$

D is returned to the owner if the juror votes in accordance with the outcome of the plurality otherwise the blocked tokens are redistributed to the winners.
During this period, the parties are invited to provide evidence. Again, Kleros developers introduced the standard *ERC-1497* that can make it easy to switch between one Arbitrator contract and another. We can identify two classes of evidence, the *Evidence* and the *MetaEvidence*. The former allows parties to show their point of view. MetaEvidence gives additional information to the dispute, such as the category or a human-readable description of the options to be voted on. Consequently, MetaEvidence must be submitted when the dispute is created. In the end, let us observe that both MetaEvidence and Evidence are not stored directly in the blockchain. Only their IPFS path is recorded within the blocks.

3.2.7 Commit period

At the end of the Evidence period, Jurors must choose from several options ranging from 0, meaning *"Refuse to vote"*, to the number of options that parties entered in the early stage. Option 0 is provided if a juror finds the evidence sent by parties or the context of the dispute unclear. We specify that *"Refuse to vote"* is considered a vote, so if the outcome does not match 0 then the locked tokens will slash.
At present, most disputes involve two options since the plurality system is particularly

effective for binary votes.
If the sub-court requests that votes must be submitted hidden, the dispute enters the *Commit period*. Else, we move to the next period.
Keeping the vote hidden prevents a judge to influence others, either randomly or on purpose. The risk is that jurors could base their vote on convenience and not fairness. Imagine the case where an attacker is the first to vote and the choice falls on a mischievous option. The other jurors might be pressured to vote for the malicious option as well for fear of being excluded from the reward and losing the stake.
Therefore, some courts require that the vote remains hidden until all jurors have chosen an option. The mechanism used to cast the vote is analogous to the bit commitment protocol seen in Chapter 1. Kleros establishes a hash function h. Each juror randomly chooses a *salt* and computes the value:

$$h(vote + salt + address).$$

In the Kleros yellowpaper, the authors cite *keccak-256*, which is the same function developed on Ethereum.

3.2.8 Vote period

The *Vote period* is developed in two different ways.
If the court requires the votes to be secret, then the jurors must send the pair $\{salt, vote\}$ to the Kleros Court smart contract. The platform computes the hash value by adding the juror's address and checks that the value matches the one committed in the Commit period. The judge who fails to reveal the vote loses the locked PNKs.
Instead, if the court does not require secrecy of votes, jurors choose a ruling and send it to Kleros Court.
In both cases, the platform penalizes judges with lazy behaviour. In fact, those who do not vote within the time limit are penalized in the same way as incoherent jurors.
Next, Kleros computes the outcome of the vote and notifies the parties about the partial outcome generated by that round.

3.2.9 Appeal period

After the Vote period, the Arbitrator issues a ruling. The party that lost the round may decide to appeal the decision and start a new round of voting. In the next round, the protocol includes twice the number of jurors plus one. If this number exceeds the *jurorForCourtJump* value, then the dispute is moved to the parent court.
The appeal is created within the Arbitrator through a function *appeal* that can be called only by the Arbitrable contract. Parties have to submit a transaction containing the amount of arbitration fees used to pay the jurors. Kleros does not specify a fixed number of times a party can appeal a decision. The dispute reaches the final round when the total

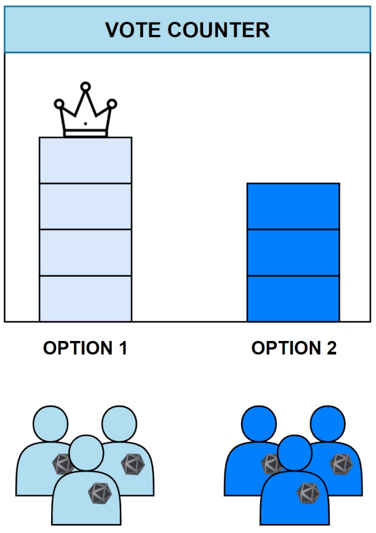


Figure 3.6. Kleros counts the votes and elects the winning option.

number of jurors in the entire platform is exceeded, then the final ruling is made by the General Court.
The cost of arbitration fees to pay jurors for an appeal is:

$$ArbitrationFee = feeForJuror \times (2 \times Number\_jurors\_lastRound + 1)$$

Kleros requires both parties to be able to cover this cost because arbitration fees must be paid by the loser of the appeal. The appealing party also decides on the option they believe is the fairest. The other party is automatically assigned the decision of the previous voting. As a result, both parties must deposit an amount of ETH at least equal to the cost of arbitration fees in *Arbitrable* in order to fund their option during the Appeal period. If only one option is fully funded at the end of the period, then the dispute turns in favour of the respective party without any vote.
Kleros offers the opportunity for third parties, through a *crowdfunding system*, to help litigants in funding appeals. The purpose is to prevent the economic mismatch between the parties from generating a ruling that is not considered fair by the voting system. So, during the Appeal period, any user can decide to partially fund the appeal fees. In return, crowdfunders receive a financial reward proportional to their contribution if the option they fund for wins.
The basic idea is that both parties have to deposit an amount higher than the needed arbitration fees, in order to initiate an appeal. To be specific, the appealing party, who wants to change the outcome of the voting, has to provide a number of ETH equal to:

$$depositAppellant = feeForJuror \times (2 \times Number\_jurors\_lastRound + 1) \times coefAppellant.$$

$coefAppellant$ is a coefficient that indicates how greater must be the appellant's party deposit in respect to the $ArbitrationFee$. On the other hand, the defending party is required to submit a smaller deposit than the challenger, equal to:

$$depositDefence = feeForJuror \times (2 \times Number\_jurors\_lastRound + 1) \times coefDefence.$$

$coefDefence$ is a coefficient that indicates how greater must be the defence's party deposit in respect to the $ArbitrationFee$. So, the coefficients must comply with the following relation:

$$coefAppellant > coefDefence.$$

Thus, the total cost to finalize an appeal is given by the sum $depositAppellant + depositDefence$, but only the value $ArbitrationFee$ will be distributed among coherent jurors. The remainder of the fees goes to the users who contributed to fund the winning outcome.

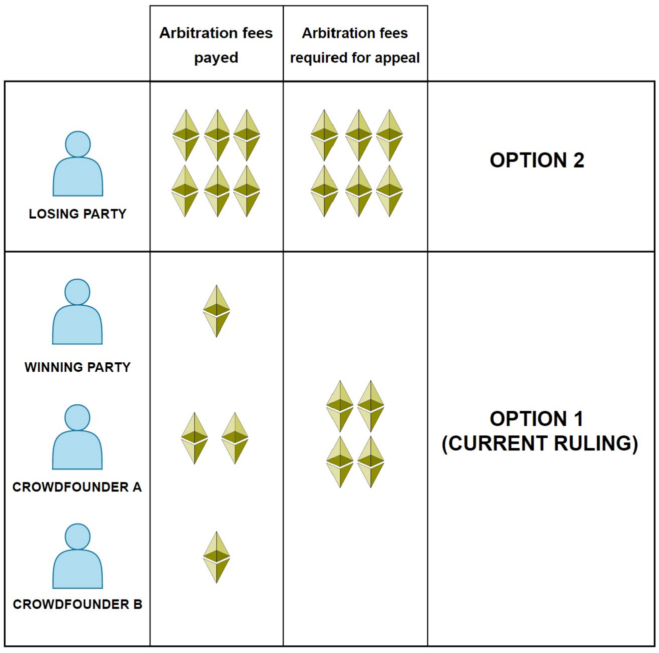


Figure 3.7. Appeal mechanism.

If no option is fully funded, then the previous ruling is maintained.
The creators devised a system such that that the cost increases exponentially with the number of appeals. As a result, a dispute generally does not go further than two or three appeals.
When the appeal is properly created, the dispute returns to the Evidence period where the new jury will be drawn and the parties are allowed to add more evidence.

3.2.10 Execution period

If the losing party does not appeal in the previous period, the ruling indicated by the plurality system is the dispute's outcome. The *Arbitrator* sends the award to the *Arbitrable* which enforces it. Then the *Arbitrator* handles transactions involving jurors and eventually crowdfunders. Jurors who voted coherently with the plurality outcome receive a reward equal to:

$$\frac{ArbitrationFee + depositsIncoerentJurors}{numberCoherentJurors}$$

The value above is calculated for each voting round. If a juror voted for an option other than the final outcome, they lose their deposit, which is later redistributed to the winners.
The entire solving process in Kleros is resumed in the scheme 3.8.

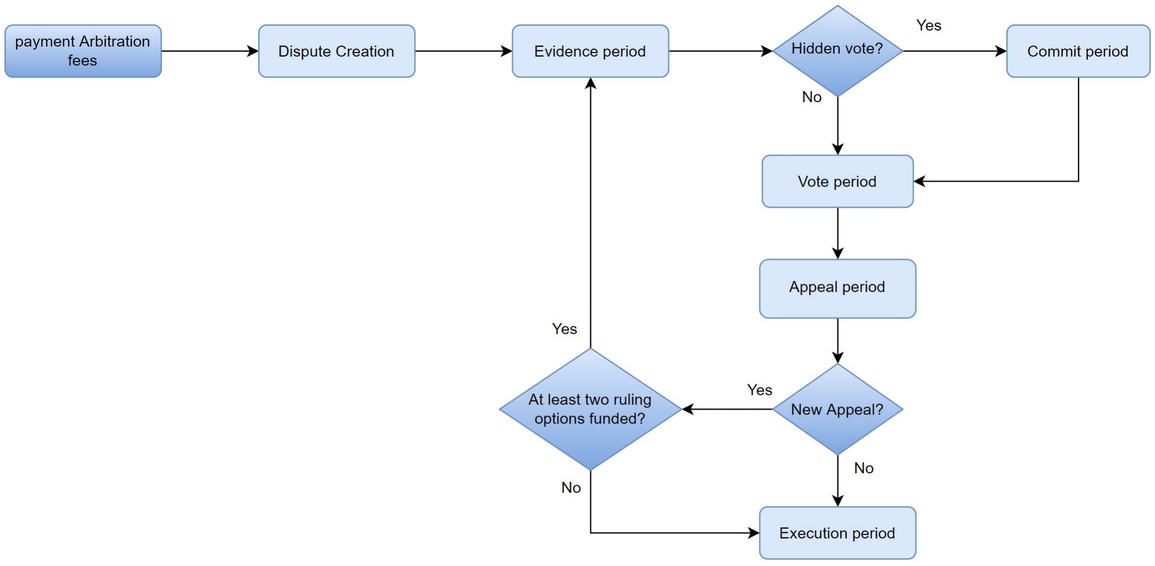


Figure 3.8. Kleros' workflow.

3.2.11 Conclusions

Through the website Klerosboard.com, we can access the disputes settled by Kleros over the years. Such data is helpful in understanding the amount of demand in this industry and if the service has resolved disputes outside the crypto world.
Let us begin with a general overview. Kleros has solved a total of 1266 disputes (data were taken on September 12, 2022) and the platform is divided in 24 courts. There are 773 active jurors available and 525 of them have been drawn at least once. The number of PNK stakes within the Kleros contract is 130,038,448, representing approximately 17% of the total supply. In addition, the total commissions paid to jurors who voted coherently exceed one million dollars.
In the pie chart 3.9, we present the distribution of disputes among the different courts.

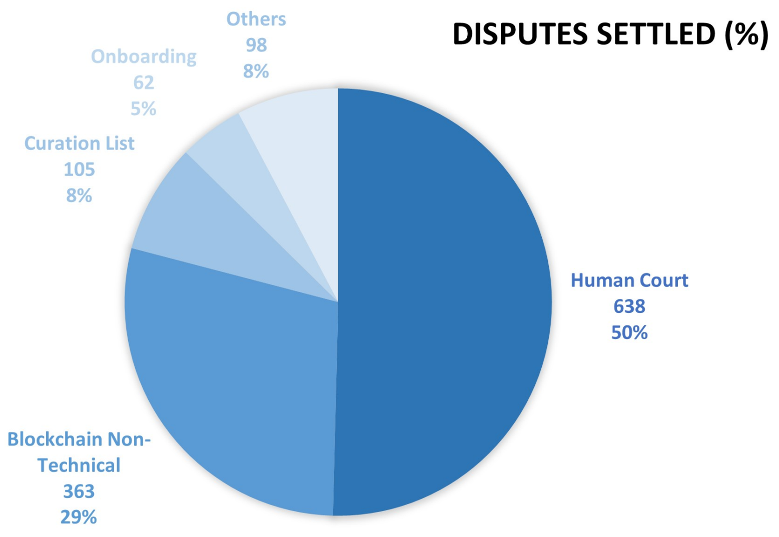


Figure 3.9. Distribution of disputes within Kleros courts.

