Ontology of Blockchain Technologies. Principles of Identification and Classification

Paolo Tasca[1]

Thayabaran Thanabalasingham[2]

Claudio J. Tessone[3]

[1]University College of London

[2]J.W. Goethe University Frankfurt

[3]University of Zurich

Summary:

A comparative study across the most widely known blockchain technologies is conducted with a bottom-up approach. Blockchains are disentangled into building blocks. Each building block is then hierarchically classified in main and subcomponents. Then, alternative values (i.e., *layouts*) for the subcomponents are identified and compared between them. Finally, an ontology matrix summarises the study and provides a navigation tool across different blockchain architectural configurations.

Correspondence to: Paolo Tasca (P.Tasca@ucl.ac.uk).

1 INTRODUCTION

1.1 BACKGROUND

The original combination of a set of existing technologies (distributed ledgers, public-key encryption, merkle tree hashing, consensus protocols) gave origin to the peer-validated decentralised cryptocurrency called Bitcoin, origiinally introduced by Satoshi Nakamoto in 2008 [1]. That year was the advent of a new technological milestone: the *blockchain*. Indeed, blockchain has an impact well beyond the specific case of Bitcoin. Blockchain allows new forms of distributed software architecture to be developed where networks of untrusted participants can establish agreements on shared states for decentralised and transactional data without the need of a central point of control. More broadly, it is a revolutionary method to ensure trust among anonymous counterparts in decentralised systems without the need of central, supervisory, authorities in charge of verifying the correctness of the records in the ledger. While blockchain is still in its emergent technological phase, it is fast evolving with the potential to have applications in many sectors of our socio-economic systems. According to some statistics summarised by the World Economic Forum the interest on blockchain expanded globally, [2]. Almost thirty countries are currently investing in blockchain projects. In the finance sector, 80% of the banks predict to initiate blockchain-related projects by 2017. Additionally, venture capital investments with a focus on blockchain activities raised to over 1.4 billion USD from 2014 to 2016. Since blockchain digital currencies combine together the characteristics of money with those of a payment system [3], also central banks started to look into the technology. Currently, over nineteen central banks worldwide do blockchain research and development. Some of them - e.g. the Bank of England - already have commissioned studies on CBDC (central bank digital currency). From the industry side, over one hundred corporations have joined blockchain working groups or consortia and the number of patents filled increased to more three thousand at the moment of writing. These figures show the importance and awareness of blockchain as one of the most promising emerging technologies that, together with Artificial Intelligence, Internet of Things or nanotechnology, will have a pervasive impact on the future of our society.

1.2 MOTIVATION

At the moment of writing, we may argue that – according to the Technology Life Cycle theory –, we are at the beginning of the so called phase of "fermentation" which is characterised by technological uncertainty due to the evolution of the blockchain into alternative technological paths. The industry promotes different model designs favouring functional and performance aspects over others in order to meet specific business goals. Currently there are thousands of blockchain projects worldwide under development, some of them are based on forks of successful technologies such as Bitcoin or Ethereum, while others propose completely new functionalities and architectures. For this reason, instead of blockchain, in the remainder we refer to *blockchains* or *blockchain technologies* in order to encompass all the possible architectural configurations and, for the sake of simplicity, also the larger family of distributed ledger technologies, i.e., community consensus-based distributed ledgers where the storage of data is not based on chains of blocks.

An heterogeneous development combined with a lack of interoperability may endanger a wide and uniform adoption of blockchains in our techno- and socio-economic systems. Moreover, the variation of blockchain designs and their possible configurations represent an hindrance for software architectures and developers. In fact, without the possibility to resort on a technical reference model, it is difficult to measure and compare the quality and the performance of different blockchains and of their applications sit on the top of them. To summarise, current variations of blockchain software architectures pose greatest concerns from different perspectives: user, functionality, implementation and deployment. The solution to these problems requires the setting up of software reference architectures where standardised structures and respective elements and relations shall provide templates for concrete blockchain architectures.

A standard for software reference architecture is thus necessary in order to enable a level playing field where every industry players and community members can design and adopt blockchain-enabled products or services under the same very conditions with possibility of data exchange. As it is for the Internet, several institutes of standardisations (e.g., ETF in cooperation with the W3C, ISO/IEC, ITU) set a body of standards. Internet standards promote interoperability of systems on the Internet by defining precise protocols, message formats, schemas, and languages. As a result, different hardware and software can seamlessly interact and work together. Applied to World Wide Web (as a layer on the top of the Internet), standards bring interoperability, accessibility and usability of web pages. Similarly, the adoption of blockchain standards will promote the blossoming and proliferation of interoperable blockchain-enabled applications. Thus, if we envisage a future where blockchains will be one of the pillars of our society's development, it is necessary to begin discussing and identifying standards for blockchain reference architectures.

The aim of this study is to highlight the need for standard technical reference models of blockchain architectures. This is timely aligned with the industry sentiment which currently pushes organisations for standardisation to set industry standards. In order to support an appropriate co-regulatory framework for blockchain-related industries, it is necessary a multi-party approach as it is for the Internet where both national standards, international standards and a mixture of standards and regulation are in place. In the mid-long term, lack of standards could bring risks related to privacy, security, governance, interoperability and risk to users and market participants. From a preliminary survey conducted in 2016 by Standards Australia, more than 88% of respondents indicate the role for standards in supporting the roll out of blockchain technologies [4].

1.3 METHODOLOGY

In light with the motivations described above, this study aims at proposing a rich *blockchain ontology*: a reference architectural model for blockchains and their possible configurations. Based on component-based design, the blockchain ontology decomposes the blockchains into individual functional or logical components and identifies any possible different layout. The blockchain ontology proposes to assist in the design, implementation, deployment and performance measurement of different blockchain architectures.
The methodological approach is composed of the following steps (see Figure 1):

1. Comparative study of different blockchains. A pre condition is the analysis of vocabulary and terms to sort out ambiguities and disagreements. A literature review of the existing technologies is the starting point to limit complexity and organise information in schematic order. To avoid dis-ambiguities the analysis is supported by a merge of common blockchain terminologies developed so far in the literature and grouped in the an online database [?]. This brings together a vocabulary of key blockchain terms to provide readers with a foundation upon which to understand the classification and ontology developed in the rest of the analysis. The identification of the blockchain components is the crucial part of the analysis and it is conducted by running a comparative studies across *XX* different blockchain technologies. With regard to this, blockchains have been grouped in "digital currencies", "asset registries", "application stacks" and "asset-centric" technologies. See [5] for more information.

2. Framework setting. Components identification and classification. Following this comparative study, a hierarchical ontology (a tree structure of classifications for a given set

[?]URL: `http://arstweb.clayton.edu/interlex/`

of components) has been defined and populated by *main*, *sub* and, when necessary, *sub-sub* components.

3. Layout categorisation. Finally, for the components in the lowest level of the hierarchical structure, different layouts are introduced and compared. However, for the sake of simplicity, we limit our study to two or three layouts per each *sub* or *sub-sub* component.

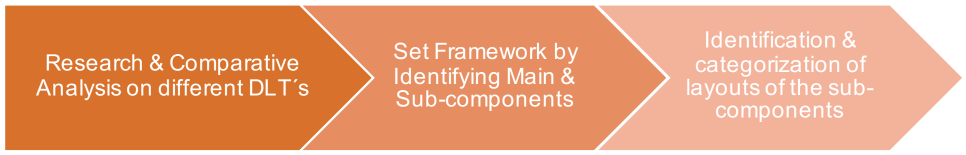

Research & Comparative Analysis on different DLT´s
Set Framework by Identifying Main & Sub-components
Identification & categorization of layouts of the sub-components

FIGURE 1: Methodological Approach

1.4 RESULTS

The result of the component-level analysis is a universal blockchain ontology that groups (in a hierarchical structure) the major components, identifies their functional relation and possible design patterns. This blockchain ontology can be of practical importance in many cases. For example, it can:

1. support software architectures to explore different system designs and to evaluate and compare different design options;

2. be propeadeutic to the development of blockchain standards with the aim to increase the adoption at large scale of blockchain-enabled solutions and services;

3. enable research into architectural framework for blockchain-based systems in order to boost the adoption of blockhain-enabled systems, their interoperability and compatibility;

4. create gateway models to multiple blockchains and design governance framework;

5. promote blockchain predictability;

6. be used to promote a regulatory framework that provides a mix of both legal and technical rules (i.e., regetech for blockchain-based systems).

2 BACKGROUND ON BLOCKCHAIN TECHNOLOGIES

2.1 BLOCKCHAIN KEY DRIVERS

As an ICT technology, a blockchain is driven by fundamental principles of data decentralisation, transparency, security and privacy, [6].

Decentralisation of consensus. The distributed nature of the network requires untrusted participants to reach a consensus. In blockchain, consensus can be on "rules" (that determine e.g., which transactions are allowed and which are not, the amount of bitcoins included in the block reward, the mining difficulty, etc.) or on the history of "transactions" (that allows to determine who own what). The decentralised consensus on transactions govern the update of the ledger by transferring the responsibilities to local nodes which independently verify the transactions and add them to the most cumulative computation throughput (longest chain rule). There is no integration point or central authority required to approve transactions and set rules. No single point of trust and no single point of failure.

Transparency. Records are auditable by a predefined set of participants, albeit the set can be more or less open. For example, in public blockchains everyone with an Internet connection to the network hold equal rights and ability to access the ledger. The records are thus transparent and traceable. Moreover, every participants to the network can exercise their individual (weighted) rights (e.g. measured in CPU computing power) to update the ledger. Participants have also the option to pool together their individual weighted rights.

Security and immutability. Blockhains function under the principle of non-repudiation and irreversibility of records. Blockchain is a shared, tamper-proof replicated ledger where records are irreversible and cannot be forged thanks to one-way cryptographic hash functions and community consensus. Immutability eliminates the need for reconciliations providing a historical, unique reconciliated version of the truth. Non-repudiation, non-forgeability and immutability of the records generate trust in the historic transaction flow. History is thus recorded in perpetuity. Indeed, it becomes very difficult for an individual or any group of individuals to tamper with the ledger, unless these individuals control the majority of "voters".

Other non fundamental properties of blockchain include data automation and data storage capacity.

Automation and smart contracts. Without the need for human interaction, verification or arbitration, the software is written so that conflicting or double transactions are not permanently written in the blockchain. Any conflict is automatically reconcile and each valid transaction is added only once (no double entries). Moreover, automation regards also the development and deployment of smart legal contracts (or smart contract codes, see [7]) whith payoff depending on algorithms which are self-executable, self-enforceable, self-verifiable and self-constraint.

Storage. The storage space available on the blockchain networks can be used for the storage and exchange of arbitrary data structures. The storage of the data can have some size limitations placed to avoid the blockchain bloat problem [8]. Metadata can be used for example to issue meta-coins: second-layer systems that exploit the portability of the underlying coin used only as "fuel". Any transaction in the second layer represents a transaction in the underlying network. Alternatively, the storage of additional data can occur "off-chain" via a private cloud on the client's infrastructure or on a public (P2P or third-party) storage. Some blockchains like Ethereum allow to store data also as a variable of smart contracts or as a smart contract log event.

2.2 BLOCKCHAIN CATEGORISATION

Since the Bitcoin inception in 2009, many blockchain software architectures have been deployed to meet different technical, business and legal design options. Given the current complex dynamic of the blockchain architectural development, it would be neither exhaustive nor comprehensive to provide a picture of the existing blockchain technologies developed so far. Therefore, as in [5] we limit our analysis to the introduction of four general groups according to thw blockchains' functional properties: *digital currencies*, *application stacks*, *asset registry technologies* and *asset-centric technologies*.

Digital currencies. Although Bitcoin is still the most widely adopted currency and parent of many others that forked from it, at the moment of writing there are more than seven hundreds digital currencies in circulation with positive capitalisation (https://coinmarketcap.com/). Digital currencies are founded on: (1) decentralised peer-validated time-stamped ledgers (instead of trust-based centralised ledgers); (2) cryptographic hash function message encryptions (although some digital currencies like Ripple do not rely on cryptographic techniques, all cryptocurrencies are digital currencies); and (3) proof-of-concept principles. These features allow for: (1) anonymous and trust-less pwwr-to-peer network transactions; (2) authenticity verification of the network transactions and automatic bookkeeping of the public ledgers; (3) expansion of the total amount of coins (i.e., monetary base) at a constant (or controlled) pace. Digital currencies are money expressed as a string of bits sent as a message in a network that verifies the authenticity of the messages and their order or arrival via different mechanisms such as Proof-of-Work (PoW) or Proof-of-Stake (PoS). Most digital currencies exhibit a publicly visible distributed ledger which is shared across a computing network. A comparison between digital currencies should be based both on technical and monetary functional properties. Technical properties include the governance model, security and transaction capability, i.e., how many transactions are executable per unit of time. Monetary properties include accountability, divisibility, volatility, fungibility and money supply rate. A part from Bitcoin, the most widely adopted digital currencies are Ether (https://www.ethereum.org/), Ripple (https://ripple.com/) and Litecoin (https://litecoin.com/).

Application stacks. Application Stacks are "non-currency" blockchain-based platforms used for the development and execution of complete applications on top of decentralised networks like Bitcoin. By complete applications we mean smart contracts, decentralised applications (Dapps) and Distributed Autonomous Organisations (DAOs). A smart contract can be considered as a simplified type of decentralised application. It is a set of "rules by code" which facilitates the automatic agreement among two or more parties (in fix number), the identification of the legal socio-economic function of the act, its object, its form and its final conclusion. The entire process is automated and the contractual terms are recorded in a computer language as a set of instructions. Smart contracts are entirely digital and written using dedicated programming code languages such as Solidity, LLL or Serpent (in Ethereum blockchain), or generic languages such as JavaScript and Python. The code defines the rules and consequences of the agreement. An encrypted version of the code is sent out to other computers (counterparties) via the blockchains. In a public permissionless blockchain like Bitcoin, the contract is broadcast and the ledger updated similarly to the way that a network update of a Bitcoin transaction would occur. This can also be done in a permissioned or hybrid distributed ledger platform such as the R3's CORDA distributed ledger. Execution (unique or continued) and enforcement of the agreement are also automated: self-execution and self-enforcement. Smart contracts facilitates the parties to upload digitised assets on the blockchains and automatically redistribute them and eventually their revenues and losses among the parties according to specific conditions based on the occurrence of certain events that are predefined at the time the contract is initiated. As any traditional organisation, a decentralised organisation (DO) is governed under specific divisional, functional structures according to which, decisions are taken (at different levels along the hierarchy) based on predetermined set of rules, routines and codes of conduct. The DO simply brings a centralised organisation process and decentralise it. In this respect, blockchain shifts the boundary between hierarchical organisations and non-territorial, spontaneously ordered, self-organising economies where, although the decision-making process is still controlled by humans, any internal and external transaction occur under an incorruptible set of business rules. For example DOs use on-chain voting systems, on-chain accounting and production systems, on-chain shareholders registry, etc. T The information is managed and process into the DO by the humans which control the information flow. Thus, humans are in control of the organisation because they control the source of information and the set of rules. Differently from the DO model, in the Decentralised Autonomous Organisations (DAOs) the decision-making process is not anymore in control of the humans but in some fashion it is in control of the DAO itself. The DAO is composed of a pool of smart contracts and autonomous agents linked together and endowed with an initial capital. Through a pre-defined enforceable tamper-proof set of rules coded into smart contracts, a DAO can run business or social activities (either online or off-line) completely autonomously from human intervention. Differently from traditional hierarchical organisations where the information flow is (sometime) opaque, DAOs use a open-source blockchain software environment which are decentralised (distributed across the computers of the stakeholders), transparent, secure and auditable. Current application stacks that allow for implementation of decentralised automation are NXT, Ethereum, and Monax, which distinguish themselves based on their core functions.