Let us observe that just over half of the total disputes (638 out of 1266) were settled in the Humanity Court. We recall that the aim of this court is to compile a list of verified users that will be drawn as jurors in later versions through a mechanism called Proof of Humanity which ensures resistance to Sybil attacks. At present, the Humanity Court is the only one that has solved disputes in the last 30 days.
Subsequently, the chart shows that more than 90% of the disputes are limited to just 4 out of 23 courts. Among these there is the Onboarding court, which has the highest number of active jurors in a Kleros court, 228.

The Onboarding court is designed for beginner users who want to practise with the platform and offers the opportunity to participate in simple disputes related to general

topics. The latest litigation settled in Onboarding was about two years ago (Dispute 530, 2020-12-23).
Another disconcerting fact reported in Klerosboard.com is that 11 courts have not yet resolved a single dispute since their creation.
Our analysis focuses on disputes that we define as real-world related, i.e. outside the blockchain environment. On Klerosboard.com, one can view all Arbitrable contracts that interacted with the Kleros Arbitrator smart contract. Among these, we find the class *Dispute Resolver*. The disputes that are resolved with this kind of Arbitrable contracts can be summarised as follows:

- **Refunds**: One party provides a service to the other. Disputes of this type involve the drafting of a escrow contract containing the payment of the customer or a security bond for the supplier.
  Let us take Dispute 574 as an example, which deals with a tenancy dispute. The landlord of a property and the tenant have a monthly rental agreement. In the agreement, a security bond is deposited within the smart contract in favour of the landlord if the tenant is judged liable for any financial loss.
  In the present case, the tenant notified to leave the accommodation outside the time limit stipulated in the contract. As a result, the landlord did not find a new tenant in time and lost the rent for that month and subsequent ones. The landlord held the tenant responsible for the loss and opened a dispute with Kleros in order to claim the security bond. Seven judges were drawn for the dispute and won the option to reimburse the landlord.

- **News verification**: Through this second application, two users discuss about the accuracy of certain reports or tweets from the network trying to detect fake news.

- **Insurance**: A user can use Kleros to manage insurance claims. For example, in Dispute 550 the owner of a property demands compensation from the insurer for damage caused by the tenants' animal. The insurer sends as evidence the contractual terms of the client's insurance showing that their plan does not cover that kind of damage. As a result, the jurors voted not to compensate the landlord.

- **External judgement**: In some cases, the parties try to establish which offer is better among a set of proposals through the resolution of a dispute.
  This is the case in Dispute 541. A couple booked a honeymoon through an agency that organises luxury tours of Europe. Due to bad weather the tour schedule was revised. The couple was not at all satisfied with the service provided and demanded personalised compensation by submitting an offer. The company then suggested a counter-offer to the couple. The jurors involved in this case had to choose which offer was the fairest.

The disputes listed above represent interesting use cases. The problem is that only 83 disputes out of 1266 were settled by *Dispute Resolver* contracts, which amounts to about 6% of the total. So at present, Kleros, which is the most widely used BDRP, is engaged in solving disputes concerning the inclusion of objects within lists. Thus, the lack use of the platform led Kleros to create the same disputes that it will later solve, rendering its impact in the legal world *de facto* ineffective.

3.3 Aragon

Figure 3.10. Aragon logo.

Aragon is a dApp that enables the development and maintenance of DAOs running on the Ethereum Virtual Machine. The Aragon Network is an internal organisation responsible for providing the infrastructure and a range of services to support Aragon users. Governance of the platform relies on a mechanism of proposals and voting by *Aragon Network Token (ANT)* token holders. In addition, the ANT token allows holders to use the Aragon Network services. Our interest is focused on one of these services: the *Aragon Court*.

3.3.1 Aragon Court

Aragon Court is a decentralised dispute resolution protocol based on a Schelling game scheme. The Aragon Network provides this service for the resolution of subjective binary disputes inside Aragon organisations that cannot be expressed under the conditions of a smart contracts.
The ruling of a dispute is achieved through the plurality system. The protocol assumes that jurors, also referred to as *guardians*, are randomly drawn from a pool of candidates and cannot communicate with each other during the voting phase. The vote is requested to be kept secret to prevent one watchman from conditioning the others in any way. Also, in this scheme, the basic assumption is that jurors may converge to the fairest ruling that coincides with the focal point of the system.
We now describe the process of becoming a guardian. Firstly, within the Aragon Court

each user is provided with two ANT token balances. The first balance is defined as *Inactive* and holds the inactive ANT tokens. Instead, the second balance is defined *Active* and contains the active ANT tokens. A user can activate or deactivate their tokens by transferring them from one balance to another. Users who want to participate as guardians must stake ANT tokens in their Active balance. The probability of being selected for a dispute depends on the number of active ANT tokens deposited. Moreover, a guardian can be drawn more than once for the same dispute ensuring a higher weight in the voting process.
Whenever a guardian is drawn, a portion of their ANT tokens in the Active balance is locked until the end of the dispute. A guardian receives a reward in DAI and previously locked tokens are unlocked if his vote coincides with the final ruling. In contrast, if a juror makes a mistake in committing his vote or is part of the minority at the end of the dispute their locked ANT tokens are slashed. The DAI is a Ethereum-based stable coin, i.e. it keeps its value constant, and is pegged to the US dollar. DAIs are generated and released by the decentralised autonomous organisation MakerDAO.
Unlike Kleros, guardians of Aragon Court can earn even if they do not participate in a dispute. The Aragon Protocol releases monthly to the guardians a sum in DAI referred to as *Subscription fees* proportional to the active stake. This mechanism is designed to prevent guardians from leaving the network. Subscription fees derives from the monthly fees that organisations pay in order to use the Aragon Court in case a dispute arises.
The protocol also proposes rewards for users who want to participate in the dynamics of Aragon Court, but that are not necessarily guardians. These rewards are split into two classes. In the first one, a user can earn small sums of DAI through the accomplishment of maintenance actions from the Aragon Court Dashboard. The Aragon Court Dashboard is the offchain interface that allows users to interact with the dispute resolution tool. There, guardians can manage their ANT token balances and view ongoing and already settled disputes. As mentioned above, Aragon Court users receive rewards by performing specific actions in the Dashboard. One example is the draft of guardians for a dispute. A user of the Aragon Court can trigger this process by pushing a button in the Dashboard and receive a reward in return. The second class of recompense allows users to earn amounts of DAI through winning appeals. We will discuss the latter mechanism in more detail within the cycle of a dispute.

3.3.2 Cycle of a dispute

The Aragon Court disputes are divided into three different phases: *Pre-draft state*, *Adjudications rounds* and *Final ruling.*
Before going into the details of single phases, let us introduce the concept of *Court Term.* The *term* is a unit of time measurement used in Aragon Court to organise the timing of operations and phases. A Court Term has a constant duration of 8 hours. Thus, each phase of the dispute in Aragon has a duration in hours equal to a multiple of one term.

The transition from one term to the subsequent one takes place via a transaction called *heartbeat*, which is part of the maintenance actions described in the previous section. Updating to the latest term is essential for the execution of most of the court's functions.
The transition between one term and the next also allows to update guardians' balances. In fact, a user can request to deactivate his active tokens at any time. However, the effective transfer of tokens from the Active to the Inactive balance occurs only at the end of the term. Therefore, a guardian may be selected for a dispute before the deactivation process has been completed. In this case, the tokens will remain active until the dispute is settled.

3.3.3 Pre-draft state

During the *Pre-draft state*, the dispute is created and parties may submit evidence. A dispute is managed through the use of smart contracts. Consequently, if a user wants to open a dispute, they must set up an *Aragon Arbitrable Contract* to be submitted to the Aragon Court. Let us specify that the standard used for these contracts is not the same as the one adopted in Kleros.
After the arbitrable contract address is created, the parties must deposit funds inside the contract in order to cover dispute fees. There are three initial costs to be incurred in creating the dispute:

- *Juror fees* are the fees used to pay the reward to the guardians who vote in favour of the final ruling through the plurality system.
- *Draft fees* cover transaction fees for the guardian drafting mechanism.
- *Settlement fees* are fees that cover the cost of transactions for penalties of guardians who vote with the minority or exhibit unfair behaviour.

As a result, the initial cost to create a dispute is

$$(Juror\_fee + Draft\_fee + Settlement\_fee) \times 3,$$

where $Juror\_fee$, $Draft\_fee$ and $Settlement\_fee$ are values fixed by the Aragon Court setting. The previous costs are multiplied by three because the initial number of jurors is three.
The next step is to set up dispute fields. The structure of the dispute requires one or more *Actions* to be specified. *Actions* are the set of transactions that the contract will execute if the dispute ends successfully in favour of the proposing party. Thereafter, both parties have a 7-day period, referred to as *evidence period*, to submit evidence which guardians will view after the draft.

3.3.4 Adjudication rounds

This is the phase leading to a ruling. In each adjudication round, a group of guardians is drawn to analyse the evidence, participate in the voting phase, and to pronounce a verdict. The protocol allows a ruling to be appealed, and consequently a dispute can have multiple rounds. In the current configuration of Aragon Court, the maximum number of available rounds is four. The first adjudication round begins when the evidence period ends.
In the first phase of an adjudication round, there is the draw of guardians. This time is also referred to as *summon guardians period.* During this period, Aragon Court users can call the function to draw a guardian directly pushing the *Summon guardian* button through the Dashboard. Accomplishing the task ensures a reward in DAI equal to the value of $Draft\_fee$. The process is repeated until jury fulfilment is reached.
Aragon Court implements the same draw mechanism as Kleros. At the beginning of each term, a k-ary sum tree is built from the list of active guardians and their active balance. At each extraction, the protocol locks a portion of active tokens to the selected guardian. The precise amount is 30% of the *minimum Active balance* present in the network at that Term. We observe that a guardian can be drawn more than once for the same dispute. A multiple selection locks a number of tokens proportional to the number of times in which one is drawn. However, that juror will have more voting power than the others.
Let us make two observations. The former is that if a guardian has already been drawn, their Active balance decreases so the probability of being drawn again is lower. In fact, the locked tokens are transferred to another contract, referred as *treasury contract*, that processes the payments at the end of the dispute. The second observation is that tokens in the Inactive budget cannot be locked.
Once the jury has been assembled, the guardians enter into the *voting period* which lasts two days. At this stage, jurors are invited to closely examine the evidence and after that they must commit a vote. There are three possible options:

- *Allow*: if this option wins, the contract executes the *Actions* proposed in the contract;
- *Block*: if this option wins, the *Actions* proposed in the contract are refused;
- *Refuse to vote*: this option was introduced in case jurors feel that the dispute request or evidence are not understandable or they are too vague. If a guardian decides to abstain from voting on one of the two previous options, they must consider that the final ruling is obtained anyway through the plurality rule. So, if another option wins the blocked tokens will be lost.

In the Aragon Court as well as in the Kleros protocol, voting is kept secret to prevent one juror from influencing others in their choice of an option. Also in this voting mechanism, the protocol implemented is like the bit commitment protocol presented in Chapter 1. First, a judge chooses the option to be voted on and then presses the "Commit

a vote" button. The mechanism generates a *one-time-use-code* which is combined with the vote through a hash function. More specifically, the one-time-code-use is used as an argument of the function *keccak-256* to generate the *salt* that is subsequently inserted within a hash function with the vote. Therefore, a juror sends the transaction containing the value $h(salt + vote)$ where h is the hash function used in the Aragon Court and *salt=keccak-256(one-time-use-code).*
The protocol provides a penalty when a guardian leaks its *one-time-code-use*. The cheating guardian is automatically placed on the list of the losing side of the dispute and loses the locked tokens. The platform has a section dedicated to leaks where a guardian can report compiling a form those who published the *one-time-use-code* before the end of the voting period. A juror has an incentive to expose cheating guardians since the tokens subtracted from the penalty are redistributed among the winning side of the dispute.
Guardians can reveal their votes at the end of the voting period. This phase lasts two days like the previous one.

Since the Aragon Court expects guardians to be efficient in resolving a dispute, those who fail in committing or revealing the vote are penalized and their locked tokens are slashed.
The dispute at this stage has produced an initial ruling and the protocol updates the lists of judges who were part of the majority and minority. An Aragon Court user who disagrees with the ruling has two days from the end of the vote disclosure period to make an appeal request. Formally, the appealing party deposits a collateral in DAI and proposes the option that should have won. Now, we need to distinguish between three cases:

1. If no appeal is approved, then the ruling is confirmed and the dispute proceeds to the next stage;
2. Nobody confirms the appeal: the dispute automatically moves in favour of the appealing party overturning the previous ruling;
3. Another user of the Aragon Court confirms the appeal, by submitting an amount of DAI equal to the appealing party's deposit. An appeal has to be confirmed within two days from its request.

In the latter case, the previously generated ruling is frozen, and a new adjudication round begins. With each appeal, the number of judges is tripled from the previous one. The last round, which as we have already mentioned corresponds to the fourth appeal, is an exception. In fact, the number of judges is equal to the value of the ratio of the total active stakes to the minimum active balance within the Aragon Court. The cost of an appeal must be such that it can sustain the costs we have seen earlier for creating the dispute and, in addition, that it can guarantee a win for one of the two appealing parties. For the first three rounds, the cost to appeal a ruling is:

$$(Juror\_fee + Draft\_fee + Settle\_fee) \times 3 \times juror\_number. \qquad (3.1)$$

On the other hand, the cost to confirm an appeal is:

$$(Juror\_fee + Draft\_fee + Settle\_fee) \times 2 \times juror\_number. \tag{3.2}$$

In the final round, costs are reduced by 50%. We observe that the cost increases considerably between appeals. The idea, as in Kleros, is to reduce the time frame for dispute resolution. So. the cost to appeal becomes very high and discourages users from moving forward in the dispute.

3.3.5 Final ruling

When a ruling is no longer appealed, it becomes definitive and the final ruling is sent to both the contract of the arbitration instance and the treasury contract. The former is in charge of executing the ruling. The second manages transactions to either reward or penalize jurors. Guardians who voted for the winning outcome according to the plurality rule receive a reward composed by the Juror fees and the redistribution of the losing guardians' locked tokens.
If appeals have taken place, then the deposit of the losing appealing party is used partly to pay the cost of dispute fees and partly to reward the opponent. The locked collateral of the user who wins the appeal is returned.
In the end, we observe that a dispute can end with two options having the same number of votes. In this case, the setup of the Aragon Court dictates that there is an option with a higher level of priority and it wins automatically unless there are appeals.

3.3.6 Conclusions

The above protocol describes how the new version of the Aragon Court operates. The latter was released in recent months and is still in an early state of development. The number of disputes resolved in this new version is 7. This low use of the court can be justified by the non centrality of the dispute resolver in the dApp. The Aragon team aims to design an application that can resolve any dispute in the DAOs built on the platform that can be represented by a binary output. At present, the main use case is related to *Proposal Agreements*.
A *Proposal Agreement* is a human-readable document in which each Aragon organisation gives dictates on how its members must submit proposals for DAO governance. If a user believes that a proposal violates the terms of the Proposal Agreement, they may decide to open a dispute against the proposer. The previous use case is very similar to the Kleros Governor application.
Aragon Court also aims to provide a service to accelerate the resolution process of a dispute. In the worst case, which includes four appeals, the maximum duration of a dispute is:

$$7 + (2 + 2 + 2 + 2) \times 3 + 2 + 2 = 35 \text{ days.}$$

In conclusion, we observe that the parameters of a dispute are not customizable as in Kleros. Aragon Court provides a unique configuration of parameters. The latter can only be changed by a vote by all Aragon Network users.

3.4 Jur

Figure 3.11. Jur logo.

Jur introduces itself as a legal technology company based in Switzerland that aims to create a well-rounded legal ecosystem for the management of contractual relations.
The Jur platform also has a native token referred to as the *JUR* token. The latter is used both to pay for services offered by Jur and as an economic incentive for jurors of the dispute resolver. Recall that, in Kleros, adopters have to pay services using ETH, while in Aragon they are accessed via DAI.
In recent months, the platform has been undergoing redesign. Initially, Jur was developed on the VeChainThor blockchain, but the developers decided to move to the Polkadot one. Only recently the new *Light Paper*, in which the new guidelines are described, was released. For the purposes of this thesis, we decided to describe the functionality of the old and first version of Jur.
Jur includes the following services:

- *Jur Editor*: The Jur Editor service allows users to easily build Smart Legal Contracts. A Smart Legal Contract is the combination of a legally binding contract written in human-readable language and smart contract code to automate transactions. Users can create contracts from scratch or modify existing templates already in Jur. The process is simplified through *drag-and-drop* features for terms that a user wants to include in the contract. In addition, a clause allowing for dispute resolution with the Jur service is provided in all contracts. The Editor also allows users to be able to propose templates that will be use by other ones. The process is done through a *peer review.* A user must stake JUR tokens in order to propose a template. On the basis of the review result, the user recovers or loses the stake.

- *Jur Marketplace*: The Jur Marketplace allows users to sell their own templates or buy other proposals. The opportunity to exchange templates facilitates the creation of new, more refined contracts. The Jur Marketplace also includes a *legal advice service.* The latter helps users in the creation of their smart legal contracts.

- *Jur dispute resolution mechanisms*: Jur offers to its users the possibility of triggering a dispute in case the complexity of a smart contract makes full automation impossible. The platform distinguishes three different arbitration protocols for dispute resolution referred to as *Court layer, Open layer* and *Community layer.*

3.4.1 Court Layer

The *Court Layer* represents a combination of traditional arbitration mechanisms and the decentralisation offered by blockchain technology. Disputes settled within the Court Layer are legally binding and Jur recommends to adopt this protocol for cases highly complex or involving large sums of money.
Unlike the protocols described above, the Court Layer does not include either a voting system or an appeal mechanism. The arbitrators are drawn randomly and propose a ruling that is submitted to a *peer review* mechanism before being definitive. The peer review assigns a score to the arbitrator within a reputational system that allows to measure the quality of the rulings generated in the Court Layer.
Disputes are resolved within *Arbitration Hubs*, which are digital courts created by any arbitration institution or private body wishing to offer its service on the Jur platform. The creation of a Hub involves a *Hub Admin* staking a quantity of JUR tokens referred to as *Performance Bond.* The value of the bond is proportional to the maximum value of the dispute that can be settled by the Hub and covers the costs due to the proposal of a low-quality ruling or the bribery of an arbitrator.
Each Hub is characterised by rules that must be compatible with international laws or the local laws of the Admin registration address in order to make rulings enforceable.
In addition, the Admin has other parameters to set such as the *minimum* and *maximum number of arbitrators* within the Hub, the *arbitration fees*, how the latter are allocated among the court members, and the *duration of proceedings.*
Another responsibility of the Admin is to equip the Hub with arbitrators. In particular, there are two different methods.
The first one is referred to as *Centralised Selection* where the Admin personally manages the selection of arbitrators.
The second method is called *Decentralised Selection.* In this case, the Admin establishes a set of objective requirements that an arbitrator must have in order to operate within the Hub. An example might be the minimum reputation level required from an arbitrator

or a specialization in a particular topic. A user who meets the requirements and wants to join the Hub must send an application to the Admin and stakes JUR tokens. The latter will trigger a dispute within the *Open Layer* or the *Community Layer* and the outcome approves or rejects the candidate's admission.
The protocol also specifies that an Admin cannot unilaterally remove an arbitrator from the Hub, at least a minimum reputation level threshold may be fixed for the arbitrator to be eligible in a dispute.
Let us describe the steps of a dispute within an Arbitration Hub.

1. The parties must include a clause, designating the Court Layer as the resolver of any disputes arising in their contractual relationship. Moreover, the parties must select in advance the Arbitration Hub that will handle the litigation. If this clause is provided, then either party may open the dispute by sending a request to the referenced Hub and paying the arbitration fees. Thereafter, the counterparty is notified of the beginning of the litigation.
   We observe that in the case of the Court Layer, the presence of escrows in smart contracts is not mentioned. This is because the Jur protocol can potentially resolve disputes arising also from traditional contracts as the judgment is recognised valid by the legal institutions referred to by the Arbitration Hub. As a result, the enforcement of the ruling does not necessarily rely on a smart contract as it used to be for Kleros and Aragon.

2. After the dispute has begun, the next step is the random drawing of the arbitrators. The protocol specifies that for disputes with a value less than USD 150,000, a single arbitrator is assigned, while for disputes with a higher value, the jury consists of three arbitrators. The arbitrators are selected from the Hub in charge of solving the dispute.

3. Initially, the parties represented by their lawyers submit evidence and one or two statements. Then, remote hearings are scheduled and the interviewing of witnesses if there are any. Each Hub can configure the timing of these stages. Generally, the average time to reach a ruling is about 60 days.

4. The juror or jury submits a provisional ruling and the peer review begins. Three jurors within the Jur ecosystem are drawn at random. Each of them must assign a score to the provisional ruling. Reviewers gain or lose JUR tokens based on how close their score is to the median.

5. If the peer review generates a low score, the arbitrator loses reputation points. The case is then reassigned to a new jury and the costs for the new round are paid entirely with Performance Bond funds.
   The protocol allows only one reassignment of the dispute. The new ruling passes through a further peer review, but in this case it becomes definitive.

6. If the peer review produces a high overall score, then the ruling becomes definitive and is enforced.

We observe that the Court Layer's economic incentive system is designed so that the Admin has an interest in building a highly efficient Hub. As a result, the responsibilities are entrusted exclusively to the Admins who are the ones facing economic penalties. The Performance Bond is used to cover additional costs for a new reallocation of the dispute, but it also has a role as a guarantee that the court is corruption-free. In fact, the protocol encourages users to report any attempt at bribery.
Either party may open a further dispute in another hub if during the case they suspect that there was an attempt to bribe the jury. If the complainant wins the case, they receive the full value of the Performance Bond as a reward and the original dispute is reallocated to a new court. Admins also have the opportunity to safeguard themselves, so if one of their arbitrators in the Hub is proven to be bribed, they can report the identity to authorities.
Lastly, we note that unlike previous protocols, the Court Layer does not require arbitrators to be anonymous. On the contrary, the data of arbitrators are public in order to facilitate parties in choosing the best court. From the perspective of the parties, all dispute information is encrypted while maintaining a high level of privacy. Furthermore, dispute details are stored within the blockchain as hash values in order to make verifiable all steps of the proceedings.

3.4.2 Open Layer

The *Open Layer* is the second dispute resolution mechanism proposed by Jur. This layer is designed to resolve disputes of low value, e.g. to deal with *microtransactions* or *small compensation payments* (maximum USD 500). The purpose of the Open Layer is to offer a service that is able to resolve the conflict in about 24 hours and the rulings do not expect to be legally binding.
The Open Layer is designed to resolve disputes related to any type of contract involving an escrow. Two users may create the contractual relationship in the Jur platform by defining a smart contract, but it is not a constraint. In fact, the Open Layer may be used also to settle disputes that arise off-chain.
The mechanism is based on a combination of a crowdsourcing scheme and economic incentives according to game theory, similar to Kleros and Aragon protocols. The main difference with these two is the voting system. Disputes within the Open Layer involve binary options proposed by the parties, and jurors decide to vote by staking their JUR tokens on the outcome they think is fairest, and the option with the largest stake at the end of time wins. The protocol does not require parties to pay fixed fees and the rewards are totally generated by those who voted for the losing option.
Moreover, any user of the Open Layer can take part in any dispute, so the protocol does

not include a random draw of jurors. The basic idea of the developers is to build a system where jurors self-select for their competences and skills and the loss of JUR tokens discourages malicious or unqualified jurors.
Another peculiarity of the protocol is the allocation of rewards. Awarded jurors are those who actively contribute to winning the final outcome. For example, let us assume that in a dispute there are two options *optionA* and *optionB* and that the final outcome is *optionB*. If *stakeA* and *stakeB* are placed on *optionA* and *optionB* respectively, then only those jurors who voted for *optionB* possessing the minimum tokens necessary to overcome the value of *stakeA* are rewarded.
A dispute within the Open Layer goes through the following processes:

1. If there is a clause in the contract to open a dispute within the Open Layer, then one of the two parties may create an option and must fund it with at least 1% of the value of the dispute.

2. The counterparty in turn proposes an option and submits it to Jur's contract. The challenged party may not necessarily finance its own option.

3. Both parties submit evidence in their favour.

4. Users of the Open Layer can stake their JUR tokens on the option they believe to be the fairest and have a time limit set by the parties. By default, JUR proposes a voting phase period of 24 hours, but this value can be modified.

5. At the end of the voting period, the option with the most tokens deposited wins and the tokens in the losing option are reallocated as a reward to the winning jurors through the mechanism described above.

In response to a possible last-minute attack on the majority, the protocol provides for the voting period to be extended.
Another defence mechanism is the inability to deposit JUR tokens in the dispute if one of the two options has a deposit more than twice as large as the other.
In the end, in the Open Layer disputes there is a *Safety Clause* that is automatically activated if a single user exceeds a certain threshold of staked tokens within a dispute. The malicious actor is carried within a dispute in the Court Layer on a charge for attempting a 51% attack on the system.