Asset registry technologies. Asset registry technologies refer to all those applications that link digital currencies to other assets or records on top of the blockchain. Asset registry applications refer to "ledger within a ledger": literally meaning that a ledger is embedded within the other and parsed independently. These applications provide some specific registry services that benefit the users from an open, shared, and irreversible ledger which is linked to some real assets or records and built on top of an existing digital currency like Bitcoin. For such a reason, asset registry technologies also refer to "meta-coins", which are second-layer systems that exploit the portability of the underlying coin used only as fuel. Any transaction in the second layer represents a transaction in the underlying network. For example, if Bitcoin is used as fuel, the benefits are linked to its low-cost of recording transactions, its immutability, cannot be non-counterfeit, robustness, transparency and traceability, low transaction fees. When an asset registry transaction is broadcast to the Bitcoin network they are verified by Bitcoin miners and saved in the Bitcoin blockchain to make a secure, verifiable record. While originally designed to support the exchange of digital currencies, blockchain scripting languages have the potential to store small amounts of metadata on the blockchains. Metadata summarises information about the most diverse types of information (from images to texts and source codes) that users can store in the blockchain (see [9]). Alternatively to metadata stored in a blockchain transaction via specific opcodes, some blockchains like Ethereum allow to store arbitrary structured data as a variable or as a log event in a smart contract. We can identify three main categories of use cases which include asset registry applications: 1) Financial (e.g., currencies, equities, commodities, derivatives), 2) Records (e.g., of spending, trading, voting, accounting) and certifications (e.g., grades, permits, copyrights), and 3) Ownership (e.g., Security tokens, coupons, vouchers). The most widely used asset registry applications include: Colored Coins, Counterparty, Mastercoin and Namecoin.

Asset-centric technologies. Asset-centric technologies are very powerful for finite closed groups of participants who want to deploy on a common blockchain the exchange of data and value in a immutable, transparent and automatic fashion. These type of asset-centric technologies focus on the exchange of digital tokens representing real assets, such as currencies, stocks and bonds, in combination with a shared ledger, but not on a public ledger. Generally, participants have a business or social relationship even before joining the platforms. Therefore, trust is organised directly "off-chain" between participants, rather than through a mining process as in the case of Bitcoin. Since the participants (nodes) are not anonymous but rather cryptographically identifiable, the consensus among anonymous in a distributed network is not the goal of asset centric technologies. Rather, they use different consensus protocols to agree on how to updated the public ledger. Specifically, they adopt extensions and variations of the General Byzantine Fault Tolerance (GBFT) consensus mechanism to incorporate that nodes are not anonymous. Stellar, Ripple, Hyperledger, and Namecoin are the most famous examples of asset-centric technologies. These decentralised ledger infrastructures allow for the exchange of real assets like currencies, metals, stock, or bonds and in some cases even without the need to use a native digital token as in the case of Hyperledger. The element that distinguishes these asset-centric technologies with respect to the other blockchain technologies is the fact that they represent distributed exchangers whose users can exchange among themselves various heterogeneous on/off chain assets. Participants on the network commit to quoting digital assets on the network in various denominations such as USD, gold, Bitcoins. At the same time, some of the participants are responsible for converting these assets, acting as "gateways" by bridging the gap between the physical and virtual worlds. In order to exchange one digital asset for another, a market maker is required and this role is usually played by the managing companies behind the asset-centric platforms or by authorised financial intermediaries.

3 ONTOLOGY OF BLOCKCHAINS

The diversity of blockchain research and development provides an opportunity for cross-fertilisation of ideas and creativity, but it also can result in fragmentation of the field and duplication of efforts. One solution is to establish a standardised software architecture to map the field and promote coordinated research and development initiatives. However, in terms of blockchain software architecture design little has been proposed so far [10], and the problem of consistently engineering large, complex blockchain systems remain unsolved. We approach this problem by proposing a component-based blockchain ontology starting from a coarse–grained connector–component analysis. The ontology compartmentalises the blockchain connectors/components and establishes the relationships between them in a hierarchical structure. Section 2 shows that a blockchain is composed of several fundamental components interrelated between them. In order to explore all the possible domains of blockchain components and their topological layout indicating their runtime interrelationships, we conduct a comparative study across different families of blockchain applications: digital currencies, application stacks, asset registry technologies and asset centric technologies. Table 1 summarises the blockchains analysed in this study.

We adopt a reverse-engineering approach to unbundle the blockchains and divide them in *main* (coarse-grained) components. Each main component is then split in *sub* (fine-grained) and, where necessary, *sub-sub* components. For each these sub (or sub-sub) components, different *layouts* (models) are identified and compared. By deriving the logical relation between (main and sub or sub-sub) components, the study helps to clarify the alternative *modus operandi* of the blockchains and helps to develop the conceptual blockchain design and modelling.

The software engineering approach used to derive the blockchain ontology – similarly to fields like electronics or mechanics – threats blockchains as the result of gluing together prefabricated components. Although providing similar services and functions, components can be of different importance and type, and can be connected between them in different fashion. Just to provide a simplified analogy to better understand our component-based approach, one could think of an ontology developed in order to compare different cars between them. In that case, an example for a main component could be the cars' engine. From the engine, various subcomponents, e.g. the fuel pump, can be identified and categorised. However, the fuel pump can have different layouts. For example, two possible layouts could be the mechanical or the electronic fuel pump. Despite being different, the two types of pump serve the same scope of enabling the engine to work properly. A parallel with the blockchain could be that the car's engine represents the consensus and the fuel pump represents the consensus mechanism. Finally, Proof-of-Work and Proof-of-Stake could be understood as possible different layouts for the consensus mechanism. Following this logic, each of the next seven sections will introduce a new blockchain main component and its sub (and eventually sub-sub) components by describing and comparing their layouts.

Family	Example of Technology	Description
Digital Currency	Bitcoin	Forerunner and by far the most widely used cryptocurrency.
	Dash	Privacy-centric digital currency. The digital currency Dash has different functionalities e.g. instant transactions. Dash is based on the Bitcoin source code, but it allows anonymity while performing transactions.
	Monero	Monero is an open-source digital currency, with the focus on decentralisation, scalability and privacy. It is based on the CryptoNote protocol, which has an enabled anonymous layer.
	LiteCoin	LiteCoin is a P2P cryptocurrency and open source software project. The Litecoin is technically nearly identical to Bitcoin, except for the proof-of-work cryptographi function used.
	Zcash	Zcash is a decentralised and open-source cryptocurrency, which combines privacy with selective transparency of transactions.
	Peercoin	Cost-effective and sustainable cryptocurrency based on proof-of-stake.
Asset Registry	ColorCoin	A concept that allows attaching metadata to Bitcoin transactions and leveraging the Bitcoin infrastructure for issuing and trading immutable digital assets that can represent real world value.
	Omnilayer (MasterCoin)	A meta-protocol layer that enables new digital currencies, digital assets, and communication protocol to existing on top of the Bitcoin blockchain.
	NameCoin	NameCoin is an asset registry on top of the Bitcoin Blockchain, which enables a decentralised domain name system.
	Counterparty	Counterparty enables anyone to write specific digital agreements or programs known as smart contracts, and execute them on the Bitcoin blockchain.
Application Stack	NXT (Ardor)	NXT is a safe, transparent and decentralised system for sharing data and allowing payments to people all over the world. Ardor platforms enable smart contract functions with NXT.
	Ethereum	Ethereum is an open-source platform to build blockchain-based applications in different business fields.
	Monax	Monax is an open platform for developers and DevOps to build, ship, and run blockchain-based applications for business ecosystems. Monax is known as a private blockchain offered by the monax company.
	Cosmos	Cosmos is an architecture for cross-chain interoperability where independent blockchains can interact via an inter-blockchain communication (IBC) protocol, a kind of virtual UDP or TCP for blockchains each driven by the Byzantine fault tolerant (BFT) consensus algorithm, similar to Tendermint.
	COMIT	COMIT is a cryptographically-secure off-chain multi-asset instant transaction network (COMIT) that can connect and exchange any asset on any blockchain to any other blockchain using a COMIT cross-chain routing protocol (CRP).
	Synerio	Synerio is intended to become a decentralised social network.
Asset Centric Technologies	Ripple	Ripple is a global real-time financial settlement provider.
	Stellar	Stellar is a decentralised multicurrency-exchange platform for people without access to the banking system.
	Hyperledger	Hyperledger (or the Hyperledger project) is an open source blockchain platform, started in December 2015 by the Linux Foundation, to support blockchain-based distributed ledgers. It is focused on ledgers designed to support global business transactions, including major technological, financial, and supply chain companies, with the goal of improving many aspects of performance and reliability.
	Tendermint	Tendermint is a high-performance blockchain consensus engine that enables to run Byzantine fault tolerant applications written in any programming language. Tendermint is a partially synchronous BFT consensus protocol derived from the DLS consensus algorithm.
	Corda	Open-source distributed ledger platform.
	Enigma	Distributed ledger technology, created by the MIT digital currency lab.

TABLE 1: Blockchain Technologies.

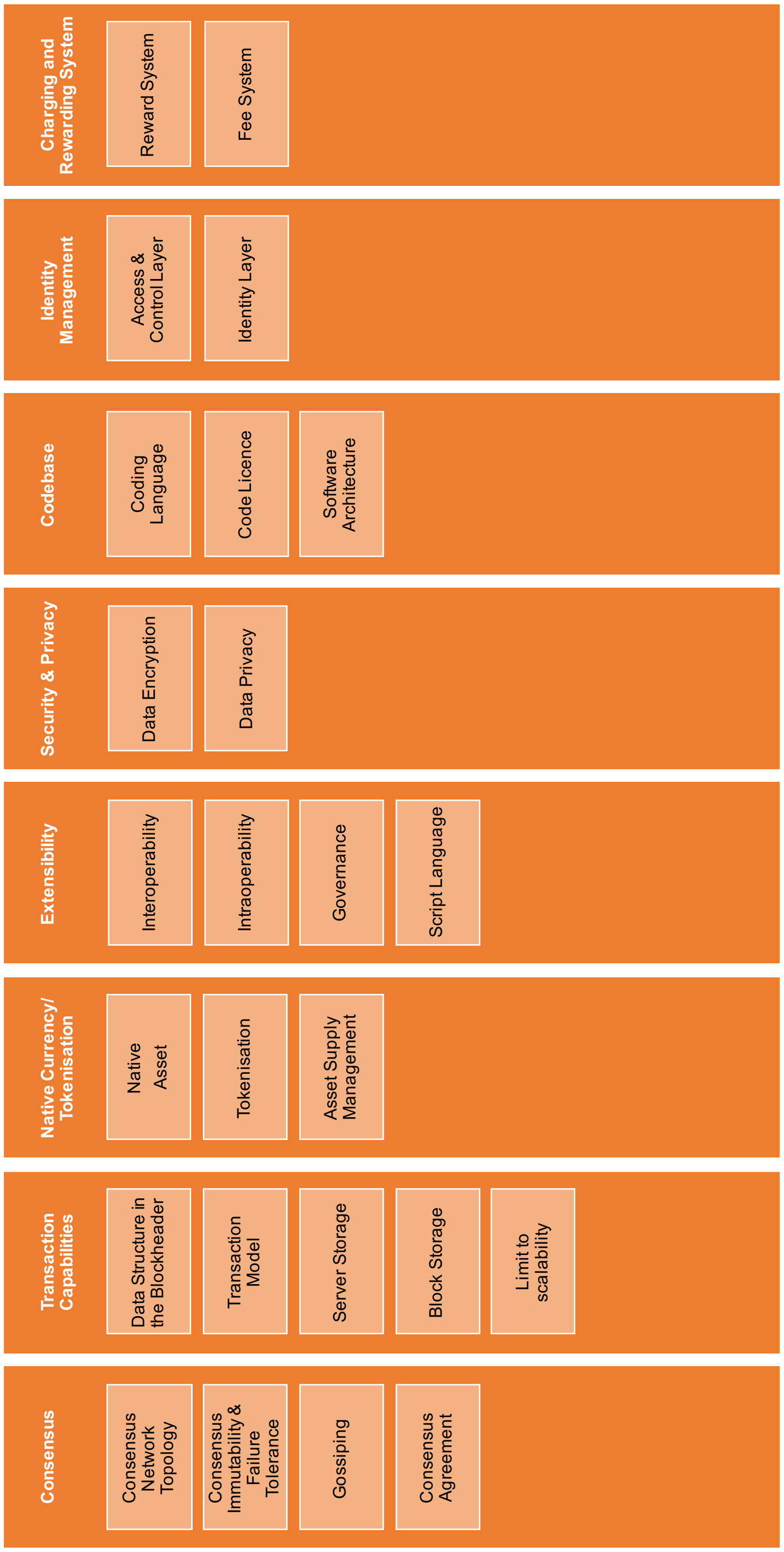


FIGURE 2: Complete Overview of the main- and subcomponents

4 CONSENSUS

The first identified main component is *Consensus*. It relates to the set of rules and procedures that allow to maintain and update the ledger and to guarantee the trustworthiness of the records in the ledger, i.e., their reliability, authenticity and accuracy. *Consensus* varies across different blockchain technologies, every consensus mechanism brings advantages and disadvantages based on different characteristics e.g. speed of transactions, energy efficiency, scalability, censorship-resistant and tamper-proof [11]. The set of rules and procedures compose the framework of the validation process necessary to overcome security issues during the validation. *Consensus* includes the following subcomponents and sub-subcomponents:

1. *Consensus Network Topology*
2. *Consensus Immutability and Failure Tolerance*
3. *Gossiping*
4. *Consensus Agreement*
    (a) *Latency*
    (b) *Agreement*

Those subcomponents are to be jointly considered when designing an active network consensus validation process because not only their individual configuration but also their combination determines when and how the overall blockchain agreement is achieved and the ledger updated.