3.4.3 Community Layer

The *Community Layer* is a combination of the layers described above. The objective is to resolve medium-value disputes between USD 500 and USD 5,000 in short timescales ranging from 24 hours to one week.
From the Court Layer, the Community Layer inherits the division into courts referred to as

Communities. Within a community, parties can open a dispute and the voting mechanism is the same as the Open Layer. Therefore, there are two binary options and the jurors stake their JUR tokens on the option they judge to be the fairest. The difference from the previous Open Layer is that voting is not open to all users of the Community Layer, but only to members of the community the parties rely on.
The *Community Creator* is the one who has the role of creator and manager of the community and recalls the figure of the Hub Admin in the Court Layer. The Creator establishes the parameters of the community. The first is *the minimum and maximum number* of members or voters. Also, in this layer, the recruitment method is left up to the Creator, who can set the requirements for voters. A user seeking to join a community may submit an application to the Creator, who centrally chooses whether to enlist them or launch a dispute in the Open Layer, which outcome dictates the candidate's fate.
Subsequently, the creator can set the *minimum and maximum number of JUR tokens* that community members can hold in their wallets. This constraint prevents one juror from having enough voting power to overrule other members. Furthermore, if a juror receives penalties such that they fall under the minimum token threshold, then they lose their right to vote. This last property is important because, without enough voters, the community enters into a dormant state and disputes cannot be resolved within it.
The Creator may require the payment of an arbitration fee to the parties in order to resolve the dispute. Whereas in the Open Layer the voting mechanism may be represented as a zero-sum game, in the case of the Community Layer the arbitration commission ensures a minimum payout with any outcome. The modalities by which these commissions are allocated are set by the Creator who may also decide to keep fees all for themselves. Fees may be constant or proportional to the value of the dispute.

3.4.4 Conclusions

Jur is the most interesting and ambitious platform among the three we have examined in this chapter. Indeed, dispute resolution mechanisms described above cover the whole range of disputes. The Court Layer is an application designed to replace traditional legal systems. On the other hand, the Open Layer has an interest in resolving the small claims debated in the emergence of ODR mechanisms.
The complexity of the project, however, affected the development of Jur. In contrast to Kleros, Jur has never settled a dispute. Thus, we cannot give a judgement on the service as it was never delivered.

Chapter 4

Aspera and Decentralised Mediation

The aim of this chapter is to introduce two mediation protocols.
The first part is dedicated to Aspera Anonymous Dispute Resolution protocol, which aims to offer as its core service an iterative mediation process that uses an the power of *Artificial Intelligence (AI)* and *machine learning*-based algorithm to propose win-win agreements to parties involved in a dispute. As a supplement for the algorithm, human intervention by a mediator or, in the most extreme cases, an arbitrator may be requested by the parties.
We emphasise that the aim of this thesis is not to discuss the topic of machine learning from a technical point of view. Nonetheless, we will give an overview on the difficulties of this topic that the inventors of Aspera faced during the design phase.
In the second part of the chapter, we propose an alternative mediation mechanism, which is referred to as *decentralised mediation.* This second protocol does not involve an ML algorithm, but it is designed along the lines of the crowdsourcing models we described above.

4.0.1 Benefits of mediation

Let us discuss the choice of mediation more in detail. We identify two different justifications:

- Mediation is more beneficial in commercial disputes. When two parties decide to settle a dispute through an arbitrator's mechanism, the outcome will make one party displeased. The conflict is likely to permanently ruin the business relationship between disputants. On the other hand, mediation attempts to find an agreement that satisfies everyone. We must specify that it is not always possible to reach an agreement. For example, Aspera's protocol also provides an arbitration mechanism that may solve the dispute in a definitive manner. In decentralised mediation, however, this kind of solution is not provided and the chances of the dispute to remain unsettled are high.

- Mediation is a *non-adjudicative method*, meaning it does not force parties to accept a juror's decision. As a result, this system is more closely aligned with the blockchain technology's ideal of decentralisation and aversion to central authority. We have seen that, through arbitration, the smart contract automatically executes a ruling generated by the voting of anonymous jurors. Once the ruling has been issued, the parties are no longer allowed to intervene unless they appeal. In this case, we are referring to an *adjudicative method*. Jurors play the role of *Trusted Third Parties (TTPs)*, which the blockchain seeks to eliminate.

In the previous chapter, we observed that all ODR mechanisms on blockchain offer exclusively arbitration services. Aspera may become the first BDRP dedicated to mediation and this represents a possible advantage over future competitors.

4.1 Aspera

Figure 4.1. Aspera logo.

Aspera is designed for a B2B business model that we find more sustainable in the long run. The aim is to establish commercial agreements that guarantee a periodic payment in exchange for the opportunity to use the dispute resolution mechanism if a conflict arises. In a first phase, Aspera would be implemented on the Ethereum blockchain for development and testing. In the future, the idea is that the protocol could resolve disputes on any blockchain.
Compared to the resolution mechanisms described above, the first mediation protocol we present is not properly decentralised. In fact, the dispute is assigned either to the machine learning algorithm or to a single human actor. In addition, the human mediators and arbitrators are practitioners. The idea is that parties may be more comfortable entrusting

their case to professionals rather than to a group of jurors with no references, in complete contrast to the protocols of the previous chapter.
Nevertheless, we point out that the following mechanism is independent from TTPs since the parties may always decide to reject the submitted mediation proposals. In the case of arbitrator intervention, disputants rely on this kind of solution of their own choice.
Aspera's resolution process takes place through the execution of two smart contracts that we refer to as *Mediable* and *Mediator*. The idea was inspired by Kleros' Arbitrable and Arbitrator contracts for its design and features.

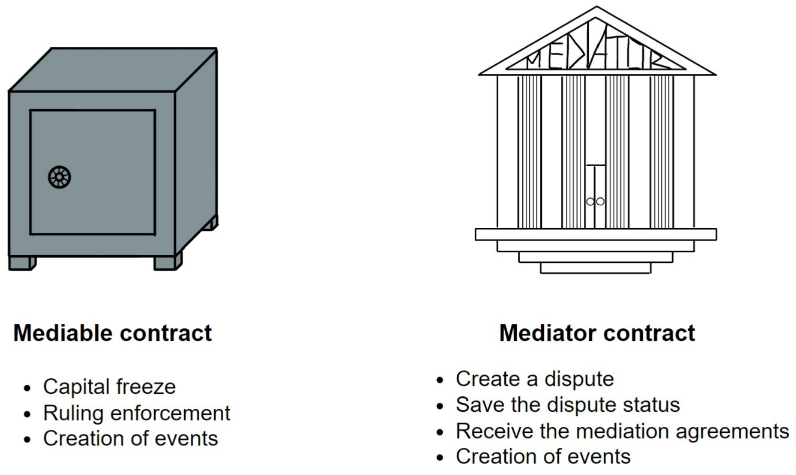


Figure 4.2. Mediable and Mediator contracts.

The *Mediable* contract allows parties to sign an agreement that includes a mediation clause in order to make use of Aspera's service when a dispute arises. This agreement must be able to freeze funds within it as soon as the clause is activated and redistribute them at the end of the dispute enforcing the ruling. The third function that we require is the creation of events recording the parties' commission payments, mediation proposals sent and if those are accepted or declined.
The *Mediator* contract is designed to manage the steps of the resolution process. Within this contract, we define the functions for creating and saving the status of a dispute. In addition, the Mediator contract communicates with Aspera's off-chain platform where there are the core tools for dispute resolution.
The first tool is the *machine learning algorithm*, designed for the formulation of a mediation proposal. The algorithm requires input data to be processed for each dispute. Data is provided by a *chatbot* that interrogates the parties regarding the conflict. Finally, there is a *Virtual Camera* that allows parties to meet anonymously in an online court with the human mediator in search of a more classical resolution.
In any case, the communication of the mediation proposal to the Mediator contract takes

place via oracles. The Mediator contract then sends the result of the resolution process to the Mediable contract and parties decide whether to accept the agreement.
The basic idea is to create a Mediable and Mediator contracts standard similar to the ERC-792 one proposed by Kleros for Arbitrable and Arbitrator contracts. The advantage of having a standard is that it leaves the freedom for parties to choose the best service for their needs if a new mediation platform is later launched in the market.
Let us now describe the Aspera's mediation process. The protocol is structured into four phases and the dispute may be solved at the end of each of them.
The resolution mechanism involves three mediation attempts and, as a last chance, the reaching of a ruling through arbitration. During the first two phases, the attempts are carried out by the machine learning algorithm developed by Aspera. The third phase requires the intervention of a human mediator. If the third attempt also fails, the parties can request an arbitration process to definitively resolve the dispute.

4.1.1 Dispute creation

Parties may trigger Aspera's clause at any time. Activation requires that either disputant sends a transaction to the Aspera's Mediator contract containing their half of the mediation fees and the address of the opposing side. The latter is promptly notified of the dispute request and has two alternatives. If they agree to pay their half of the fees, then the protocol starts from Phase 1. Otherwise, the two parties may agree to move directly to the intervention of the human mediator by paying a higher crypto amount.
Moving from one phase to the next requires an extra fee to be added to the initial cost.

4.1.2 Phase 1

In the first phase, both parties are interviewed by a chatbot, which submits predefined questions. The information is sent to Aspera's database where the data are processed by the machine learning algorithm in order to generate an initial mediation proposal to be sent to the parties.

The questions are of a general nature and are the same for both, e.g. "What is the reason for the dispute?" or "What would be a possible solution to the problem?". The latter is particularly helpful because the algorithm may start from the proposals of both parties to formulate the mediation proposal to be submitted.
Subsequently, the protocol provides for the disputants a deadline, which could be 24 hours, to accept or reject the proposal. If the parties accept the mediation, the dispute ends and the Mediable contract enforces the judgment. If at least one of the parties rejects the offer, then the first mediation attempt fails.

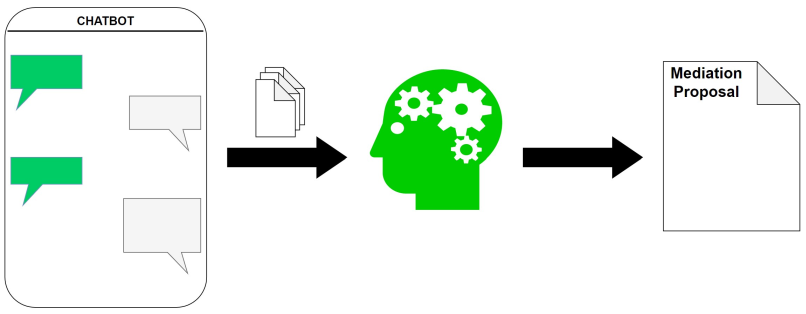


Figure 4.3. Aspera's Phase 1.

4.1.3 Phase 2

The second phase aims to generate, through the machine learning algorithm, a second, more elaborate mediation proposal. The parties must interact again with the chatbot. The difference from the previous phase is that the questions submitted to the disputants are more specific and personalised. The chatbot has a predetermined list of possible questions and must present each party with a subset of these.

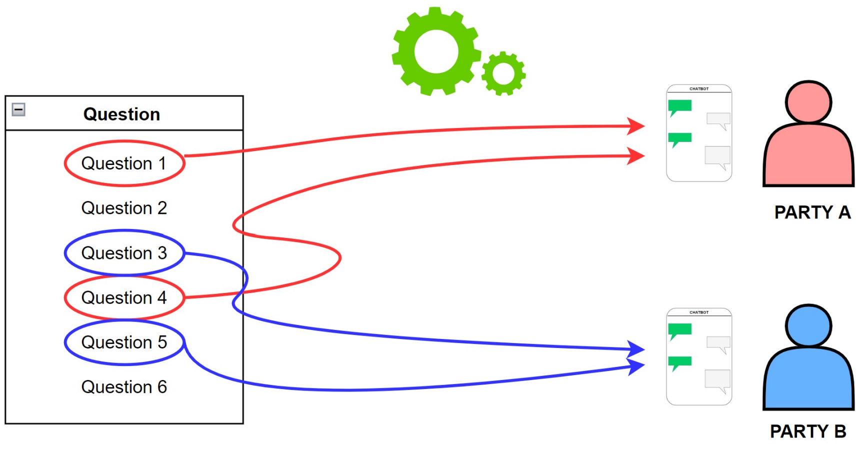


Figure 4.4. Aspera's Phase 2.

We now establish the criteria with which the questions should be chosen. A first hypothesis is to depend each party's new questions on the answers given by the contender in the previous phase. The choice of a cross pattern allows to understand the position of one litigant in relation to the other one. Indeed, Phase one required the point of view of each party and the algorithm develops the mediation proposal on the basis of two perspectives

that we may define as independent. Consequently, the first proposal is built with inputs that are intrinsically incomplete. By contrast, the second phase allows the chatbot to present the counterpart's point of view to the party, and according to the responses, the algorithm is able to elaborate a better proposal by having a more complete view on the dispute.
A second hypothesis that facilitates the mediation process is to induce parties to provide a text in which they explain the reasons why the first mediation agreement was rejected, in order to provide more information to the algorithm and show in what way the latter previously failed.
Once the chatbot's interview is over, the answers are sent to the database and the algorithm generates the mediation proposal. Similarly to step one, if both disputants agree, then the dispute is resolved and the Mediable contract executes the ruling. Otherwise, the parties can decide whether to advance to phase three by sending the requested fees.