4.1 CONSENSUS NETWORK TOPOLOGY

Consensus Network Topology describes the type of interconnection between the nodes and the type of information flow between them for transactions and or validation purpose. For efficiency reasons, historically, systems have been designed in a centralised manner. This centralisation lowers dramatically the costs for system configuration, maintenance, adjustment (and the costs of arbitration in case of conflict) as this work has to be performed only once in a central place. While highly efficient in many situations, these kind of systems induce a single (or very limited set of) point(s) of failure and suffers of scalability issues. With respect to the network topology, this hierarchical arrangement is still present in most our techno and socio-economic systems (one example is the modern electronic payment system). Enlarged,

these centralised arrangements become hierarchical constructs which exhibit larger scalability and more redundancy, while keeping efficient communication. Alternative to those centralised topologies, decentralised solutions have been proposed. Already since the dawn of the Internet, technical systems have evinced a transition towards decentralised arrangements [12] where all the nodes are equal to each other. For most applications, blockchain based systems - with its federated set of participants - is a clear example of this kind. Blockchain based systems resort on specific network topologies to create the peer network that ultimately determines how the validation process will evolve.
It is important to mention that *Consensus Network Topology* is linked to the level of (de)centralisation in the validation process but it is not the only determinant. Also other factors like for example the rewarding mechanism (see Section 11) influence the validation. Take for example Bitcoin where a decentralised validation process may still be accompanied by the concentration of validation power of network participants. Indeed, during the period 2013-2015, the cumulative market share of the largest ten pools relative to the total market hovered in the 70%–80% range, [5].

LAYOUTS

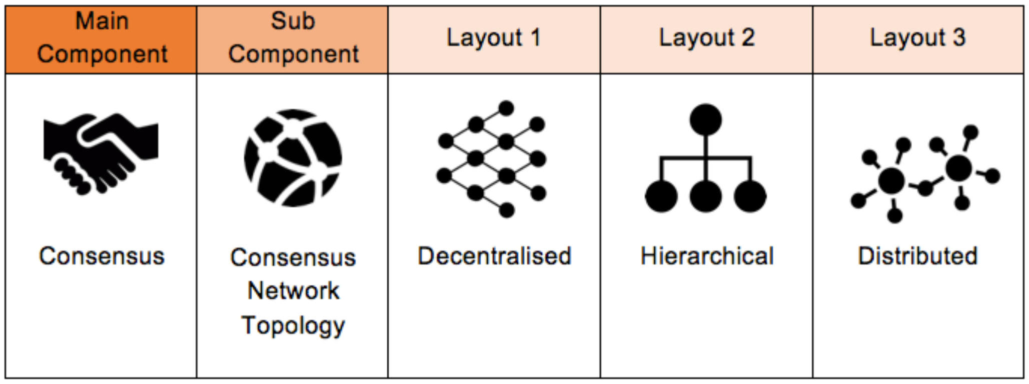
We identify three possible layouts for *Consensus Network Topology*:

1. **Decentralised.** There exist implementations that are decentralised. Bitcoin as the pioneer in digital currencies established a distributed P2P network, which enables direct transactions to every node within the network. The validation process within the Bitcoin network is decentralised through miners and full nodes who validate the transactions within the network [1]. This network illustrates a decentralised *Consensus Network Topology*. Obviously, this layout is independent of the *Consensus Immutability* layout (Peercoin and NXT also show decentralised network topologies).

2. **Hierarchical.** There are other implementations that are not decentralised, and there exists an inhomogeneity of the role the nodes have. For example, in Ripple the network topology is divided into tracking and validating nodes. The tracking nodes are the gateway for submitting a transaction or executing queries for the ledger in addition to that the validating nodes have the same functions as tracking nodes, but they can also contribute additional sequences to ledger by validation [13]. This kind of solution yields (or can be extrapolated to) a hierarchical network topology. In the community of developers these hierarchical topologies are usually referred to as “Consortium blockchains”.

3. **Centralised.** In some specific implementations, a central authority may need (or wish) to

control what is added to the ledger (an example for this are digital versions of fiat currencies: the so called central bank digital currencies). This kind of solution yields a third layer, “Centralised topology”, which is intimately related to private blockchains. It is important to mention that a centralised solution would normally speak of a non-properly working design (ar a non-solution) if implemented in terms of a blockchain, as it would have been implemented otherwise in a more transparent manner. Normally, some level of federation and redundancy are key to blockchain systems.



Main Component	Sub Component	Layout 1	Layout 2	Layout 3
Consensus	Consensus Network Topology	Decentralised	Hierarchical	Distributed

FIGURE 3: *Consensus Network Topology* - Ontology Matrix

4.2 CONSENSUS IMMUTABILITY AND FAILURE TOLERANCE

The development of blockchain technologies in the past years has been coupled with an increasing number of different mechanisms that help to keep immutable the information contained in the blockchain. In this vein, the immutability of achieved consensus differs with respect to the resources required to keep high the security of the network. It represents the validation process which guarantees that the agreed consensus will not change in the future and the full data history is recorded in perpetually. Indeed, the decentralised solution represented by blockchains to the storage of information requires no central database and many duplications where all the servers hold a *replica* of the ledger. Any new record is costly (often measured in terms of computational power) to be added to the ledger, but cheap to be verified by peers. At the same time, the mechanisms for *Consensus Immutability* together with the subcomponents of *Consensus Agreement* determine that *Failure Tolerance* of the blockchain. The Byzantine failure tolerance is the characteristic of a system to tolerates the class of failures known as the Byzantine Generals’ Problem in which components of a system fail with symptoms that prevent some components of the system from reaching agreement among themselves, where such agreement is needed for the correct operation of the system [14].

LAYOUTS

We identify six main layouts for *Consensus Immutability and Failure Tolerance*:

1. **Proof-of-Work**. Bitcoin uses Proof-of-Work (PoW) to ensure the immutability of the transaction records. In this setup, miners connected to the network perform the task of validating the transactions proposed for addition to the blockchain by solving the inversion of a cryptographic function whose solution can only be found by brute force. In PoW the probability of mining a new block depends on the instantaneous computational power devoted to the task by all miners connected to the network under the rule "one CPU one vote". As of this writing, the mining process needs several requirements to be successful [15]. These include specialised hardware which is needed to perform the computational tasks and ever increasing amounts of electricity to power the hardware. A clear drawback of the PoW mechanism is the inherent inefficiency from the resource point of view, and the large-scale investments needed, which has led to long-term centralisation of the mining power. At the moment of writing, in every second almost five quadrillion SHA256 computations are performed by the Bitcoin network. Regretfully, these computations do not have any practical or scientific relevance a part from protecting the system from sybil attacks. However, when adversaries coordinate, it is sufficient that they hold only the 25% of the total computing power to mount an attack [16]. In contrast, BFT consensus mechanisms tolerate at most n/3 corrupted nodes in the asynchronous communication protocol and even higher levels in the synchronicity case. Electricity consumption can be estimated around 0.1 to 1 W/GH corresponding to around 1GW of electricity consumed every second. Therefore, other developers within the area of blockchain technologies continuously attempt to develop novel mechanisms to achieve the same goal.

2. **Proof-of-Stake**. Proof-of-Stake (PoS) links the validation to the proof to show ownership of a certain amount of digital assets (e.g., digital currencies) linked to the blockchain. The larger the stake is, the higher the probability for the participant (prover) to be considered trusted validator [11]. Two alternative PoS methods have been devised. The first is the randomised block selection (used by e.g., NXT and BlackCoin) which proposes a formula that looks for the lowest hash value in combination with the size of the stake. The second is the coin-age based selection (used by e.g., Peercoin) which combines randomisation with coin-age (a number derived from the product of the amount of the as-

sets held by the prover and the length of time it has been held for). Although PoS has the chance to solve two issues with PoW (risk of monopoly mining and resources wasted in the mining process), it is affected by the "nothing at stake" issue. Because there is little cost in working on several chains (unlike in PoW), one could abuse by voting for multiple blockchain-histories which would prevent the consensus from ever resolving (double spending). This problem has been addressed by Delegated Proof of Stake (DPoS), a generic term describing an evolution of the basic PoS consensus (utilised by e.g., BitShares, Casper by Ethereum, Tendermint) where blocks are forged by a predetermined set of users who are rewarded for their duty and are punished for malicious behaviour (such as participation in double-spending attacks). This principle of pre-authorised forgers is generalised by the Proof-of-Authority mechanism.

3. **Proof-of-Authority**. In this case, participants are not asked to solve arbitrarily difficult mathematical problems like in PoW, but instead they are asked to use a hard-configured set of "authorities" empowered to collaborate "trustlessly". Namely, some nodes are exclusively allowed to create new blocks and secure the blockchain. Typically, Proof-of-Authority (PoA) mechanism fits well for consortium private networks where some pre-selected real entities (i.e., the authorities) are allowed to control the networks. Those nodes will receive a set of private keys that will used to "sign" the new blocs. Simply stated, the main idea behind the PoA scheme is that blocks can be verified only by *trusted signers*. Thus, every block (or header) that a client sees can be matched against the list of trusted signers. The challenges brought by PoA are related to: 1) control of mining frequency, 2) distribution of mining load (and opportunity) between the various signers and; 3) maintainance of the the list of signers such to be robust from malicious attacks even in presence of dynamically mutation of the trusted signers.

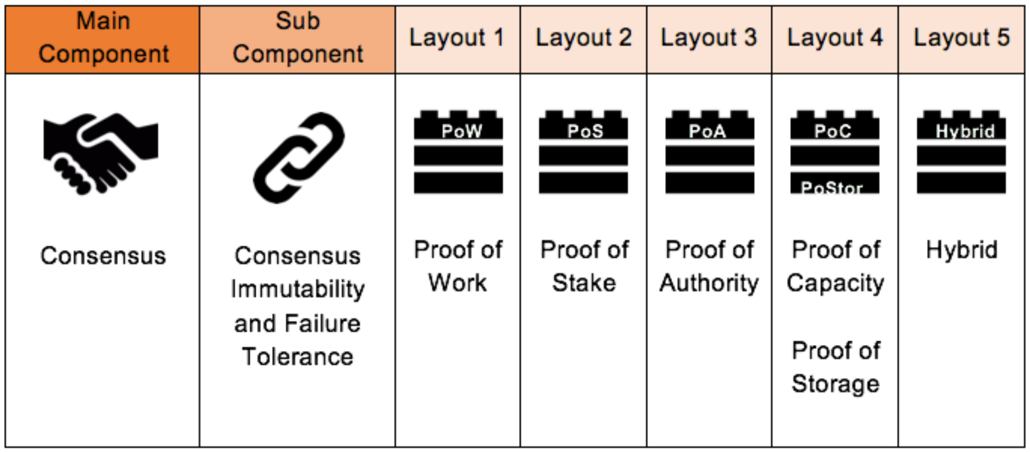
4. **Proof-of-Capacity and Proof-of-Storage**. Proof-of-Capacity (PoC) also called Proof-of-Space (PoSpace) and Proof-of-Storage (PoStorage) are implementations of the popular idea of "megabytes as resources". Here the focus is not on the CPU cycles but on the amount of actual memory the prover must employ to compute the proof. Nodes are asked to allocate significant volume of their hard drive space to mining instead of using CPU-bound space as in PoW. Miners are incentivised to invest in hard-drive capacity as those who dedicate more disk space have a proportionally higher expectation of successfully mining a block and reaping the reward. The PoC makes use of hash trees to efficiently allow verification of a challenge without storing the tree. The PoC is a more fair and green scheme that PoW. The reason mainly comes from the lower variance of

memory access times between machines and the lower energy cost achieved through the reduced number of computations required. Several practical implementations adopt the PoC consensus algorithm like Permacoin, SpaceMint and Burstcoin, to cite a few. In PoW, miners are continuously asked to brute-force a one-way hash function by computing new hash values based on the combination of the previous hash values contained in the message, the new transaction block and a nonce, such that the new hash value will start with a given number of zeros ≤ target. Those computations are run by dedicated machines (ASICs) which are very espensive and not useful in itself and are resource-intensive as contribute to a large electricity expense for cryptocurrency miners [17]. As explained by [18], PoC consists of an initialisation and (sometime later) of an execution phase between a prover P and a verifier V. Rather than P proving to V that some amount of work has been completed, P proves to V that she has allocated some number of bytes of storage After the initialisation phase, P is supposed to store some data F of size N. Instead, V only holds some small piece of information. At any later time point V can initialise a proof execution phase, and at the end V outputs reject or accept. In an (N0, N1, T)–PoC the miner shows that she either: 1) had access to at least N0 storage between the initialisation and execution phases and at least N1 space during the execution phase; or 2) used more than T time during the execution phase. Solutions to the "Mining multiple chains", and "grinding blocks" problems of PoC algorithms have been proposes by [19] among the others. The Proof-of-Storage (PoS) mechanism is similar to PoC but the designated space in it is used by all participants as common cloud storage [20].

5. **Proof-of-Burn**. In Proof-of-Burn (PoB) miners must prove that they burned some digital assets. They do so by sending those assets (e.g., digital currencies) to a verifiable unspendable address belonging to them. Similarly to the PoS, also the PoB logic is to minimise the waste of resources generated by PoW. However, at the current stage, all PoB mechanisms function by burning PoW-mined digital currencies. This is therefore an expensive activity as the digital currencies once required to work as "fuel" in a PoB system cannot be recovered. PoB can be used also to bootstrap a token off of another (see e.g., Counterparty or Mastercoin).

6. **Hybrid**. The more advances hybrid consensus immutability and failure tolerance methods are "PoB and PoS" where Proof-of-Burn blocks act as checkpoints and "PoW and PoS" where PoW blocks act as checkpoints containing no transactions, but anchor both to each other and to the PoS chain. Peercoin uses PoW/PoS consensus. To solve the "nothing at stake" issue, Peercoin uses centrally broadcast checkpoints (signed under the de-

veloper's private key) according to which no blockchain reorganisation is allowed deeper than the last known checkpoints. Here the problem is that the developer becomes the central authority controlling the blockchain.



Main Component	Sub Component	Layout 1	Layout 2	Layout 3	Layout 4	Layout 5
Consensus	Consensus Immutability and Failure Tolerance	PoW Proof of Work	PoS Proof of Stake	PoA Proof of Authority	PoC PoStor Proof of Capacity Proof of Storage	Hybrid Hybrid

FIGURE 4: *Consensus Immutability and Failure Tolerance* - Ontology Matrix

4.3 GOSSIPING

Blockchains are also decentralised, redundant storage systems. This redundancy makes it very difficult to hijack the information stored in them. How this information travels through the network of computers is a characteristic that varies from one blockchain system to another. Given the lack of a central routing authority (like it would exist for example in traditional electronic payment systems) nodes must transmit the information they possess – in general new blocks, but it may be also the full blockchain to new nodes that enter the network – to peers they know are participating of the system. To this aim, nodes possess a list of peer nodes. Whenever a new block is added to the local blockchain of a node, the later passes the block to others in its peer list by *Gossiping*.