4.1.4 Phase 3

The third phase is the last attempt to find a mediation agreement between the parties and involves the participation of a human mediator. Aspera offers the disputants the opportunity to discuss the resolution of the dispute in a virtual courtroom, where the mediator conducts the meeting. At the end of the hearing, the mediator generates the agreement and sends it to both parties, which again may decide whether to accept or reject the proposal.

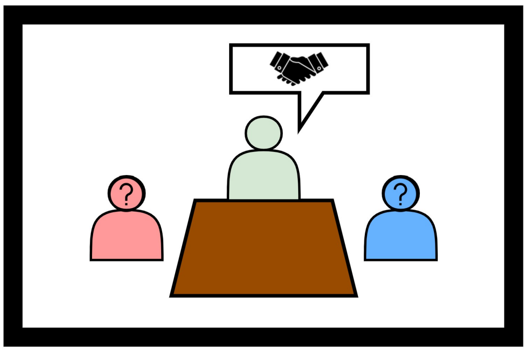


Figure 4.5. The virtual meeting between the parties and the mediator during Phase 3.

We identify two possible ways in order to select the human mediator. The first one involves the parties choosing the mediator by mutual agreement. As mentioned earlier, the list of mediators is public and they are categorised according to their experience on the platform or the areas in which they have the most skills and expertise. Thus, the parties can decide on the mediator who best suits their dispute.
The second proposal introduces a random drawing of the mediator, who is anonymous

to the parties. We may consider the introduction of a PoS-type protocol for managing the selection mechanism. This approach requires the presence of a native token, which we define as *Aspera token*. An Aspera mediator stakes a sum of Aspera tokens in the Aspera's contract, in order to be drawn. Upon drawing, a part of the deposit is locked until the end of Phase 3. We note that penalties related to the outcome of the dispute are not expected, but we can prevent the case where a mediator does not perform their duty. We can introduce a penalty for lazy conduct in this instance. Therefore, if the mediator does not organise a meeting with the parties or fails to send a mediation agreement in time, the locked capital is not released and is placed back on the market.
A further discussion point is related to the meeting procedure in the virtual courtroom. The blockchain is a pseudo-anonymous environment, so the aim is to keep the identities of the parties and the mediator anonymous during the dispute. At the same time, an in-person meeting must be reproduced as closely as possible. Aspera's protocol involves a virtual camera using artificial intelligence-based systems that camouflage faces and voices by means of a deep-fake technique in such a way as to ensure users involved cannot be recognised. Another important aspect is to keep the facial micro-expressions of the parties recognisable during the encounter so that the mediator will be able to formulate a better mediation proposal.

4.1.5 Phase 4

The fourth and final phase involves the resolution of the dispute through a human arbitrator. The latter is drawn randomly from the list of arbitrators on the Aspera's platform and the identity is kept secret from the parties until the end of the dispute to prevent bribes or intimidation. All the considerations made in the previous paragraph apply to the method of extraction.
During the fourth phase, the arbitrator receives all the documentation produced in the previous phases and formulates the ruling to be sent to the Mediator contract, which will inform the Mediable contract of the ruling to be executed.

4.2 The Machine Learning question

The use of a machine learning-based algorithm is one of the focal points in Aspera's protocol. An ML system must be able to learn from examples, reasoning and experience and must improve decision-making capabilities without the need for human intervention. According to Tom Michael Mitchell's definition, *'Machine learning is the study of computer algorithms that allow computer programs to automatically improve through experience'*. In the case of Aspera, the system must be able to construct a mediation proposal from data on previously settled disputes and from evidence provided by the parties.
On the one hand, the use of these algorithms makes it possible to reduce costs significantly due to the very high level of automation. In addition, this technology goes well with

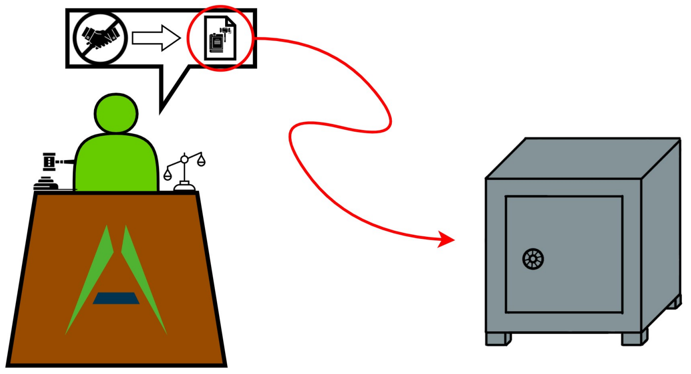


Figure 4.6. Aspera's arbitrator deliberates the ruling and submits it to the Mediable contract.

the blockchain environment. An ML algorithm cannot predict outcomes that it has not previously seen. In the case of blockchain dispute resolution, there is no risk of committing this mistake, since a conflict is handled by a smart contract that is deterministic and thus allows knowing all admissible rulings a priori.

On the other hand, there are several problems that complicate the realisation of this project. The first obstacle is the need to have a huge amount of initial data collected in a dataset that the algorithm uses to generate a proposal. Indeed, these algorithms require an initial training phase where they refine the technique of generating an outcome from existing data.

The first phase must be dedicated to filling the database and we propose two different ways. The first option involves manually entering synthetic data. This approach raises a further problem. The process is highly centralised as developers must choose which data to input. Consequently, the algorithm's solving capability is influenced by external factors that may result non-objective. The ideal data source is identified as from the outcomes of other blockchain-related disputes. However, we have seen that these platforms are rarely used and moreover the use cases are still disconnected from the offchain reality. Consequently, this kind of disputes is not functional for our application.

The second proposal is to resolve the first disputes directly in Phase 3, allowing the algorithm to learn from human mediators.

A further problem with Machine Learning is related to Phase 2 of the protocol. The database where the parties' answers are stored is represented as a matrix with twice as many columns as the predefined list of questions and each row is associated with a dispute. The entries in each row coincide with the inputs to be fed into the ML algorithm in order to formulate the mediation proposal for that specific case. We observe that during this phase, parties only reply to a subset of questions. As a result, there are *missing inputs* and the ML algorithm has difficulty training and learning since each dispute will have a

different combination of input values.
The last issue related to the use of machine learning is that the algorithm needs to be able to understand the answers sent by the parties. This is particularly difficult since the disputants may communicate with the chatbot using different languages and the meaning of some words may be misused depending on the context of the dispute.

4.3 Decentralised Mediation

Let us now introduce a second dispute resolution protocol based on mediation that we refer to as *Decentralised Mediation*.
In all the mediation mechanisms we aforementioned, disputes are settled by a single mediator, whether human or not. On the contrary, this protocol achieves a mediation agreement through the combined work of several agents competing with each other. As a result, the role of the mediator is replaced by two new figures we refer to as *proposers* and *choosers*. As the name suggests, proposers are users who formulate one or more mediation proposals that are placed into a set of potential final outcomes for the dispute. The participation of a proposer in any dispute is voluntary. The formulation of a mediation proposal requires a little fee that is part of the reward for the winning proposer.
The idea behind the protocol is that, allowing access to the conflict to a wider audience of agents spread around the world with different viewpoints and cultures, it offers a potentially larger range of solutions and may represent a benefit for the whole process.
The other figure involved in this mechanism is the chooser, which must examine all the options submitted by the proposers and pick the fairest one, which will subsequently be presented to the parties as a mediation agreement. The chooser is unique for each dispute and is drawn randomly from a list of candidates. The selection mechanism is the same of the arbitration protocols described in the previous chapter. As a result, it is necessary to design *native tokens* for decentralised mediation in order to avoid the whole range of PoS issues relating to the use of ETHs that we have already listed in the Kleros section. The chooser's remuneration or loss depends on the quality of the service offered.
Finally, the parties have a deadline to accept or reject the mediation proposal.

4.3.1 Rewards and penalties mechanism in decentralised mediation

Let us discuss the mechanism of rewards and penalties, which is designed to reach a high-quality mediation process.
In our opinion, a dispute is considered solved if both parties accept the mediation agreement. However, reaching a compromise between the disputants is not always possible. The reasons that lead to the failure of mediation may be of varying natures. To begin with, a first cause could be the lack of competence exhibited by a mediator who is unable

to propose quality solutions. Another factor to be taken into account is the stubbornness of one of the parties who does not accept a loss that is perceived as excessive in relation to the opponent.
Based on the potential fallacy of the mediation process, we have developed a system of rewards and penalties depending to the dispute's outcome, in order to preserve all parties involved, both disputants and solvers. The classification of outcomes and the reallocation of fees and funds will be specifically addressed in the section 4.3.6 related to the last phase of the protocol.
As previously mentioned, the formulation of each mediation proposal requires a fee. This cost is designed to be dependent on the value of the specific dispute. Next, we assume that the payment is made in ETH as the protocol should run on the Ethereum blockchain. The introduction of this mechanism has a threefold purpose.
The first one is to accumulate a treasury for the reward of the winning proposer which is identified as the user who submitted the option selected by the chooser. The second goal is to defend the system from Denial of Service attacks. The last aim is to limit the presence of incompetent proposers trying, with many solutions, to increase their winning chances. Indeed, we observe that the greater is the number of proposals sent, the greater is the economic risk incurred by the proposer. In fact, if no option is selected, then a part of the stake (or whole) is slashed.
The winning proposer receives the maximum reward if both parties accept the mediation agreement. In the other cases the mediation process is unsuccessful, but a minimum payout is still guaranteed and most of the ETH deposited by other proposers is refunded.
Mediation fees paid by disputants are instead allocated to the chooser which proves to have selected the fairest option. If the mediation agreement is accepted by both parties, then the stake is released and the chooser receives a reward equal to the mediation fees payed by disputants. In the event that the parties mutually reject the agreement, then they are refunded. This implies that the chooser has not fulfilled the task properly, so the previously locked stake is slashed. The last case to consider is when the mediation agreement is partially accepted. The chooser's work cannot be considered a complete failure because one of the parties has considered the proposal as fair. Therefore, the stake is returned.

4.3.2 Cycle of a dispute

The dispute, also in the case of decentralised mediation protocol, is settled through the interaction between a *Mediable* and a *Mediator* contract, which we introduced previously.

In our idea, parties and users who want to participate in the resolution process must have a user-friendly portal where they can set up and view the whole process.
Parties who signed a commercial agreement through a Mediable contract may trigger the mediation clause at any time. We remark that these dispute resolution methods on blockchain are designed for escrow contracts that must already include the amount of

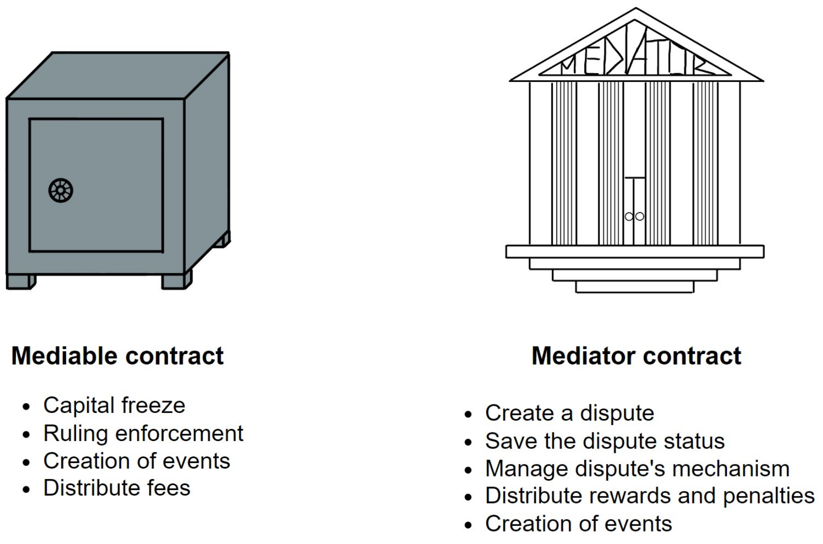


Figure 4.7. Mediable and Mediator contracts in Decentralised Mediation.

money involved. Hence, we may visualize the Mediable contract as a vault, where there are funds that are redistributed afterwards according to the instructions provided by the Mediator contract.
Activation of the mediation clause takes place through a transaction sent to Mediable contract by one party containing half of the fees required to resolve the dispute. When the payment is confirmed, the dispute is notified through the online portal to the opponent, who must send the remaining portion of fees within a set period. Meanwhile, the Mediable contract locks the funds deposited internally and communicates with the Mediator contract, which triggers the function creating the dispute formally through a unique identifier and recording the event on the blockchain.
After the dispute has been raised, the decentralised mediation protocol involves four phases.
The following explanation comes with a practical example in order to be more readable. Alice approaches the freelancer Bob for the development of her website. They decide to use a Mediable smart contract to process the payment and agree on a sum in ETH equal to D which will be paid to Bob on conclusion of the job. Therefore, Alice deposits D inside the contract and Bob begins the commission. At the delivery of the website Alice is not satisfied with the final result, so she demands a discount from the agreed price or that Bob fixes errors within a week for free. Bob thinks he has done the job correctly and requires full payment. As a result, Alice opts to trigger the mediation clause and the dispute arises.