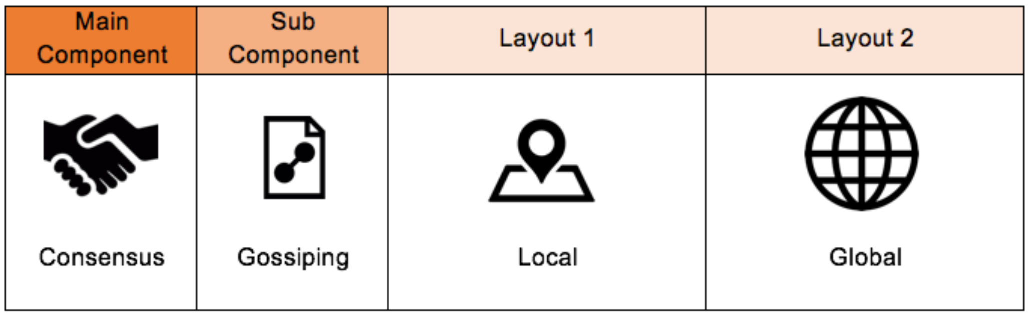
LAYOUTS

We identify two possible layouts for *Gossiping*:

1. **Local.** *Gossiping* occurs first in a local manner (through a local validation process) until consensus is reached. This is also called "federated consensus" used e.g., in Ripple [13] in which nodes can share transaction records to another node and reach consensus without directly knowing all the nodes in the network. Therefore most information travels

"locally" – in terms of the P2P network – such that a consensus is reached at this initial level. Only then, the information is send throughout all the other nodes. In this layout, the *Gossiping* can be termed "local".

2. **Global.** In most implementations – Bitcoin, Ethereum, etc. – *Gossiping* occurs to a list of peers that have been selected by what in the Bitcoin network are called fallback nodes. These fallback nodes maintain a list of all peers in the network and upon connection of a new node, submit a randomly chosen list of peers to the entrant node. In this way, the *Gossiping* process becomes *global*.



Main Component	Sub Component	Layout 1	Layout 2
Consensus	Gossiping	Local	Global

FIGURE 5: *Gossiping* - Ontology Matrix

4.4 CONSENSUS AGREEMENT

The *consensus agreement* defines the set of rules under which records of transactions are independently updated by the nodes of a distributed systems. This is important to understand how a distributed system is able to handle the so called Byzantine failures, i.e., how the system composed of n nodes can achieve consensus in the presence of f malicious nodes ready to trigger sybil attacks. In this regard, it is very important to understand how the nodes communicate between them.

4.4.1 LATENCY

Latency is a sub-subcomponent which describes the rule of message propagation in the networks.

LAYOUTS

We identify two possible layouts for the subsubsection *Latency*:

1. **Synchronous Communication.** Systems which set upper bounds on ”process speed interval” and ”communication delay” such that every message arrives within a certain known time interval (Δ) are called *synchronous*. This does not preclude the possibility of having message delays due for example to network latency, but the delay is bounded and any message that takes longer than Δ is discarded. Synchronicity assumptions apply also to the Bitcoin blockchain. For example, a block is rejected if contains a timestamp: 1) lower than (or equal to) the median timestamp of the previous eleven blocks; and 2) greater than (or equal to) the ”network-adjusted time” + 2 hours. Another example of blockchain adopting *synchronous communication* is Ripple through the use of clocks. Specifically, the Ripple ”LastLedgerSequence” parameter assures that a transaction is either validated or rejected within a matter of seconds.

2. **Asynchronous Communication.** Systems which do not set any bound on ”process speed interval” and ”communication delay” such that every message/packet can take an indefinite time to arrive are called *asynchronous*. Although this type of communication protocols brings some advantages (e.g., calls/requests do not need to be addressed to active nodes and nodes do not need to be available when a new information is sent to them by peers) the main disadvantage of *asynchronous communication* is that response times are unpredictable and it is harder to design applications based on them. Synereo is an example of blockchain using the an asynchronous communication protocol.

4.4.2 FINALITY

Consensus *finality* is a sub-subcomponent which describes the probability of achieving consensus. A distributed system has a number of nodes which contribute to maintaining the state. In other words, the distributed system must have a shared or global truth that all nodes can agree on. While this concept is related to the idea of correctness (existence of a single global set of recognised transactions in the system), *finality* refers to the ability of the distributed system to be resilient to the same correct transaction made multiple times (this is also referred to as the problem of double spending) and against forks. Accordingly, consensus *finality* implies that if a correct node appends block ”M” to its copy of the blockchain before appending block ”N”, then no other correct node will append block ”N” before ”M” to its copy of the blockchain, [21].

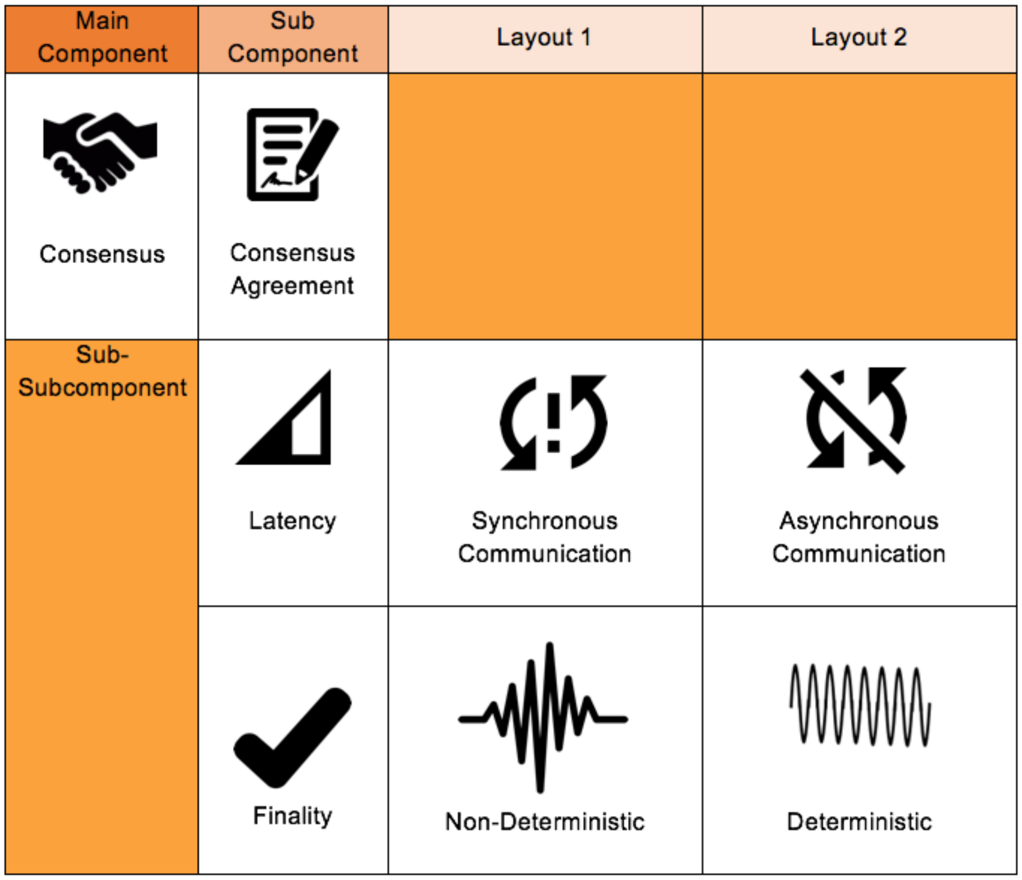
LAYOUTS

We identify two possible layouts for the sub-subsection *Finality*:

1. **Non-Deterministic**. In this case, *Consensus Agreement* “eventually converges”. Non de-

terministic are randomised or probabilistic consensus (also called stabilising consensus) in which the probability to disagree, decreases over time. For example in the Bitcoin blockchain the block frequency is adjusted (with respect to the block-mining rate and indirectly to the computational power of the nodes) such to minimise the possibility of forks. However, even in presence of honest nodes, the possibility of forks is not ruled out because the concurrency control mechanism (which ensures that correct results for concurrent operations are generated as faster as possible) is non deterministic and the propagation of he blocks over the network may have delays [22]. Thus, even though the heuristic “wait until 6 confirmed blocks are appended to the chain” reduces the possibility of forks, it does not eliminate the probability of a previously validated block to be pruned and removed from the blockchain in the future.

2. **Deterministic**. In this case, *Consensus Agreement* converges with certainty and transactions are immediately confirmed/rejected in/from the blockchain. This property turns to be a very useful for smart contracts where, using state-machine replication, consistent execution of the contracts can be achieved across multiple nodes. All the blockchains based on Lamport Byzantine Fault Tolerance achieve deterministic consensus.



Main Component	Sub Component	Layout 1	Layout 2
Consensus	Consensus Agreement		
Sub-Subcomponent	Latency	Synchronous Communication	Asynchronous Communication
	Finality	Non-Deterministic	Deterministic

FIGURE 6: Consensus Agreement - Ontology Matrix

5 TRANSACTION CAPABILITIES

The second main component, *Transaction Capabilities*, is important to illustrate scalability of transactions and usability in possible applications and platforms. One of the major challenges for the blockchain technology is to increase the transaction throughput to compete with other solutions already available in the market e.g. credit cards. In order to achieve these improvements, quantitative parameters e.g. data storage in block header, TPS (transactions per second) need to be redesigned to realise such improvements. In our description, *Transaction Capabilities* includes the following subcomponents:

1. *Data Structure in the Blockheader*
2. *Transaction Model*
3. *Server Storage*

4. *Block Storage*

5. *Limits to Scalability*

5.1 DATA STRUCTURE IN THE BLOCKHEADER

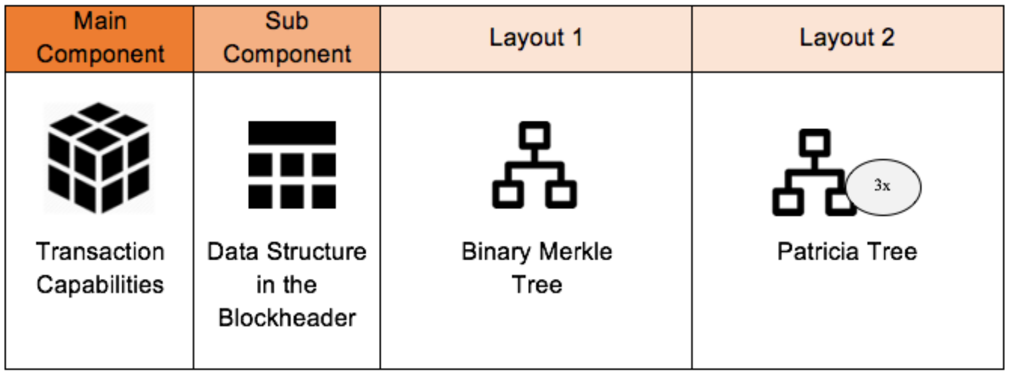
The data stored in the block header has different functions. On the one hand, it includes the transaction hashes for validation purpose; on the other, it contains additional information for different application layers or blockchain technology platforms. The *data structure in the block-header* describes the capabilities of the system to store transaction information. The original application of Merkle proof was implemented in Bitcoin, as described in [1].

LAYOUTS

We identify two possible layouts for *D*ata Structure in the blockheader:

1. **Binary Merkle Tree.** Bitcoin uses the Binary Merkle tree within the block header to store the transactions. The information in the block header in the Merkle tree structure contains a hash of the previous header, timestamp, mining difficulty value, proof of work nonce and root hash for the Merkle tree containing the transactions for that block, which are used for the verification process to scale up the transactions speed. By convention, the longest chain (since the so-called Genesis block) is considered to be the current status of the blockchain.

2. **Patricia Merkle Tree.** One the one hand, *Patricia Merkle Tree* (Practical Algorithm To Retrieve Information Coded In Alphanumeric [23]) allows activities like inserting, editing or deleting information referring to the balance and nonce of accounts, which enables faster and flexible validation of transactions than the one *merkle tree model* [24]. However with respect to the applications, it has the important advantage of allowing for verification of specific branches of the tree. Ethereum [25] uses the Patricia Merkle Tree within the block header to store more information than what is possible in the Binary Merkle Tree. Those contain transactions, receipts (essentially, pieces of data showing the effect of each transaction) and state [26]. Importantly, this technology allows even blocks outside of the longest chain to contribute to the validation process, building a confirmation system that is less centralised. This is the so called Ghost rule a variant of which is implemented also in the Ethereum blockchain [24].



Main Component	Sub Component	Layout 1	Layout 2
Transaction Capabilities	Data Structure in the Blockheader	Binary Merkle Tree	Patricia Tree

FIGURE 7: Data Structure in the Blockheader - Ontology Matrix

5.2 TRANSACTION MODEL

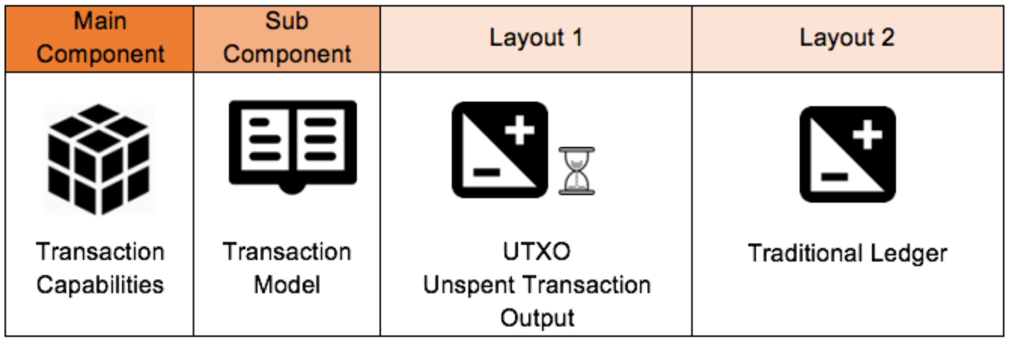
The challenge of the *transaction model* is to prevent fraudulent behaviour, like double spending, and money laundering from unknown participants. The transaction model can be imagined as an accounting ledger which tracks the inputs and outputs of each transaction. The *transaction model* describes how the nodes connected to the P2P network store and update the user information in the distributed ledger.

LAYOUTS

We identify two possible layouts for *Transaction Model*:

1. **The Unspent Transaction Output (UTXO).** *UTXO* model includes a refractory number of blocks during which network participants are prevented of using the transaction output in new transactions. In this way, it prevents miners from spending transactions fees and block rewards before stable validation status of the block chain. This measure prevents the *forking problem* of blockchains [27]. This transactions mechanism is available in blockchain technologies like Bitcoin.

2. **Traditional Ledger.** In comparison to the *UTXO model*, different smart contract companies like *Stellar* and *Ripple* use a more traditional ledger model to record the transactions. In particular Stellar lists every single transaction in the *Stellar* distributed ledger history. Also, *Ripple* uses the traditional ledger transaction model to register increment and clear account balances. In *Ethereum* transactions are used to execute actions in smart contracts. Those transactions can be seen as order executions of stakeholders which perform the actions out of said smart contracts.