4.3.3 Initial Phase

The initial phase focuses on the dispute set-up. One of the parties, in our case Alice, has sent her half of the mediation fees to the Mediable contract.

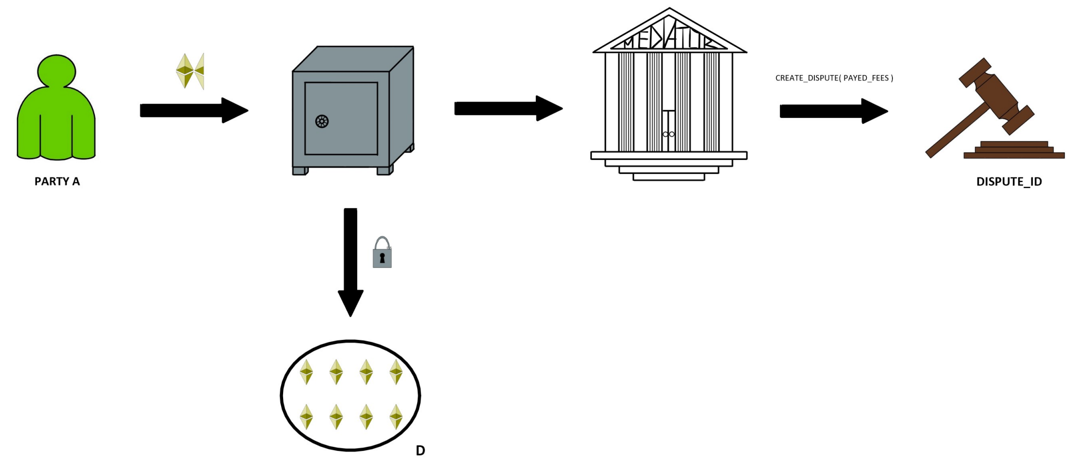


Figure 4.8. Alice pays her fees and dispute begins.

At this stage, the protocol requires that the counterpart, whom we identify as Bob, also pays his fees. In order to prevent the contender from not adhering to his obligation, we opted for a radical remedy. If fees are not paid after a fixed period from dispute creation, then all funds in Mediable contract are transferred to the user who requested the mediation. This means that Alice receives a full refund if Bob does not adhere to the protocol. The paid fees, however, are sent to Mediator contract as a service provider.

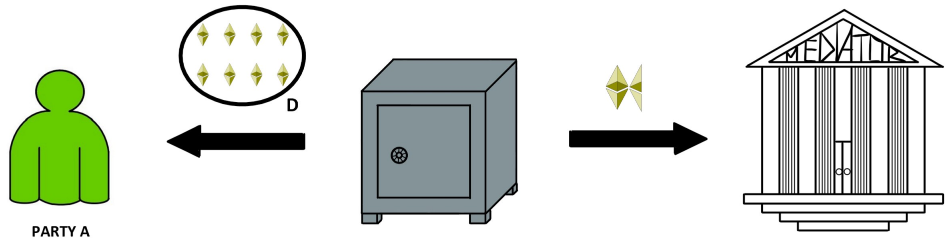


Figure 4.9. Bob did not pay his part of fees.

Let us briefly discuss the amount that fees should correspond to. ODR systems have cost reduction as one of their main goals and are designed for small claims. Therefore, we argue that fees should be proportional to the value of the dispute. Let us introduce the coefficient α which takes a value between 0.1 and 0.2 to be multiplied to the deposit placed in the Mediable. Referring to the example, the mediation fees in the dispute between Alice and Bob are equal to

$$mediation\_fee = \alpha D,$$

$$mediation\_fee\_per\_party = 0.5 \times \alpha D.$$

After all fees result paid, parties may submit evidence through the dashboard in order to support their position. The evidence must be in digital form and can be documents, photos, videos and so on.... For our example, Bob could send the website source code and the initial instructions received from Alice. However, if both parties feel that they have provided enough material in less time, then they may notify the dApp and proceed directly to the next phase.

4.3.4 Proposal Phase

The *proposal phase* is a restricted time window in which users are invited to participate as proposers in the dispute. Decentralised mediation aims to facilitate heterogeneity of solutions and consequently requires conflicts to be carried out openly so that anyone can access the material submitted by the parties and contribute to the resolution.
For example, let us imagine that Carl is a software developer and during a break he accesses to the decentralised mediation's platform. Through the dashboard, he detects the ongoing dispute between Alice and Bob and has a look at the available evidence, especially the website code. After a careful analysis, Carl identifies errors in Bob's code, but believes that Bob has done a good job overall. Through the decentralised mediation protocol, Carl is able to express a judgement on the case by sending what he believes will be the fairest solution.
In order to participate as a proposer, a user must send a transaction containing the mediation proposal and the related stake to Mediator contract.

A proposal must meet the following constraints:

- The proposer can submit a list of transactions to be performed by the Mediable contract. The second available operation is to change the time terms of the contract, i.e. to advance or postpone the date of a payment. Alternatively, a proposal can be built based on a combination of both.
- Proposed transactions must transfer an amount of cryptocurrency equal to the value of the dispute.

In the case of the dispute between Bob and Alice, the Mediable contract has a value equal to D. Carl, in his mediation proposal, may put down a settlement that provides for a 30% refund to Alice of the sum D she was supposed to pay, with the rest sent to

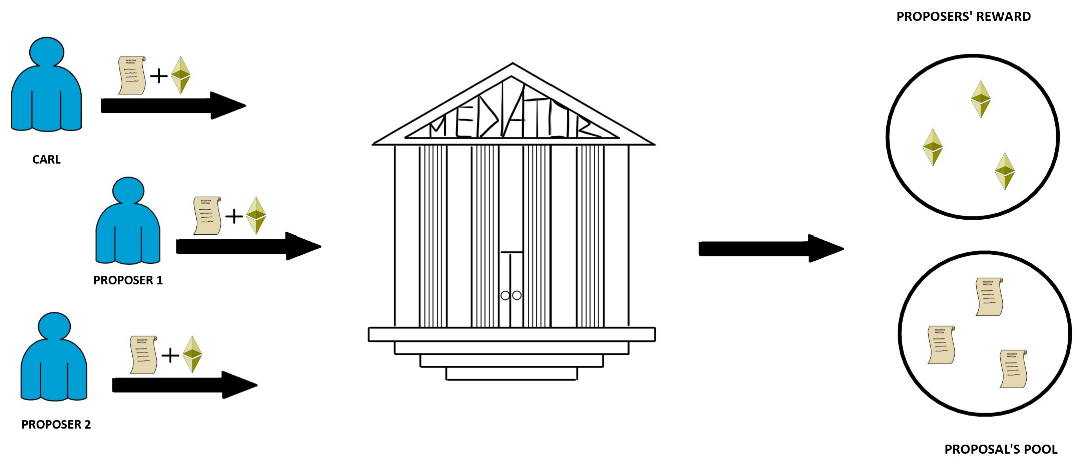


Figure 4.10. Proposal's mechanism.

Bob. Otherwise Carl may suggest that an additional week is given to Bob for resolving mistakes. The last case involves a combination of the two previous options. Thus Bob receives an extra week to fix the site and Alice is reimbursed for 30%. Other types of solutions are currently not available through smart contract. Furthermore, Carl may not propose to redistribute between the parties a sum greater than D.
Proposers may attach a concise text document in which they explain the logic they have applied in order to simplify the chooser's work in the next phase.
Furthermore, we believe that proposals should not be released in public before the end of the dispute. We want to avoid malicious proposers identifying good solutions and copy them. The second reason is to prevent parties from being influenced in the final decision. Indeed, faced with the presence of one or several more advantageous options, one of the parties could reject the mediation agreement offered without further consideration.
The cost of a proposal must also be low in order to encourage greater participation. Let us introduce a second coefficient α_1 to be multiplied by the value of the dispute. In addition, we require the cost of submitting a proposal to be less than mediation fees. Thus, α_1 must satisfy the relation:

$$\alpha_1 < \alpha.$$

We assume that a proposer has a greater incentive to participate in dispute resolution if the risk is low but at the same time foresees a large payout.
In our example, if Carl intends to send his mediation proposal he needs to pay:

$$stake\_proposal = \alpha_1 D.$$

At the end of the proposal phase, the protocol advances to the next phase or directly to the last according to the number of submitted solutions.
Let us make two final remarks. The first is that for each dispute the cost of a proposal is the same for everyone. If we assume that the platform received N proposals for a dispute, then the maximum payout is equal to:

$$max\_payout = N \times \alpha_1 D$$

The second remark is about the possibility of submitting multiple proposals, even if it is not convenient for a proposer. Whereas formulating several solutions increases the chances of winning, this behaviour both increases the financial risk one exposes themselves to and decreases the eventual net payout. Let us assume that n users participated in this phase, and each user i has submitted m_i proposals such that:

$$\sum_{i=1}^{n} m_i = N.$$

The maximum net win for each player is equal to

$$max\_net\_payout_i = (N - m_i) \times \alpha_1 D = max\_payout - m_i \times \alpha_1 D,$$

with $m_i \times \alpha_1 D$ representing the total stake of the *i-th* proposer.
We observe that $max\_net\_payout_i \searrow 0$ if $m_i \nearrow N$.

4.3.5 Selection Phase

The protocol moves into the *selection phase* if at least two proposals have been submitted. The system randomly draws a chooser, among those users with a stake in Mediator contract. The chooser is promptly notified of the assignment and a portion of token in stake is frozen by the Mediator contract until the end of mediation. The amount of tokens taken from the chooser is equal to

$$locked\_stake\_chooser = \alpha_2 S. \tag{4.1}$$

where α_2 is a proportionality coefficient and S represents the chooser's stake.
The selection mechanism is the same as in Kleros.

In addition, lazy choosers who do not fulfil their duty are penalised with the loss of locked tokens. The chooser is the only one with access to the proposals and selects the one he considers the fairest.

4.3.6 Mediation phase

During the *mediation phase*, parties examine the mediation agreement and decide whether to accept or reject the offer. The protocol allows three different scenarios depending on the number of proposals generated during the second phase:

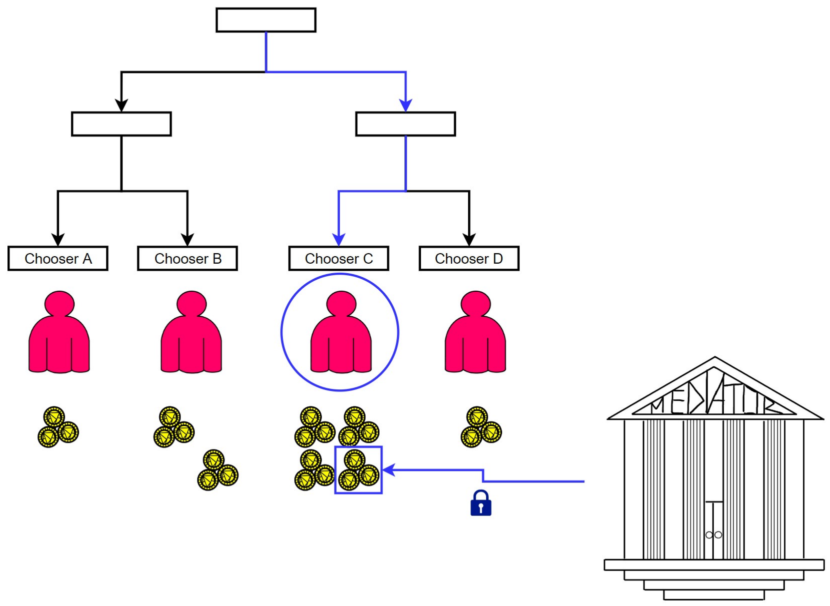


Figure 4.11. Selection Phase in Decentralised Mediation.

1. **Case with zero proposals.**

   We do not preclude the eventuality that no user offers a contribution to the dispute resolution. The protocol failed to produce any mediation agreement, so parties are fully reimbursed and will have the possibility to trigger the mediation clause in the future for a new attempt.

2. **Case with one proposal.**

   If only one proposal has been submitted at the end of the second phase, the involvement of the chooser is meaningless. The protocol presents the unique mediation agreement to the parties.
   If parties accept the solution, then the proposer receives a reward equal to the mediation fees and the proposal's ETH are refunded.
   If litigants reject the mediation agreement, then the proposer is penalised by losing fees paid to submit the proposal. Again, the parties are fully reimbursed.
   In the third case, one of the parties accepts the agreement, while the other refuses. Half of the mediation fees are returned to disputants. The remaining part is divided equally between the proposer and the Mediator contract.

3. **Case with at least two proposals.**

The presence of several proposals leads to the complete protocol. Again, we detect three possible outcomes that we treat separately due to the greater difficulty in redistribution of rewards and penalties. Let us assume that n proposers participated in the dispute and that each submitted m_i mediation proposals respectively. We refer to the winning proposer using the subscript win.

- The first scenario involves the parties accepting the mediation agreement. In this case the chooser receives all mediation fees, which we recall are equal to $mediation\_fee = \alpha D$, and the locked stake is released. The winning proposer gets a total reward of $max\_payout = N \times \alpha_1 D$. The net profit is

$$max\_net\_payout_{win} = (N - m_{win}) \times \alpha_1 D.$$

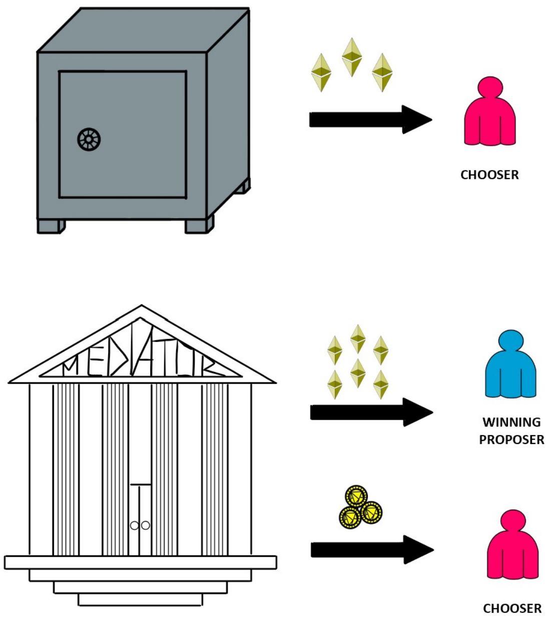


Figure 4.12. Redistribution mechanism when mediation is successful.

- In the second case, both parties reject the agreement. The responsibility for the failure of mediation is mainly attributed to the chooser who loses the locked tokens $locked\_stake\_chooser = \alpha_2 S$. On the other hand, the winning proposer receives a reward that is far less than max_net_payout. The residual amount accumulated in the proposal phase is returned to other proposers. Furthermore, the protocol fully reimburses the parties.

We define the coefficient α_3 which represents the percentage of *stake_proposal* that is returned to the non-winning proposers for each unsuccessful proposal. In contrast to the previous coefficients, we want to choose a value of α_3 close to 1. Since the parties cannot see the options in the clear, it is not certain that the best proposal was not among them. Therefore, we think it is unfair for other proposers to receive an excessively severe sanction if in the end the dispute is not even solved. The payback for a mediation proposal is equal to

$$refund\_proposal = \alpha_3 \alpha_1 D.$$

To the i-th losing proposer the protocol returns a total ETH sum

$$refund\_proposer_i = m_i \times \alpha_3 \alpha_1 D.$$

Consequently, the quantity retained by the Mediator contract is

$$penalty\_proposer_i = m_i \times (1 - \alpha_3)\alpha_1 D.$$

The protocol in this case provides that the winning proposer is sent a payout equal to

$$payout_{win} = m_{win} \times \alpha_1 D + (1 - \alpha_3) \times \alpha_1 D \sum_{i \neq win} m_i, \tag{4.2}$$

where the first term is the return of the proposal fees and the second one is the sum of the other proposers' penalties and constitutes the net reward.
We observe that the latter term goes to zero if the value of $\alpha_3 \nearrow 1$.

- In the third case, the mediation proposal is partially accepted, i.e. only one of the parties rejects the agreement received. As previously noted, we do not know in advance what may be the cause of the failure of dispute resolution. Thus, attributing more guilt and the associated penalty to a single responsible party is difficult.
  We decided to maintain the same redistribution model for proposers as in the previous instance by guaranteeing a minimum payout and a large part of the payback for participants in the proposal phase. We are not increasing the value of the reward in order to prevent that a proposer voluntarily formulates advantageous options for merely one party.
  The protocol releases the chooser from the stake locked at the time of selection. Mediation fees within the Mediable contract are sent to Mediator one. The decision to not fully reimburse the parties is designed to prevent one of the disputants from accepting a mediation agreement that is clearly in their favour.

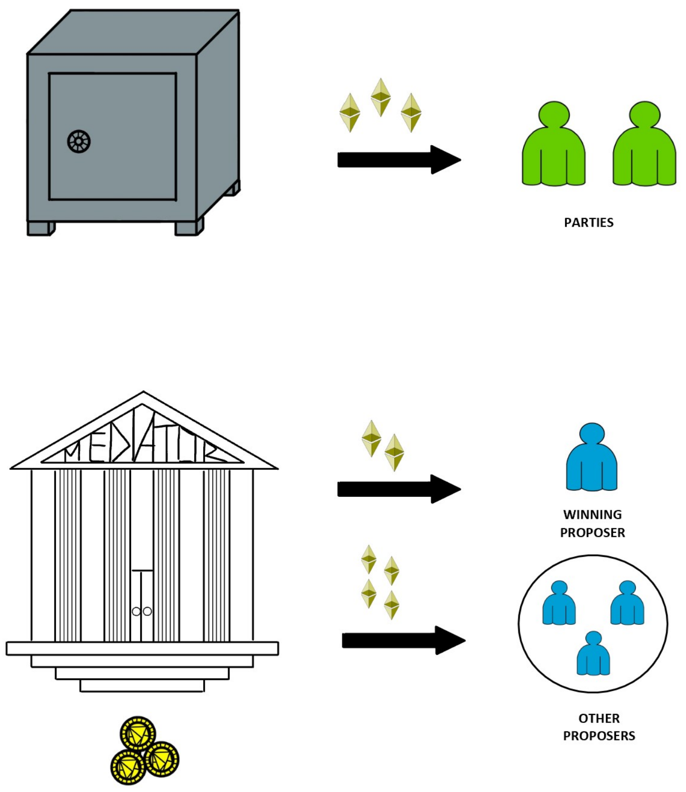


Figure 4.13. Redistribution mechanism when mediation fails and both parties refuse the agreement.

Let us summarise by composing the payout matrix of each user involved in the dispute. Let us assume that the *i-th* proposer sent m_i proposals, then the corresponding payoff matrix represented in Table 4.1.

$Proposer_i$	(YES,YES)	(YES,NO)	(NO,NO)
WIN	$(N - m_i) \times \alpha_1 D$	$(1 - \alpha_3) \times \alpha_1 D \sum_{j \neq i} m_j$	$(1 - \alpha_3) \times \alpha_1 D \sum_{j \neq i} m_j$
LOSE	$-m_i \times \alpha_1 D$	$-m_i \times (1 - \alpha_3) \times \alpha_1 D$	$-m_i \times (1 - \alpha_3) \times \alpha_1 D$

Table 4.1. Payoff matrix the *i-th* proposer.

The possible outcomes of the mediation process are listed in the columns where

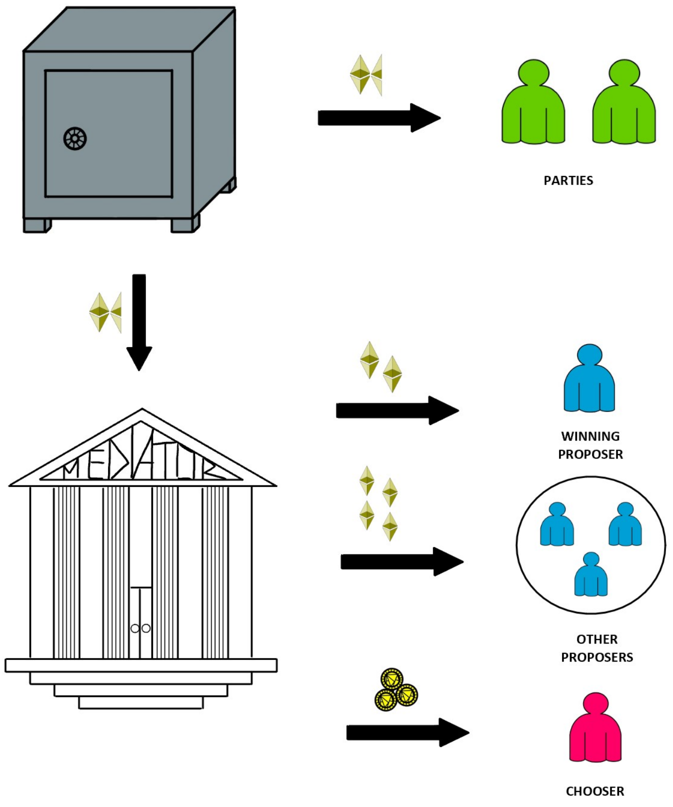


Figure 4.14. Redistribution mechanism when mediation fails and one party refuse the agreement.

'YES' indicates that the agreement was accepted, and 'NO' in case of rejection. In the rows, on the other hand, it is reported whether or not a proposer option has been selected by the chooser.
We then build the payoff matrix for the chooser in Table 4.2. Rows and columns show the actions of the parties during the last phase.

After the disputants have communicated their decision, the Mediator contract distributes the rewards and penalties among the chooser and proposer. The Mediator contract then submits the outcome to the Mediable contract.

4.3.7 Limitations about decentralised mediation

In this closing section, we propose the potential limits for this protocol.
This system is not entirely free from corruption. A proposer may bribe a chooser to send

Chooser	YES	NO
YES	αD	0
NO	0	$-\alpha_2 S$

Table 4.2. Payoff matrix for a chooser.

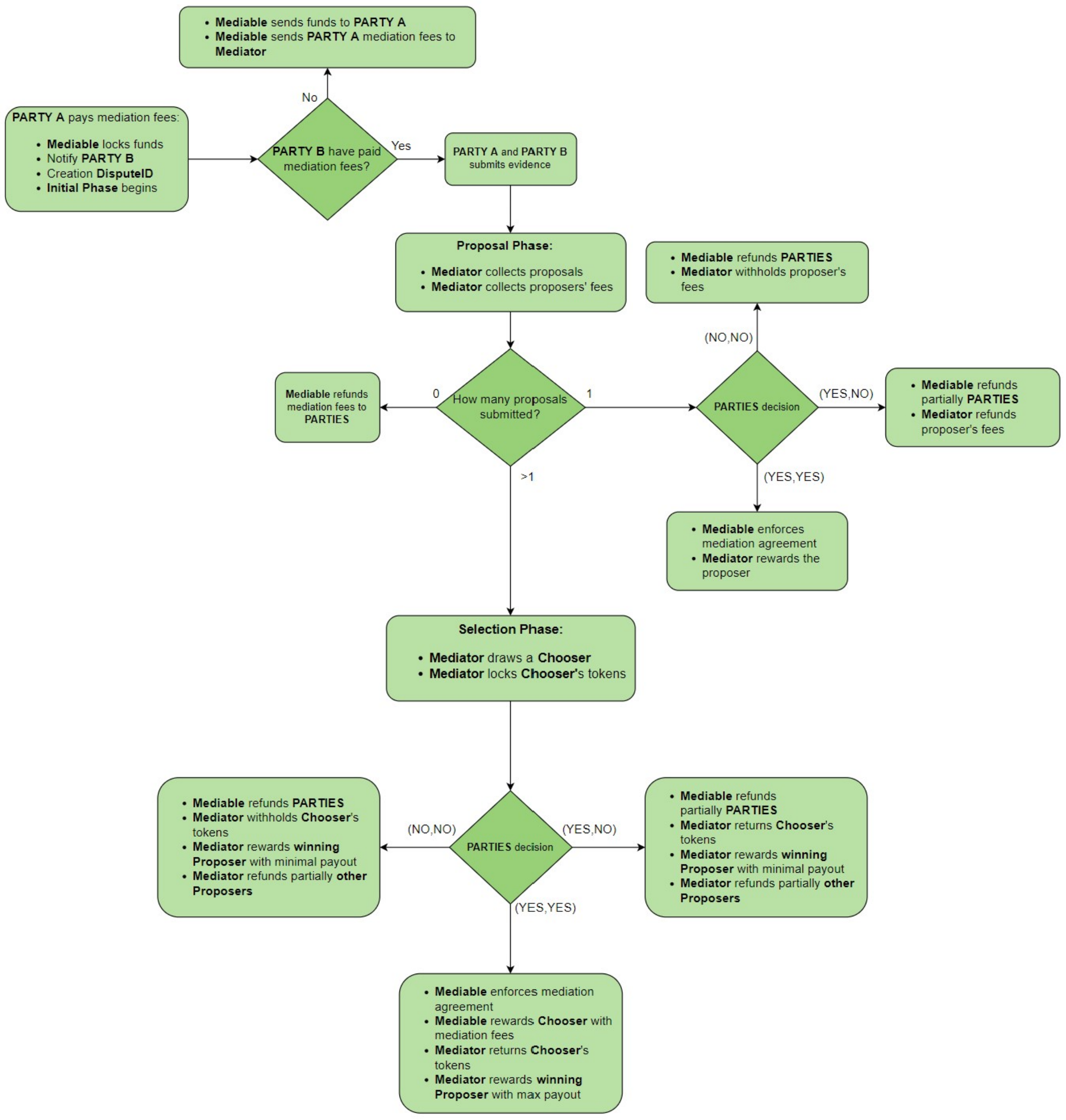
one of his proposals to the parties. If the malicious proposer offers a sum higher than the chooser's stake then selecting his option is always a better strategy. One solution to this problem could be to keep the addresses of participants in a dispute hidden, to avoid any kind of communication.
Alternatively we should set the value of locked tokens to the chooser in such a way that the deposit is higher than the maximum minimum reward for proposers, which is known at the end of the proposal phase. This value coincides with the eventual minimum reward of the user who submitted the fewest proposals. If we denote the latter with the index i^* then the potential reward is equal to

$$max\_min\_payout = (1 - \alpha_3) \times \alpha_1 D \sum_{i \neq i^*} m_i. \tag{4.3}$$