Main Component	Sub Component	Layout 1	Layout 2
Transaction Capabilities	Transaction Model	UTXO Unspent Transaction Output	Traditional Ledger

FIGURE 8: Transaction Model - Ontology Matrix

5.3 SERVER STORAGE

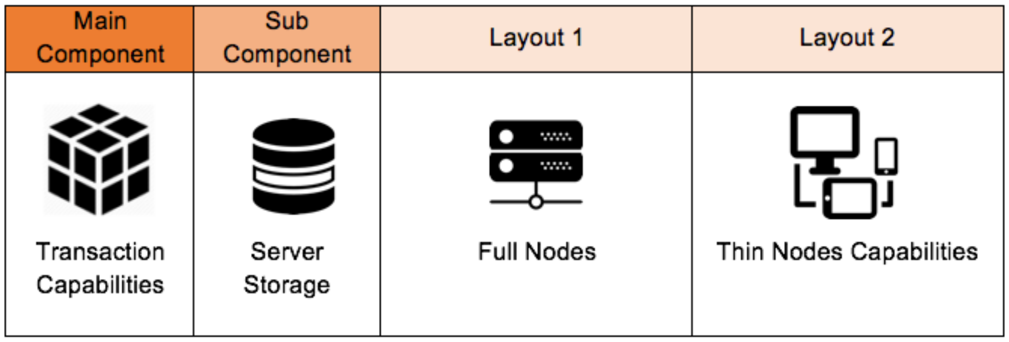
At the core of blockchain-based systems underlies their decentralised nature. This requires that nodes connected to the peer-to-peer network are indistiguishable from each other. This concept, however, cannot be fully expressed when the storage needs, computing power or bandwidth constraints of the network nodes do not permit this feature to be fully realised. In these scenarios, different nodes have access to different layers of information, those which do not store the information fully are “thin clients” connected to the peer-to-peer network.

LAYOUTS

We identify two possible layouts for *Server Storage*:

1. **Full Nodes.** All nodes connected to the network, and which are part of the validation process, are of the same kind. This is a genuinely peer-to-peer network where all the nodes are equivalent in terms of information contained. This property creates a large information redundancy, which makes the system more resilient to attacks or malfunctioning.

2. **Thin Nodes Capabilities.** In this setup, some nodes connected to the network contain only a selected subset of all the information contained in the blockchain. This creates more scalable systems (in terms of number of nodes connected to the network and the concomitant network traffic and storage needs), but may deteriorate the resiliency as only a fraction of the nodes contain the complete blockchain information.



Main Component	Sub Component	Layout 1	Layout 2
Transaction Capabilities	Server Storage	Full Nodes	Thin Nodes Capabilities

FIGURE 9: Server Storage - Ontology Matrix

5.4 BLOCK STORAGE

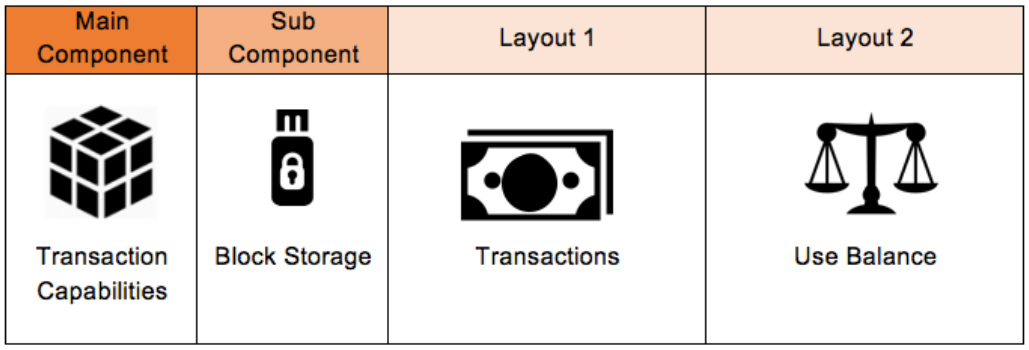
Which information is stored in the blockchain is determines the scalability of the system across some dimensions. More crucially, it also allows to understand how users and the concomitant information is abstracted within the system.

LAYOUTS

We identify two possible layouts for *Block Storage*:

1. **Transactions.** In systems like Bitcoin, only the transactions are stored. They contain both, a set of inputs and outputs that help to identify the emitter and receiver(s) of a specific transaction.

2. **User balance.** In systems like Ripple, the decentralised storage also contains information about the user balance in the specific assets.



Main Component	Sub Component	Layout 1	Layout 2
Transaction Capabilities	Block Storage	Transactions	Use Balance

FIGURE 10: Block Storage - Ontology Matrix

5.5 LIMITS TO SCALABILITY

The decentralised nature of blockchain systems and the concomitant redundancy in the storage impose different kinds of limits to the way in which a specific implementation scales when the *system size*. System size is used here in a broad sense: It refers to the number of nodes connected to the network, the number of users of the service, the set of network connections and amount of network traffic, the number of transactions, etc. It is worth remarking that these ingredients are intertwined in the real world [28], and - upon usage and continuous development - the limiting factor of a particular blockchain system may vary over time.
An important caveat here, properly defined, scaling is a property that should specify how the growth of a specific factor influences the overall performance of the system. Take for example the total network traffic induced used in *Gossiping*. If every node has a small - limited - number of connections, then the total traffic will scale linearly on the number of nodes. In mathematical terms it will be $\mathcal{O}(N)$. However, if every node is connected to each other then the traffic will be $\mathcal{O}(N^2)$, i.e. it will grow quadratically on the number of nodes. Therefore, if network traffic is the most crucial limiting factor to scalability, different systems may differ enormously still in their scalability. Acknowledging that a categorical definition is a crude simplification, we will focus here on the most important element for each system.

LAYOUTS

We identify three possible layouts for *Limits to Scalability*:

1. **Limit by number of transactions.** We start from the most common real-world example. Bitcoin has a limitation in the number of transactions it can process in every block, because of the hard-coded limit to the block size in bytes. Given that new blocks appear (on average) every ten minutes, this means that the number of transactions that can be included in a specific point of time is limited. Therefore the layout "Number of Transactions" refers - regardless of the information stored in the blockchain[29] - to the specific implementations when the number of operations that can be included in the blockchain is severely limited by design.

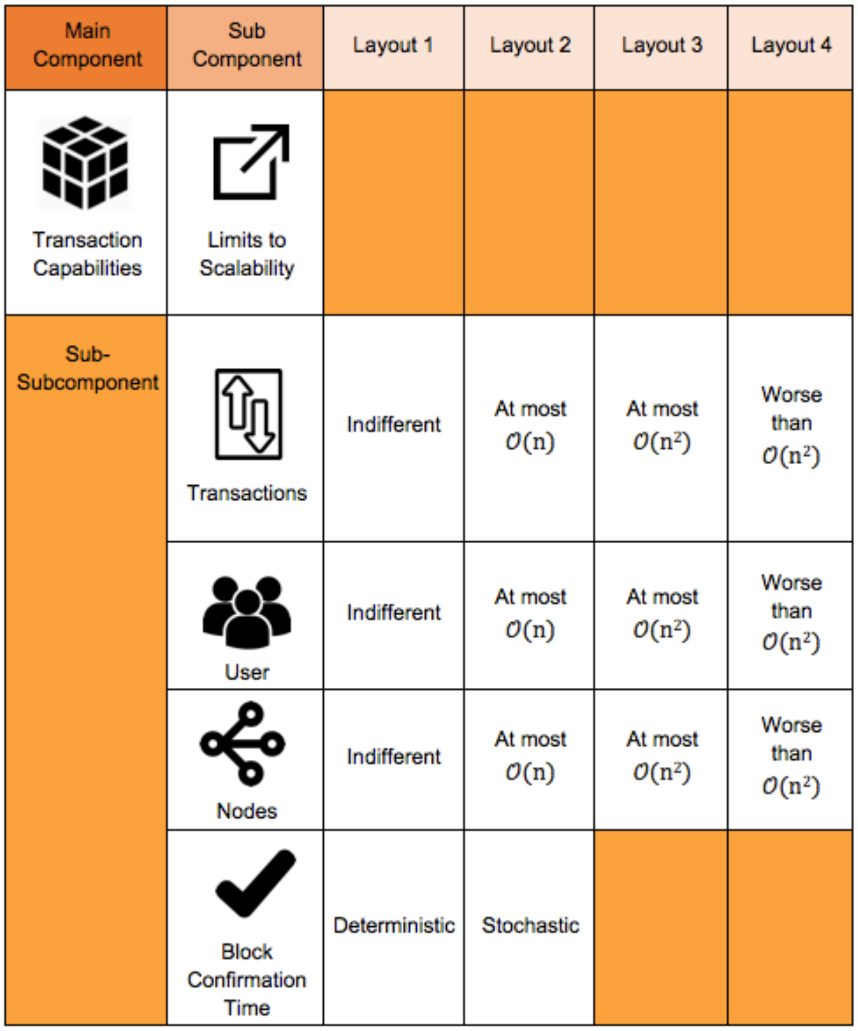
2. **Limit by number of users.** Bitcoin only stores transactions in its public ledger. This is different from other related technologies. Ripple, on the other hand, stores not only transactions, but also the state of the Ripple accounts. Therefore, in scenarios like this, it is the number of users of the system what limits its scalability. A similar problem occurs in Ethereum: Eventually the system will be constrained by the number of DAO, individu-

als, etc. that it will contain - as these are the actors that generate activity in the system - .Therefore, the term "Number of Users" for this layout is a broad reference to the number of objects of which its state is stored. Needless to say, this layout is somehow related to the previous, the number of transactions will depend on the number of users. However, one limiting factor still can appear regardless of the other.

3. **Limit by number of nodes.** The number of nodes connected to the network, acting as verifiers for the information that is stored on the blockchain, presupposes a limiting factor because of the mechanism of information diffusion adopted. *Gossiping* is a process that requires larger times in decentralised networks to propagate into a consensus state [30], and may even reach a point - where the relative time taken by network traffic is very long - where consensus cannot longer be reached and the blockchain naturally forks. therefore this process naturally limits the applicability of fully decentralised solutions.

   Possible values. These three layouts can have three different values, regarding on how detrimental is a specific layout to the overall performance of the system. The possible values that each layout can have are divided into four: (i) **Indifferent**, (ii) **At most linear**, (iii) **At most quadratic**, (iv) **Worse than quadratic**. The first value is assigned when the relevant global characteristics of a system is independent of the number of specific class; the next three, express three categorical values that are assigned to the dependency of the number of elements in said class. For example, the number of users is largely irrelevant to the performance of the Bitcoin network (because this number is never translated into any property of the network). However, the number of transactions increases linearly a penalty on the local network traffic.

4. **Confirmation time.** The time it takes a specific action to be confirmed ultimately depends on the time it takes for it to be added to the blockchain, and to be validated to further blocks later appended to it. Different approaches can be taken to this process: **deterministic** addition of new blocks at regular intervals (taken by Peercoin) and **stochastic** addition like in Bitcoin, where the process of mining induces an Exponential distribution of inter-block discovery time.



Main Component	Sub Component	Layout 1	Layout 2	Layout 3	Layout 4
Transaction Capabilities	Limits to Scalability				
Sub-Subcomponent	Transactions	Indifferent	At most $O(n)$	At most $O(n^2)$	Worse than $O(n^2)$
	User	Indifferent	At most $O(n)$	At most $O(n^2)$	Worse than $O(n^2)$
	Nodes	Indifferent	At most $O(n)$	At most $O(n^2)$	Worse than $O(n^2)$
	Block Confirmation Time	Deterministic	Stochastic		

FIGURE 11: Limits to scalability- Ontology Matrix

6 NATIVE CURRENCY/TOKENISATION

The aforementioned incentive scheme is to be provided in a token, whose value is assigned precisely because of the cost associated with its production. Initially, solutions like Bitcoin have created its own (and single) asset class (the bitcoin) that can be transacted within the system. This single solution is not the only possible one. Further, the [31] native currency possibilities and tokenisation enable different use cases of the blockchain technology like asset-transfers via tokens, exchanges, etc. Moreover, it is of uttermost importance (in the way the users perceive incentives to participate in the validation process) how these assets are supplied into the system.

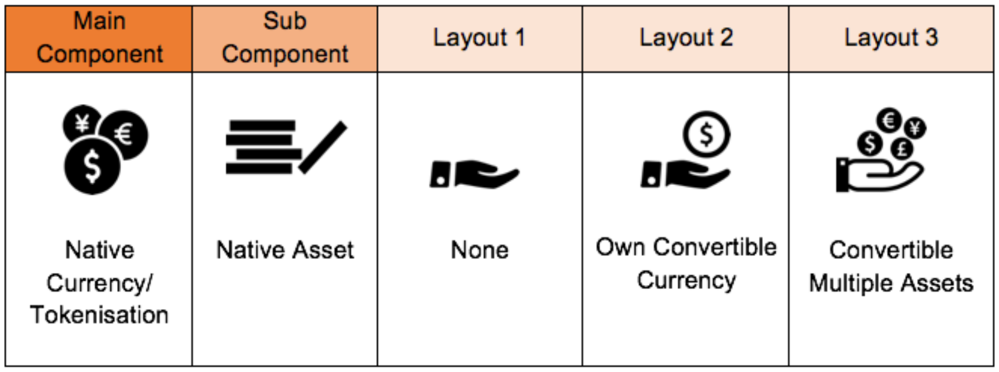
6.1 NATIVE ASSET

Blockchain technologies have underlying native asset (which is also called currency) which are tokens that allow to run the daily activities on the platforms or communities. According to [32] convertibility of digital currencies increases the network participation.

LAYOUTS

We identify three possible layouts for *Native asset*:

1. **None.** Private blockchain implementations do not require a native asset within to incentivise participation. In these cases, there is no native asset incorporated into the system
2. **Own Convertible Currency.** The pioneer in digital currency only uses its own underlying currency. Bitcoin or Ethereum are examples of technologies with single asset compatibility [33]. These technologies are limited to their own underlying digital currency, but it can also have off-chain solutions to interoperate with other currencies to execute transactions or to enroll into smart contracts.
3. **Convertible Multiple Assets.** Other technologies like Counterparty, Ardor or Ripple do have their own underlying currencies or tokens to execute tasks. However, these technologies also enable the possibility of exchange of assets expressed in others outside the native to the platform. This approach of multiple, convertible, currencies have the advantage of allowing for exchange markets be directly reflected into the system.