Accordingly, the protocol must lock to the chooser an amount equal to

$$stake\_proposer_{ETH} > max\_min\_payout, \tag{4.4}$$

where $stake\_proposer_{ETH}$ is a quantity in ETH to be converted into the platform's native tokens. This new assumption implies that the stake of a chooser must depend on the value of the dispute and the definition (4.1) is no longer valid.
Let us observe that the protocol does not exclude the scenario in which a proposer may be drawn as a chooser for the same dispute. The constraint (4.4) prevents the latter from selecting one of his options anyway without examining alternatives. This is because the loss due to the stake is greater than the minimum win. In this case, the chooser is incentivised to choose a better option than his own.
The solution proposed in (4.4) is impracticable due to two drawbacks. The first problem consists in introducing a minimum threshold of tokens that the chooser must commit in order to participate in the dispute. Thus, the resolution process becomes less accessible, in contrast to our philosophy.
The second drawback is that the chooser may have to commit an amount of tokens that potentially exceeds the reward given by mediation fees in the case of a large number of proposals submitted. Thus, the position of the chooser would be even more disadvantageous than in the original protocol.



• Mediable sends funds to PARTY A
• Mediable sends PARTY A mediation fees to Mediator
No
PARTY A pays mediation fees:
• Mediable locks funds
• Notify PARTY B
• Creation DisputeID
• Initial Phase begins
PARTY B have paid mediation fees?
Yes
PARTY A and PARTY B submits evidence
Proposal Phase:
• Mediator collects proposals
• Mediator collects proposers' fees
• Mediable refunds PARTIES
• Mediator withholds proposer's fees
(NO,NO)
Mediable refunds mediation fees to PARTIES
0
How many proposals submitted?
1
PARTIES decision
(YES,NO)
• Mediable refunds partially PARTIES
• Mediator refunds proposer's fees
>1
(YES,YES)
• Mediable enforces mediation agreement
• Mediator rewards the proposer
Selection Phase:
• Mediator draws a Chooser
• Mediator locks Chooser's tokens
• Mediable refunds PARTIES
• Mediator withholds Chooser's tokens
• Mediator rewards winning Proposer with minimal payout
• Mediator refunds partially other Proposers
(NO,NO)
PARTIES decision
(YES,NO)
• Mediable refunds partially PARTIES
• Mediator returns Chooser's tokens
• Mediator rewards winning Proposer with minimal payout
• Mediator refunds partially other Proposers
(YES,YES)
• Mediable enforces mediation agreement
• Mediable rewards Chooser with mediation fees
• Mediator returns Chooser's tokens
• Mediator rewards winning Proposer with max payout

Figure 4.15. Decentralised mediation's workflow.

The second important limitation of decentralised mediation is the risk associated to the lack of participation in the resolution process. The mechanism is designed with the optimistic view of a platform visited by many users. As mentioned earlier, we set very low fees to make the resolution process accessible to as wide an audience as possible.
Proposers have an incentive to formulate more reasonable mediation agreements when there is more competition because a high number of options submitted by different users allows them to accumulate a higher reward. In this way, the protocol provides several

quality solutions for the parties. Moreover, a chooser has the opportunity to offer a better service if there are different options to select from.
The final limit we have identified lies in the high potential of failure in the mediation process proposed. Presenting a single mediation agreement may be limiting for the parties. A solution to this problem is the introduction of several rounds to the protocol. Let us recall that the parties are entitled to a full or partial refund depending on the number of refusals during the last phase. In the case of a full refund, the disputants may decide not to withdraw their ETH from the contract and start a new attempt. When, on the other hand, one of the two parties accepts the proposed agreement, only half of the fees are refunded. Thus, to start a new round, a supplement for the remaining part of the fees is required.
A further consideration is related to the management of proposals. In this case, the best approach would be to keep the previous proposals as well as giving the opportunity to create new ones. The problem arises that the protocol has elected the winning proposer and automatically redistributes the treasury. The solution we suggest is that the parties decide whether to start a new round before the rewards and penalties are reallocated to the proposers.

Chapter 5

Conclusions

Blockchain represents an opportunity to improve the legal sector. The public and distributed nature of this technology allows parties to have more control over the entire resolution process. Moreover, the ideal of a global network that does not require the presence of a central authority and the executive power of smart contracts has inspired a kind of restoration of direct democracy in dispute resolution.

As a result, several platforms have emerged in recent years with the aim to propose decentralised protocols for solving disputes through blockchain. Existing approaches tend to apply arbitration building a voting mechanisms based on a Schelling's game scheme. Via Aspera and Decentralised Mediation, we show how mediation is also a suitable alternative in dispute resolution within blockchain technology. The non-adjudicative nature of this approach allows parties from leaving their affairs in the hands of a Trusted Third Party.

The blockchain dispute resolution platforms' landscape is still in an early stage. This perception is confirmed by Kleros, which is the leading platform, but more than 90 per cent of the disputes settled are unrelated to the real world. In other projects, the dispute resolver is only a supplementary tool as in Aragon. Many platforms, on the other hand, met failure before they even became active, as in the case of Jur.

In our opinion, the main reason why blockchain dispute resolution platforms have not spread is the lack of mass adoption of the technology behind them. As a result, these projects are not adequately sponsored and tend to be abandoned in the long-term.

With the increasing presence of blockchain in ordinary life over the next few years, new dApps will certainly emerge that will attempt to resolve the issue of disputes, perhaps by making greater use of cryptographic primitives such as zero-knowledge proofs in such a way to improve the privacy of the users involved.

Bibliography

[1] Danilo Bazzanella, Andrea Gangemi, *Lecture notes on Blockchain and cryptoeconomy*, Politecnico di Torino, Slides (2021)

[2] Satoshi Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System* (2008)

[3] Vitalik Buterin, *Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform* (2014)

[4] Ethereum.org, https://ethereum.org/it/developers/docs/

[5] Andrea Corbellini, *Elliptic Curve Cryptography: a gentle introduction*, https://andrea.corbellini.name/2015/05/17/elliptic-curve-cryptography-a-gentle-introduction/ (2015)

[6] Andreas M. Antonopoulos, Dr. Gavin Wood, *Mastering Ethereum*, O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472 (2019)

[7] Dr. Gavin Wood, *Ethereum: a secure decentralised generalised transaction ledger* (2022)

[8] Stuart D. Levi, Alex B. Lipton, Skadden, Arps, Slate, Meagher & Flom LLP, *An Introduction to Smart Contracts and Their Potential and Inherent Limitations*, Harvard Law School Forum on Corporate Governance (2018)

[9] Shafaq Naheed Khan, Faiza Loukil, Chirine Ghedira-Guegan, Elhadj Benkhelifa, Anoud Bani-Hani, *Blockchain smart contracts: Applications, challenges, and future trends* (2021)

[10] Giacomo Como, Fabio Fagnani, *Lecture notes on Network Dynamics*, Politecnico di Torino (2020)

[11] John F. Nash, *Non-cooperative Games*, Annals of Mathematics, Second Series, Vol. 54, No. 2, pp. 286-295, Mathematics Department, Princeton University (1951)

[12] Thomas Schelling, *The Strategy of Conflict*, Harvard University Press, Cambridge (1960)

[13] Roger B. Myerson, *Game Theory: Analysis of Conflict*, Harvard University Press, Cambridge (1991)

[14] Andrew M. Colman, *Game Theory*, Volume 2, pp. 688–694 in Encyclopedia of Statistics in Behavioral Science, Editors Brian S. Everitt David C. Howell, John Wiley Sons, Ltd, Chichester (2005)

[15] Vitalik Buterin, *SchellingCoin: A Minimal-Trust Universal Data Feed*, Available online: https://blog.ethereum.org/2014/03/28/schellingcoin-a-minimal-trust-universal-data-feed (2014)

[16] Vitalik Buterin, *The P+epsilon Attack*, Avaiable online: https://blog.ethereum.org/2015/01/28/p-epsilon-attack (2015)

[17] William George, Clement Lesaege´, *An Analysis of p+ϵ Attacks on Various Models of Schelling Game Based Systems* , Available online: https://cryptoeconomicsystems.pubpub.org/pub/george-schelling-attacks/release/7 (2021)

[18] James Metzger, *The current landscape of blockchain-based, crowdsourced arbitration*, University of New South Wales (2019)

[19] Gudkov A.V., *Crowd Arbitration: Blockchain Dispute Resolution*, Legal Issues in the Digital Age, no 3, pp. 59–77 (2020)

[20] Matthew Dylag Harrison Smith, *From cryptocurrencies to cryptocourts: blockchain and the financialization of dispute resolution platforms* (2021)

[21] Amy J. Schmitz, Colin Rule, *Online Dispute Resolution for Smart Contracts*, 2019 J. Disp. Resol. (2019)

[22] Orna Rabinovich-Einy, Ethan Katsch, *Blockchain and the Inevitability of Disputes: The Role for Online Dispute Resolution*, 2019 J. Disp. Resol. (2019)

[23] Online Dispute Resolution | European commision, https://ec.europa.eu/consumers/odr/main/?event=main.statistics.show

[24] Civil Resolution British Columbia, https://civilresolutionbc.ca/

[25] Kleros official site, https://kleros.io/

[26] Clément Lesaege, William George, Federico Ast , *Kleros, Long Paper v2.0.2*, Available online: https://kleros.io/yellowpaper.pdf (2021)

[27] Kleros book, https://kleros.gitbook.io/docs/

[28] Enrique Piqueras, *KlerosLiquid.sol*, https://github.com/kleros/kleros/blob/master/contracts/kleros/KlerosLiquid.sol

[29] Clément Lesaege, *ERC-792: Arbitration Standard*, https://github.com/ethereum/EIPs/issues/792

[30] Enrique Piqueras, *An Efficient Data Structure for Blockchain Sortition*, Available online: https://medium.com/kleros/an-efficient-data-structure-for-blockchain-sortition-15d202af3247 (2018)

[31] Kleros Arbitration contracts, Available online: https://github.com/kleros/kleros-interaction/tree/master/contracts/standard/arbitration (2021)

[32] William George, *The Kleros TCR Appeal System, or The Third Way of Participating in Kleros*, Available online: https://blog.kleros.io/kleros-decentralized-token-listing-appeal-fees/ (2019)

[33] Klerosboard.com, https://klerosboard.com/

[34] Aragon official site, https://aragon.org/

[35] Luis Cuende, Jorge Izquierdo, *Aragon Network a decentralized infrastructure for value exchange, Whitepaper Version 1.1* (2017)

[36] Aragon User Documentation, *Aragon Court*, https://documentation.aragon.org/products/aragon-court

[37] Aragon, *Aragon Court v2-Technical spec*, https://github.com/aragon/protocol/tree/development/docs (2022)

[38] Facu Spagnuolo, *Aragon whitepaper github*, https://github.com/aragon/whitepaper (2019)

[39] Aragon Court Dashboard, https://court.aragon.org//dashboard

[40] Facu Spagnoulo, *Aragon Court contract: IArbitrable.sol*, https://github.com/aragon/aragon-court/blob/master/contracts/arbitration/IArbitrable.sol (2020)

[41] Jur official site, https://jur.io/

[42] Jur team, *Jur, whitepaper V.2.0.2* (2019)

[43] Aspera official site, https://aspera.rocks/

[44] Fadi Barbàra, Andrea Gangemi, Giulio Stefano Ravot, *Aspera Anonymous dispute resolution - short paper* (2021)

[45] Andrea Gangemi, *Blockchain dispute resolution: the example of Aspera*, Slides (2022)