Main Component	Sub Component	Layout 1	Layout 2	Layout 3
Native Currency/ Tokenisation	Native Asset	None	Own Convertible Currency	Convertible Multiple Assets

FIGURE 12: Native Asset - Ontology Matrix

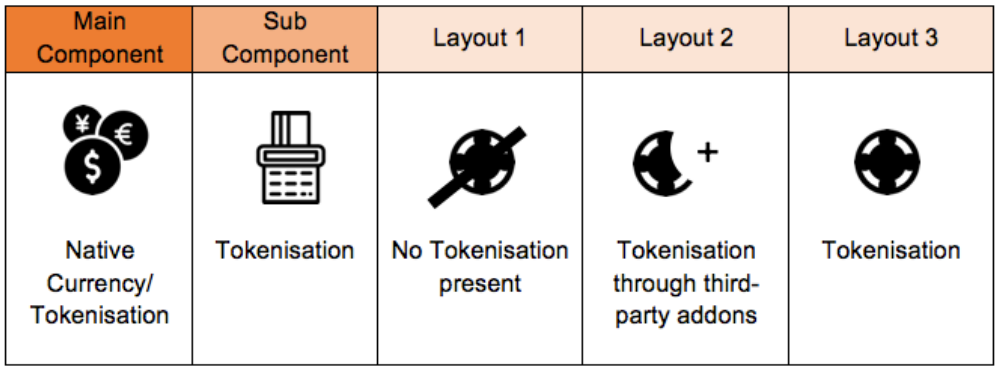
6.2 TOKENISATION

A token acts as a digital bearer bond, whose ownership is determined by the data embedded in the blockchain. Ownership of the tokens is transferable between holders using other transactions with associated “transfer” metadata. This does not require the approval of any other authority. The possibility of tokenisation enables a range of possible use cases for the blockchain technologies outside the purely financial world[31].

LAYOUTS

We identify three possible layouts for *Tokenisation*:

1. **No tokenisation present.** Without third-party technologies, Bitcoin does not have implemented technologies that enable tokenisation.
2. **Tokenisation through third-party addons.** Bitcoin plus ColorCoin enables the existence of tokenised transactions in the Bitcoin blockchain. Such solution is based on the cryptographic nature of Bitcoin addresses and the script language.
3. **Tokenisation.** The tokenisation possibilities together with the extensions of metadata are available in several implementations and constitute the backbone of blockchain-based property registries.



Main Component	Sub Component	Layout 1	Layout 2	Layout 3
Native Currency/ Tokenisation	Tokenisation	No Tokenisation present	Tokenisation through third-party addons	Tokenisation

FIGURE 13: Tokenisation-Ontology Matrix

6.3 ASSET SUPPLY MANAGEMENT

The process of the digital asset (usually referred to as *currency*) creation varies across different blockchain technologies. Each approach has taken different economic frameworks in most cases fixing a specific monetary policy the future of a particular system. This is also a pillar of the incentive scheme that users have to participate (or not) in the validation process [30].

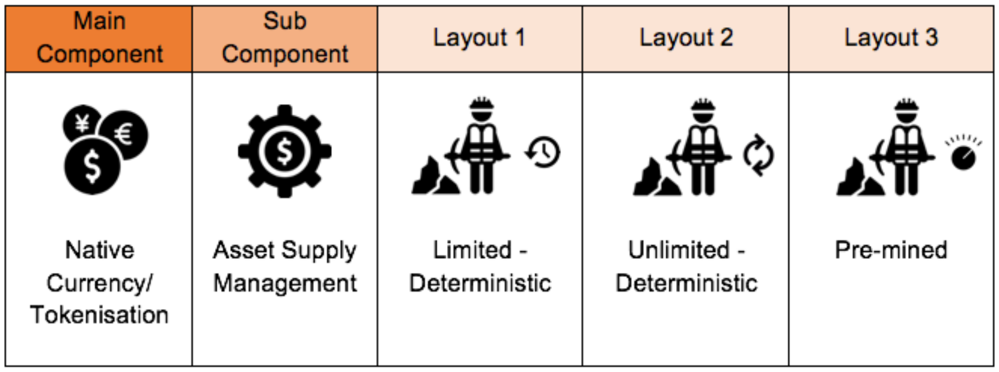
LAYOUTS

We identify tree possible layouts for *Asset Supply Management*:

1. **Limited - Deterministic.** The most replicated system in the world of blockchain is the limited supply as introduced in Bitcoin. Not only the supply grows sub-linearly over long periods of time (in contrast to what occurs in normal fiat currencies), but it is designed to have a well defined limit. It is important that, while this incentivises users to adopt the technology and contribute to the process of verification - for which they get a retribution -, on the other hand, it also creates an incentive to hoard the asset, limiting transactions.

2. **Unlimited - Deterministic.** Very few (eventually not broadly adopted) digital currencies based on blockchain attempted to create unlimited supply, like Dogecoin or Freicoin.

3. **Pre-mined** Some altcoins (with the purpose of funding the development of the platform, or with the sole idea of profiting) have distributed all the assets before the starting of the system. Then, a reward system induces some kind of redistribution.



Main Component	Sub Component	Layout 1	Layout 2	Layout 3
Native Currency/ Tokenisation	Asset Supply Management	Limited - Deterministic	Unlimited - Deterministic	Pre-mined

FIGURE 14: Asset Supply Management - Ontology Matrix

7 EXTENSIBILITY

The alignment of the interoperability, intraoperability,governance and script language determine the future ecosystem of the blockchain network and the integration possibilities of variety of blockchain technology or blockchain related technology. *Extensibility* includes the following subcomponents:

1. *Interoperability*
2. *Intraoperability*
3. *Governance*
4. *Script Language*

7.1 INTEROPERABILITY

Interoperability illustrates the overall capability of blockchains to exchange information with other systems, outside of blockchains. It allows inflow, outflow and information retrieval of data providers that are not necessarily a blockchain-based system, e.g. financial data providers.

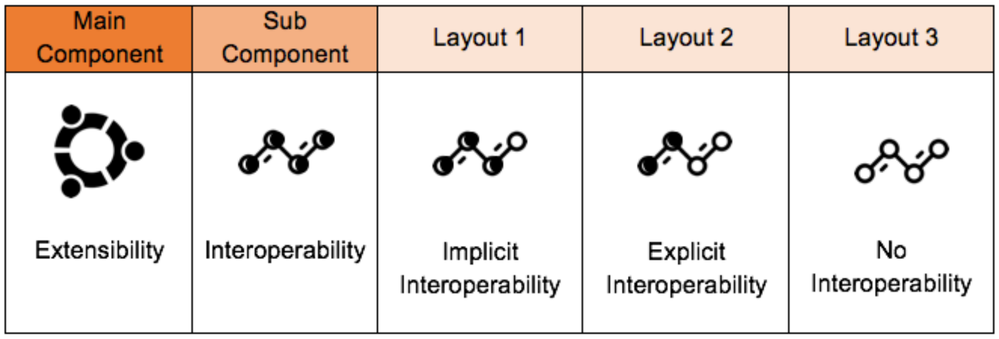
LAYOUTS

We identify three possible layouts for *Interoperability*:

1. **Implicit interoperability.** It occurs when the smart contracts that specify conditions under which a particular transaction (or event) is to take place can be written in a Turing-complete blockchain script language. In this context, implicitly any kind of condition can be specified, even those involving specific status in other systems. This implies an (albeit cumbersome) way of interaction from a blockchain solution to any API tool or interface.

2. **Explicit interoperability.** If the script language is not Turing complete or the system has specific tools implemented that enable interoperability with the real world (like Bitcoin with Counterparty), then we talk about explicit interoperability, as it is brought purportedly into the system and one of its design principles.

3. **No Interoperability.** A blockchain without any kind of possibility to interact with other systems. As implemented, Bitcoin in absence of external solutions (i.e. off the chain layers) has no interoperability implemented. It applies to most existing blockchain-based systems whose script language is not Turing complete.



Main Component	Sub Component	Layout 1	Layout 2	Layout 3
Extensibility	Interoperability	Implicit Interoperability	Explicit Interoperability	No Interoperability

FIGURE 15: Interoperability - Ontology Matrix

7.2 INTRAOPERABILITY

Intraoperability illustrates the overall capability of blockchains to exchange information with other blockchains. It allows inflow, outflow and exchange of data between different blockchains.

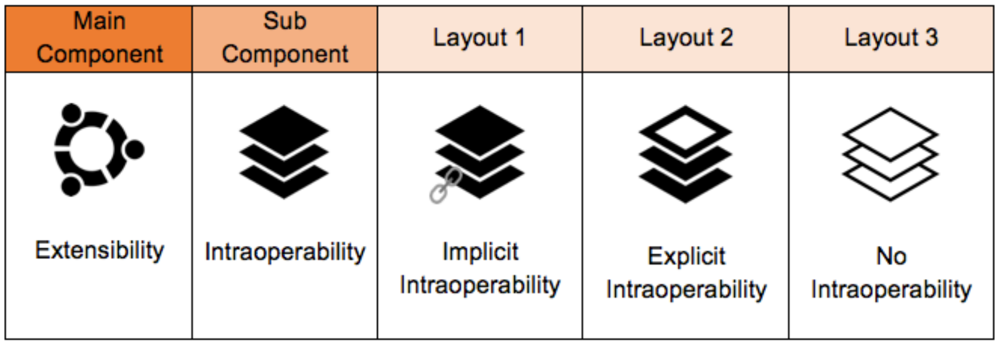
LAYOUTS

We identify three possible layouts for *Intraoperability*:

1. **Implicit intraoperability** It occurs when the smart contracts that specify conditions under which a particular transaction (or event) is to take place can be written in a Turing-complete blockchain script language. In this context, implicitly any kind of condition can be specified, even those involving specific status in other blockchains.

2. **Explicit intraoperability** If the script language is not Turing complete but is specifically designed to allow for intraoperability, then we talk about explicit intraoperability, because it is brought purportedly into the blockchain and it is one of its design principles. An example of this is Bitcoin with Counterparty.

3. **No intraoperability** A blockchain without any kind of possibility to interact with other blockchains. As implemented, Bitcoin in absence of external solutions has no intraoperability implemented. Solutions for non intraoperable blockchains resort on: 1) Trusted proxies to connect blockchains; 2) Pegged blockchain systems; 3) Distinguishing tokens in the same blockchain based system.



Main Component	Sub Component	Layout 1	Layout 2	Layout 3
Extensibility	Intraoperability	Implicit Intraoperability	Explicit Intraoperability	No Intraoperability

FIGURE 16: Intraoperability - Ontology Matrix

7.3 GOVERNANCE

Effective governance rules are crucial for the successful implementation of the blockchains and for their capability to adapt, change and interact. As the blockchain deployment structures (public chain, private chain, consortium chain) are different, their management patterns are also quite different. We identify two type of governance rules: 1) *technical rules* of self-governance defined by the participants. Technical rules are composed of software, protocols, procedures, algorithms, supporting facilities and other technical elements; 2) *regulatory rules* defined by external regulatory bodies composed of regulatory frameworks, provisions, industry policies and other components. Regulatory rules are by definition not technical in nature and therefore outside the scope of this ontology. We focus instead on techical rules which are particularly interesting for their feedback loop with the proposed technological solutions.

LAYOUTS

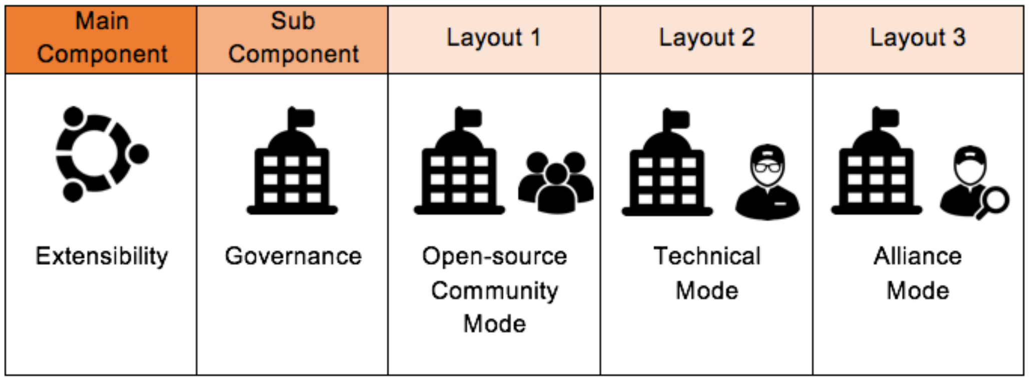
We identify three possible layouts *Governance*:

1. **Open-source Community Mode**. In this case, open communities of developers (following open-source principles) and validators (very often in coordination with the blockchain foundation) coordinate upgrades and technical adjustments of the blockchain. For example, Bitcoin is mainly maintained by a team of core developers who in coordination with miners agree on changing parameters or other settings of the Bitcoin network. Also Ethereum and Hyperledger (backed up by the Linux Foundation) follow an open-source community model.

2. **Technical Mode**. Since the blockchain technology is very versatile and can be applied to many business cases, enterprises with a strong technical strength (e.g., IBM and Mi-

crosoft) have proposed themselves as technical solution providers for blockchain architectures (proprietary hardware and software systems and basic services). In these cases, the technical rules of blockchain governance are dictated by the companies according to their business goals. For example, in 2015 Microsoft collaborated with ConsenSys to create the Ethereum blockchain technology service and took it as part of the Microsoft Azure service (EBaaS) to provide distributed ledger technology trials for enterprise customers, partners and developers. Moreover, in order to protect their proprietary blockchain architectures, these companies generally apply also for patents. According to [2] 2,500 patents on this topic have been filed from 2014 to 2016.

3. **Alliance Mode**. This is the blockchain governance model proposed by industry consortia (e.g., B3i, R3) composed of companies with common business or technological progress demands. The alliance mode has the scope to sharing technology platforms to build common business models and standards. Only companies that meet certain criteria (e.g., payment of the fees, qualification of the organisation) are legitimised to collaborate to set technical rules of blockchain governance. Those companies join together to promote commercial and technological progress in the area of blockchain under mutual benefit and common contribution.



Main Component	Sub Component	Layout 1	Layout 2	Layout 3
Extensibility	Governance	Open-source Community Mode	Technical Mode	Alliance Mode

FIGURE 17: Governance - Ontology Matrix

7.4 SCRIPT LANGUAGE

Widespread programming languages are Turing-complete, which in formal terms refers to the fact that it is possible to implement an algorithm on it to simulate any Turing machine. These are therefore general purpose languages, in which arbitrary computations can be performed. Languages that are not of this kind, are so because of design reasons.

Blockchain systems allow to modify the conditions under which certain information (e.g. transactions) will be included into the public record. These conditions must be specified in an algorithmic manner, and in some contexts are termed *smart contracts*. These algorithms are elicited in a *scripting language* designed purely for this purpose. Therefore the intended flexibility given to the users with respect to the scope that the algorithm can develop affect tremendously the degree of freedom to create conditions for some actions to occur (on the one hand) and the hypothetical computational effort that may be necessary to assess if a particular condition is fulfilled or not.
It is worth remarking here that how limited is the scope of the scripting language is another design decision developers must carefully choose before the implementation of the blockchain, as abrupt changes (or bugs) may deride the logic of particular transactions.

LAYOUTS
We identify four possible layouts for *Script Language*:

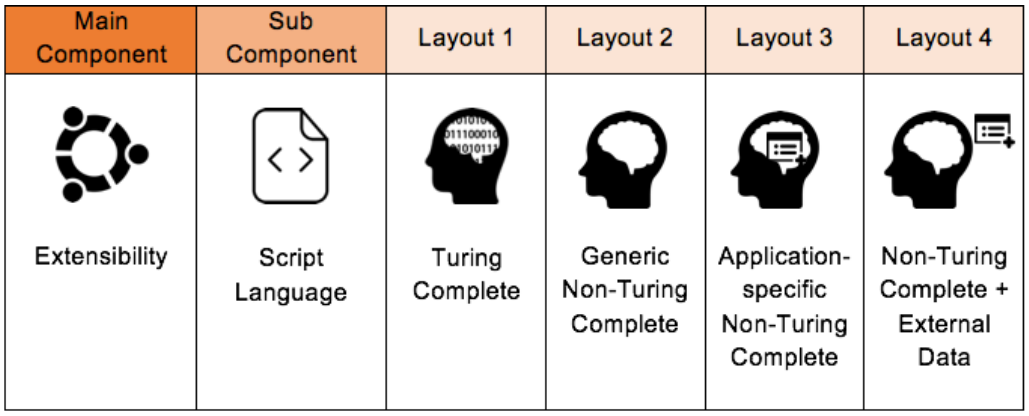
1. **Turing Complete.** Ethereum refers to a suite of protocols that define a platform for decentralised applications. With respect to scripting languages, on the one end of the spectrum, the Ethereum Virtual Machine (EVM) can execute code of arbitrary algorithmic complexity. In the terms described above, Ethereum is "Turing complete", because developers can create applications, which runs on the EVM.Furthermore, Counterparty also uses Ethereum's entire smart contract platform to enable users to write Turing complete smart contracts. It has been pointed out that there exist scalability and security concerns regarding the usage of Turing-complete for scripting languages in blockchain systems. As of this writing, these have not been resolved.

2. **Generic Non-Turing Complete**. When designing Bitcoin, a decision was made to keep the scripting language limited in scope, to allow for a low impact of these calculations in the efficiency of the system. It is therefore a non-Turing complete language, and most blockchain implementations have followed this path. There is no connectivity in these to so-called "oracles" that allow obtaining data from sources that gather data that is exogenous to the blockchain.

3. **Application-specific Non-Turing Complete**. There some non-Turing complete languages that are more expressive than the generic ones and purposely designed for certain cases. By restricting the language to only be able to write programs relevant to specific limited cases, the potential outputs of those programs becomes predictable. This allows those

outputs to be queried and easily analysed. One example is Digital Asset Modelling Language (DAML) which is designed to codify only financial rights and obligations for execution in private networks. DAML is also more expressive than Bitcoin' script language and easier to read from a non techical audience.

4. **Non-Turing Complete + External Data.** There exists a third category barely used so far that (while keeping the nature of the scripting language non-Turing complete) allows for existence of oracles. These oracles are considered trustful sources and add a layer of simplification on the validation to be performed by the language, empowering above Turing-completeness (as long as the oracles are reliable). This layout is then "non-Turing Complete + External Data".



Main Component	Sub Component	Layout 1	Layout 2	Layout 3	Layout 4
Extensibility	Script Language	Turing Complete	Generic Non-Turing Complete	Application-specific Non-Turing Complete	Non-Turing Complete + External Data

FIGURE 18: Script Language - Ontology Matrix

8 SECURITY AND PRIVACY

First and foremost to preserve the consistency of the records, storing information into the blockchain must be computationally costly, but computationally cheap to validate the validity of such storage. This is obtained by means of cryptography. At the same time, this helps to meet the market standards of several industries e.g. in financial industries the transactions have to be secured by several security layers, such as a cryptography to have an immutable information flow, which is a mandatory requirement to secure those transactions. The component *Security & Privacy* includes the following subcomponents:

1. *Data Encryption*

2. *Data Privacy*

8.1 DATA ENCRYPTION

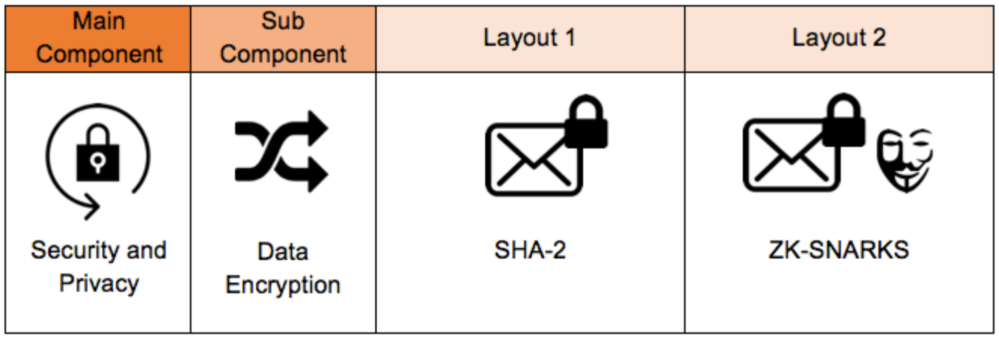
Which is the cryptographic toolset used to hash information and to validate such hashes. This makes the system makes systems more or less versatile depending on the intrinsic capabilities of the framework used. Importantly, in case of vulnerabilities (or workable attacks, like recently identified for the SHA-1 algorithm) the whole system under which a particular blockchain has been implemented becomes vulnerable, and may be targetted in multiple ways.

LAYOUTS

We identify two possible layouts for *Data Encryption*:

1. **SHA-2.** SHA stands for Secure Hash Algorithm. In its two incarnations, SHA-256 and SHA-512, SHA (originally developed by the National Security Agency, USA) is the most widely variants for hashing functions [34, 35] having first been used in Bitcoin. When issued to hash transactions, it requires a piece of information from the issuer, i.e. the public key for the validation to take place.

2. **ZK-SNARKS.** The Zero-Knowledge - Succinct Non-interactive Argument of Knowledge is a newer technology where no data whatsoever has to be provided to validate a specific hash [36]. With the hashed message and the encrypted one, is sufficient as a proof to generate the validation. This anonymises much more the individual information.



Main Component	Sub Component	Layout 1	Layout 2
Security and Privacy	Data Encryption	SHA-2	ZK-SNARKS

FIGURE 19: Data Encryption - Ontology Matrix

8.2 DATA PRIVACY

Although public/private key infrastructures and other measures like hashing functions should ensure that only the intended recipient can read the message and have access to the content of the transaction, the research shows that blockchain transactions (for e.g., in Bitcoin) can be linked together in order to extract additional information and eventually also the identity of the participants [37]. Indeed, there exists an inevitable tradeoff between a decentralised peer-validate system and the security and privacy of information. In this regard, several alternative solutions have been proposed to "encrypt" the data in such a way that even though computations and transactions occur in plain sight, the underlying information is completely kept obfuscated. Obfuscation is a way of turning any program into a "black box". This is equivalent to the original program: runs the same "internal logic" and provides the same outputs for the same inputs. But information on the data and processes is inaccessible. Of course there exist a strong interrelation between *Data Privacy* and *Data Encryption*.

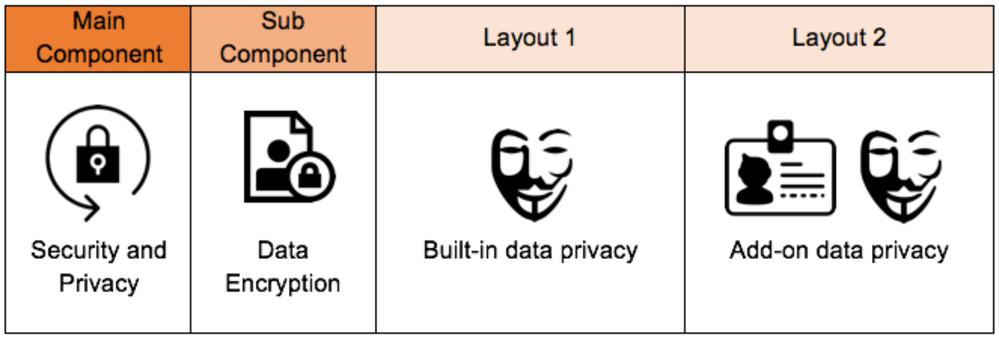
LAYOUTS

According to the solutions proposes so far to enhance *Data Privacy*, we identify two possible layouts:

1. **Built-in data privacy**. With built-in data privacy we include all those blockchains that by default provide obfuscation of information. For example Zcash uses built-in zero-knowledge cryptography to encrypts the payment information in the transactions. Although Zcash payments are published on a public blockchain, sender, recipient, and amount of a transaction remain private. Alternatively, blockchains like Enigma (a project that seeks to implement the secret sharing DAO concept) using built-in secure multi-party computation guaranteed by a verifiable secret-sharing scheme. In this case, the data can be split among N parties in such a way that $M < N$ are needed to cooperate in order to either complete the computation or reveal any internal data in the program or the state. But $M - 1$ parties cannot recover any information at all (which implies the need of trust on the majority of the participants to be honest). Finally, CORDA by R3 proposes a Node to Node (N2N) system characterised by encrypted transactions where only the parties involved in the transaction have access to the data. This is suitable for financial transactions where a high degree of confidentiality is required. Third parties like central banks or other market authorithies may have access to the data by invitation only.

2. **Add-on data privacy**. In this case, pseudonymous or public blockchains must resort on external solutions in order to obfuscate the information. One method is the *mixing* service like Coinjoin. The principle behind this method is quite simple: several transactions are grouped together so to become a unique M-to-N transaction. If for example, Alice wants to send one coin to Bob, and Carla wants to send one coin to David, a mixing transaction could be established whereby the addresses of Alice and Carla are both listed as in-

puts, and the addresses of Bob and David are listed as outputs in one unique transaction. Thus, when inspecting the 2-to-2 transaction from outside it is impossible to discern who is the sender and who the recipient. Alternative to the *mixing* service, the *secret sharing* allows data to be stored in a decentralised way across N parties such that any K parties can work together to reconstruct the data, but K-1 parties cannot recover any information at all. Alternative add-on data privacy tools are *ring signatures* and *stealth addesses* which hide the recipient of a transaction and can be used by any blockchain. Ring signatures and its variant (linkable ring signatures) allow to hide transactions within a set of others' transactions. In this case the transaction is tied to multiple senders' private keys but only one of them is the initiator. Thus, the verifier may only identify that one of them was a signer, but not who exactly that was. In the case of stealth addresses, a receiver generates a new dedicated address and a "secret key" and then sends this address to someone who he wants payment from. The sender use the address generated by the receiver plus a "nonce" (one time random number) in order to generate the address he/she will send funds to. The sender communicates the nonce to the receiver who by using the nonce and the secret key generated earlier he/she can unlock the address.



Main Component	Sub Component	Layout 1	Layout 2
Security and Privacy	Data Encryption	Built-in data privacy	Add-on data privacy

FIGURE 20: Data Privacy - Ontology Matrix

9 CODEBASE

The codebase of the blockchain technologies delivers information about which challenges a developer could face and what kind of changes the underlying programming language. Therefore the main component 'Codebase' is essential to align and increase the efficiency of blockchain related IT architectures. *Codebase* includes the following subcomponents:

1. *Coding Language*

2. *Code License*

3. *Software Architecture*

9.1 CODING LANGUAGE

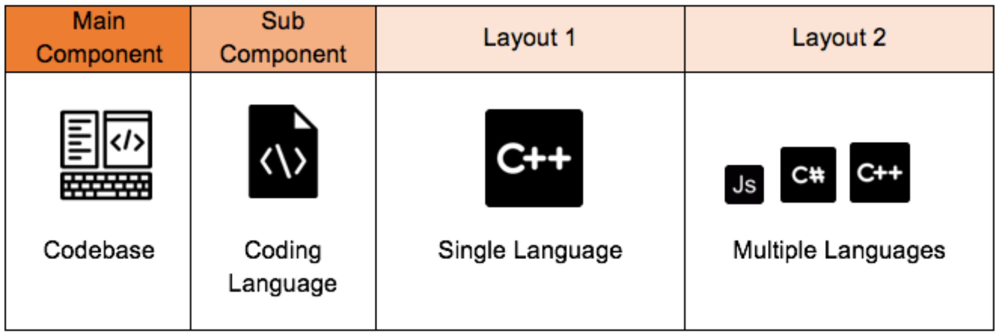
Coding language illustrates the interconnectivity of programming languages of the blockchain technologies.

LAYOUTS

We identify two possible layouts for *Coding Language*:

1. **Single Language.** Bitcoin has released The Bitcoin Core version 0.13.1 with the underlying coding language C++. As Bitcoin is open source, there have been (much less popular than the original codebase) implementations in different languages (like Java).

2. **Multiple Languages.** Ethereum uses C++, Ethereum Virtual Machine Language and Go, which enables more interaction with other languages. Stellar.org maintains JavaScript, Java, and Go-based SDKs for communicating with Horizon. There are also community-maintained SDKs for Ruby, Python, and C-Sharp.



Main Component	Sub Component	Layout 1	Layout 2
Codebase	Coding Language	C++ Single Language	Js C# C++ Multiple Languages

FIGURE 21: Coding Language - Ontology Matrix

9.2 CODE LICENCE

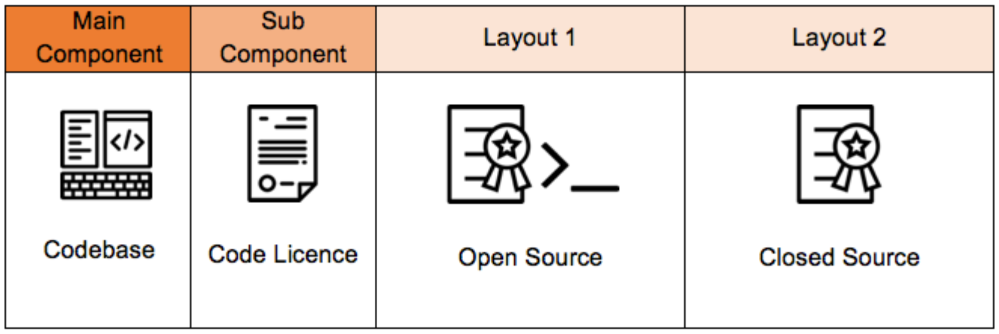
The Code License illustrates the possibility of changes to the source code of the underlying technology.

LAYOUTS

We identify thee possible layouts for *Code Licence*:

1. **Open Source.** Regardless of the exact licence used for specific projects, we refer only to the openness in the source code as the only differentiating factor. Bitcoin core developers have continuously licenced the source code under the MIT licence. Counter-intuitively, a permissive licence like the MIT one (in which other developers can take the source code and fork it) eventually prevents multiple implementations. It also allows for continued development, larger code growth and allows adoption at a faster pace. Furthermore, Ripple and Stellar have licensed their codes with the ISC License. The ISC license is another permissive licence.

2. **Closed Source.** For private implementations of blockchain-based systems, the source code is not necessarily openly distributed. In this case, and risking the existence of unadressed bugs or unreported characteristics that may violate the expected conditions of use and functioning, the code may be kept outside of reach for users.



Main Component	Sub Component	Layout 1	Layout 2
Codebase	Code Licence	Open Source	Closed Source

FIGURE 22: Code Licence - Ontology Matrix

9.3 SOFTWARE ARCHITECTURE

The *Software Architecture* refers to the high level structures of the blockchain system. Each structure comprises software elements, relations between them, and the properties that elements and relations gave. The choice of the software architecture is very important in order to better manage changes once implemented. Software architecture choices include specific structural options among the possibilities that are available for software design.

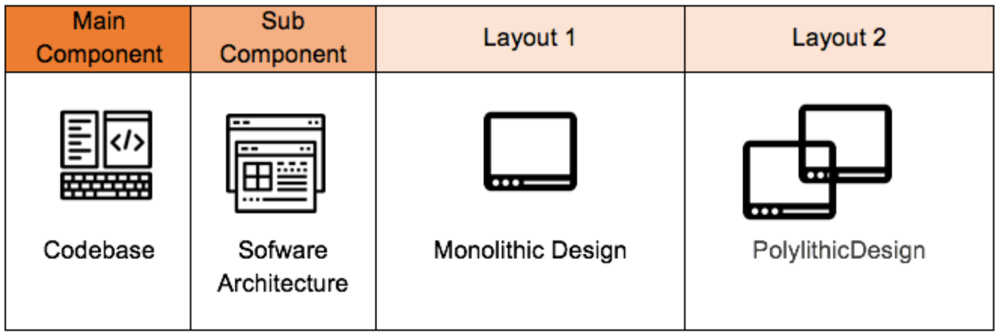
LAYOUTS

We identify two possible layouts for *Software Architecture*:

1. **Monolithic Design**. In this case, all the aspects of a decentralised ledger (P2P connectivity, the "mempool" broadcasting of transactions, criterion for consensus on the most

recent block, account balances, nature of smart contracts, user-level permissions, etc.) are handled by a blockchain built as a single-tier software application without modularity. Examples of blockchains with monolithic design include Bitcoin and Ethereum. These architectures suffer from lack of extensibility on the long run. F

2. **Polylithic Design**. The Polylithic approach decouples the consensus engine and P2P layers from the details of the application state of the particular blockchain application. For example, in Tendermint the blockchain design is decomposed. It offers a very simple API between the application process and its application-agnostic ”consensus engine” (TenderminCore) which enables to run Byzantine fault tolerant applications, written in any programming language, not just the one the consensus engine is written in. Also Hyperledger follows a polylithic design as it is composed of interchangeable modules representing different components of blockchain technology.



Main Component	Sub Component	Layout 1	Layout 2
Codebase	Sofware Architecture	Monolithic Design	PolylithicDesign

FIGURE 23: Software Architecture - Ontology Matrix

10 IDENTITY MANAGEMENT

The main component *Identity Management* ensures secure access to sensitive data and to establish a suitable governance model for the blockchain. Subcomponents:

1. *Access and Control Layer*

2. *Identity Layer*

10.1 ACCESS AND CONTROL LAYER

When establishing the right governance structure for a blockchain it is important to consider the ledger construct. Depending on its purpose, the ledger could be run by a central authority and governed by it or it could be run in a decentralised fashion according to a set of governance rules adhered to and enforced by participants on the blockchain network. The governance structure determines the authorisation and the control policy management functions. Those rules provide permission for users to access to or use blockchain resources. Those are a set of rules that manage user, system and node permissions that must be followed in security-related activities. Blockchains may have different permissions according to which access and control to data is allowed. The distinguishing features must answers to the following questions:

- Which users have “read” access?
- Which users have “write” access?
- Is it there anyone who can “manage consensus” (i.e., update and maintain the integrity of the ledger)?

According to the set of governance rules, we may have different system designs that reply to the above questions in a different ways in order to better serve either a public or a private interest with either a *general* (like in the case of Ethereum) or a *special* (like in the case of Corda) purpose. On one side, private blockchains are generally those with a set of constrained “read/write” access alongside a consensus algorithm which allows only a pre-selected group of people to contribute and maintain the blockchain integrity. Instead, public blockchains do not control “read/write” access or in the consensus algorithm for any given set of participants. Nevertheless, this does not mean that certain permission structures can not be implemented as part of a specific application.

LAYOUTS

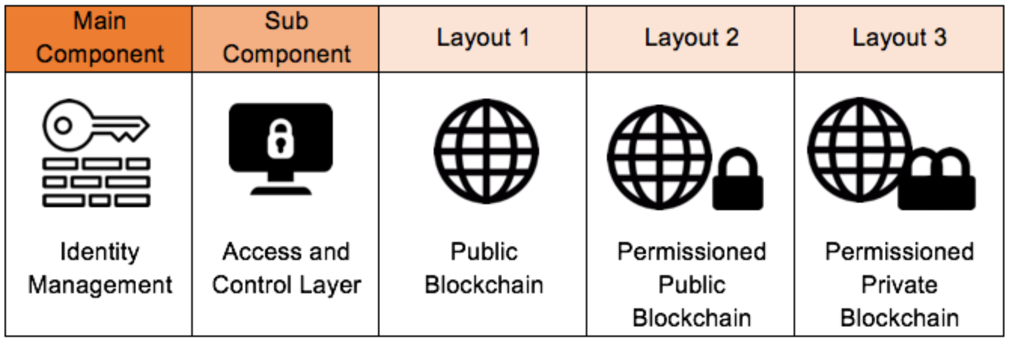
Although different variations are possible, the authority to perform transactions on a blockchain generally belong to one of the following main models of *Access and control Layer*:

1. **Public blockchain**. In this case, there is no preference in access or in managing consensus. All participants (nodes), have ”read/write” access and without any control can contribute to the update and management of the ledger. Examples of blockchains in this area include Bitcoin where every participant can either choose just to use the blockchain to exchange Bitcoins (or other data on the top of it, in general by means of third-party technologies), run a full node or even become a miner to participate in the process of transaction validation.

2. **Permissioned Public blockchain**. In this case “read” access is enabled for all users, however “write” access and/or “consensus management” require permission by a pre-selected

set of nodes. Ripple belongs to this group as to validate transactions a participant need to be part of the so-called *Unique Node List*. Some other examples include Ethereum and Hyperledger Fabric which is used for the exchange of tangible (real estate and hardware) to the intangible (contracts and intellectual property) assets between interprises.

3. **Permissioned Private blockchain**. In this case, ”read/write” and ”consensus management” rights can only be granted by a centralised organisation. An example is Monax (formerly known as Eris).



Main Component	Sub Component	Layout 1	Layout 2	Layout 3
Identity Management	Access and Control Layer	Public Blockchain	Permissioned Public Blockchain	Permissioned Private Blockchain

FIGURE 24: Access and Control Layer - Ontology Matrix

10.2 IDENTITY LAYER

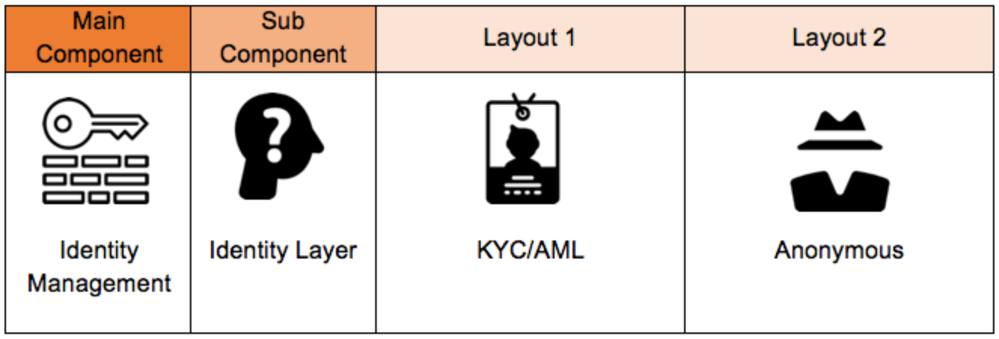
The onboarding and offboarding of nodes to the blockchain networks is handled differently by the various software solutions. AML and KYC procedures – generally required to proceed personal related data e.g. medical data, bank information or other personal related data –, are the key aspects to consider when looking at the *Identity Layer*.

LAYOUTS

We identify two possible layouts:

1. **KYC/AML.** Stellar set requirements for all integrators to implement Know-Your-Customer (KYC)/Anti-Money-Laundering (AML) identity verification process to increase the transparency of the stellar network participants. Furthermore, Ripple forces their financial services partners to implement an identity layer to verify the user information.The financial services partners have to do a due diligence, depending on the requirements they must fulfill.

2. **Anonymous.** Bitcoin has no identity layer to identify the users. Those circumstances benefit misuse of bitcoins and money laundering activities through this blockchain network.



Main Component	Sub Component	Layout 1	Layout 2
Identity Management	Identity Layer	KYC/AML	Anonymous

FIGURE 25: Identity Layer - Ontology Matrix

11 CHARGING AND REWARDING SYSTEM

Blockchain systems incur in operational and maintenance costs that are generally absorbed by the participants to the network. Different kind of cost models are applied according to: 1) the architectural configuration design; 2) the governance system; 3) the data structure and the computation required on-chain. One of the cost items which is common to the wide majority of the blockchains is the verification cost. This is required to sustain the validation process of the transactions that compete to be appended (and never removed) to the ledger. The potential financial costs incurred when taking part of a blockchain platform, require an incentive scheme that maintains consistency of the cost structure across the different stakeholders. In this respect, we identify two subcomponents of *Charging and Rewarding System*:

1. *Reward System*

2. *Fee System*

11.1 REWARD SYSTEM

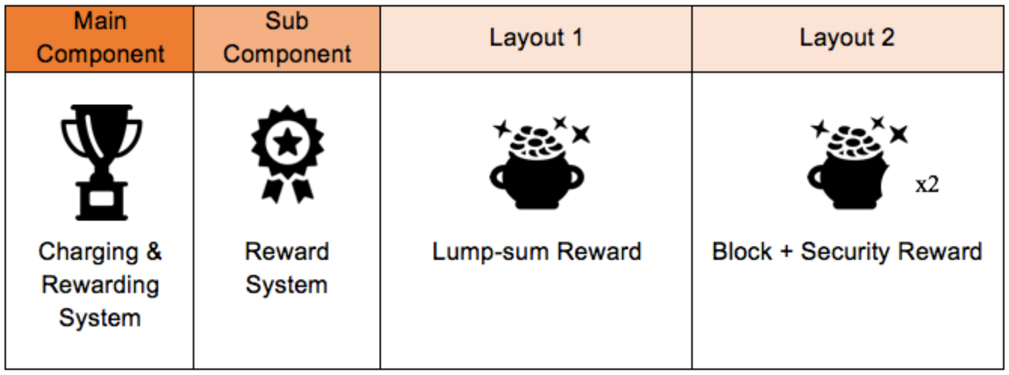
This subcomponent illustrates the rewarding mechanisms automatically put in place and trigerred by the blockchain systems in order to compensate active members contributing to data storage or transaction validation and verification.

LAYOUTS

We identify two possible layouts for *Reward System*:

1. **Lump-sum Reward.** Individuals taking part of the storage, validation or verification process (e.g., in the Bitcoin only verification is rewarded to users called *miners*) may be rewarded for their action. For example, in Bitcoin, the first transaction in each block is called *coinbase*, and the recipient is the user (or users) who created the block, that in this regard is a set of transactions verified by said user(s). The lump-sum reward can be fixed like in Enigma or variable like in Bitcoin.

2. **Block + Security Reward.** In other blockchain-based technologies, like Ethereum, blockchain the rewarding system includes, besides the block reward, a reward for including in the validation forked blocks that are otherwise valid. The design idea is to incentivise cross-validation of transactions (crucial in a setting where validation can be arbitrarily costly).



Main Component	Sub Component	Layout 1	Layout 2
Charging & Rewarding System	Reward System	Lump-sum Reward	x2 Block + Security Reward

FIGURE 26: System Rewards - Ontology Matrix

11.2 FEE SYSTEM

Other kind of rewards are those provided directly by the users to other participants of the system when launching any request in the network for storage, data retrieval, or computation and validation. With regards to this, we identify two sub-subcomponents: *Fees Reward* and *Fee Structure*.

11.2.1 FEE REWARD

Fees Reward describes the nature of the fees that the users are required to contribute when using a blockchain.

LAYOUTS

We identify three possible layouts for *Fees Reward*:

1. **Optional Fees.** In Bitcoin and related technologies [38] users can optionally pay a voluntary fee for the validation process. This fee is optional, but it is assumed that the larger the fee is, the lower is the processing time it will take to be added to a block, as miners will be more incentivised to do so. Moreover, given that the coinbase reward halves approximately every four years, currently, the reference Bitcoin client refuses to relay transactions with zero or no fees.

2. **Mandatory Fees.** Some systems like Stellar force all users to include fees in any transaction added into the system.

3. **No Fees.** In comparison, the Hyperledger Project is a blockchain solution for businesses, which combines a permissioned network an identity layer without any transaction fees.

11.2.2 FEE STRUCTURE

When provided by the system, fees can follow either a fixed or a variable structure.

LAYOUTS

There are two alternative layouts for *Fees Structure*:

1. **Variable Fees.** In this case, the fee is somehow linked to the ”size” of the request. In Bitcoin, larger the transaction size, the higher will be the fee the user shall pay in order to compensate for taking up space inside the block. Miners usually include transactions with the highest fee/byte first. The user can decide how many Satoshis (0.00000001 Bitcoins) wants to pay per byte of transaction. For example, if the transaction is 1,000 bytes and the user pay a fee of 300,000 Satoshis, he/she will be in the 300 Satoshi/bytes section (300,000/1,000=300.00). At the time of writing, this implies that the transaction will be included in the next 2 block transactions (i.e., within 20 minutes). However, to avoid queuing, the user can increase the fee. The fastest and cheapest transaction fee is currently 360 Satoshis/byte. For an average transaction size of 226 bytes, a fee of 81,360 Satoshis is currently the cheaper fee in order to get the transaction included in the first available block without delays. Also other blockchains apply variable fees and follows similar rules than Bitcoin.

2. **Fixed Fees.** In this case, the fee is linked to the request, not to its ”size”. For example, in Enigma every request in the network for storage, data retrieval, or computation has a

fixed price, similar to the concept of Gas in Ethereum. However, since Enigma is a Turing-complete system, the fee can be different depending on the specific request. Another example of blockchain with a fixed transaction fee is Peercoin which required a fixed 0.01 PPC per kilobyte.

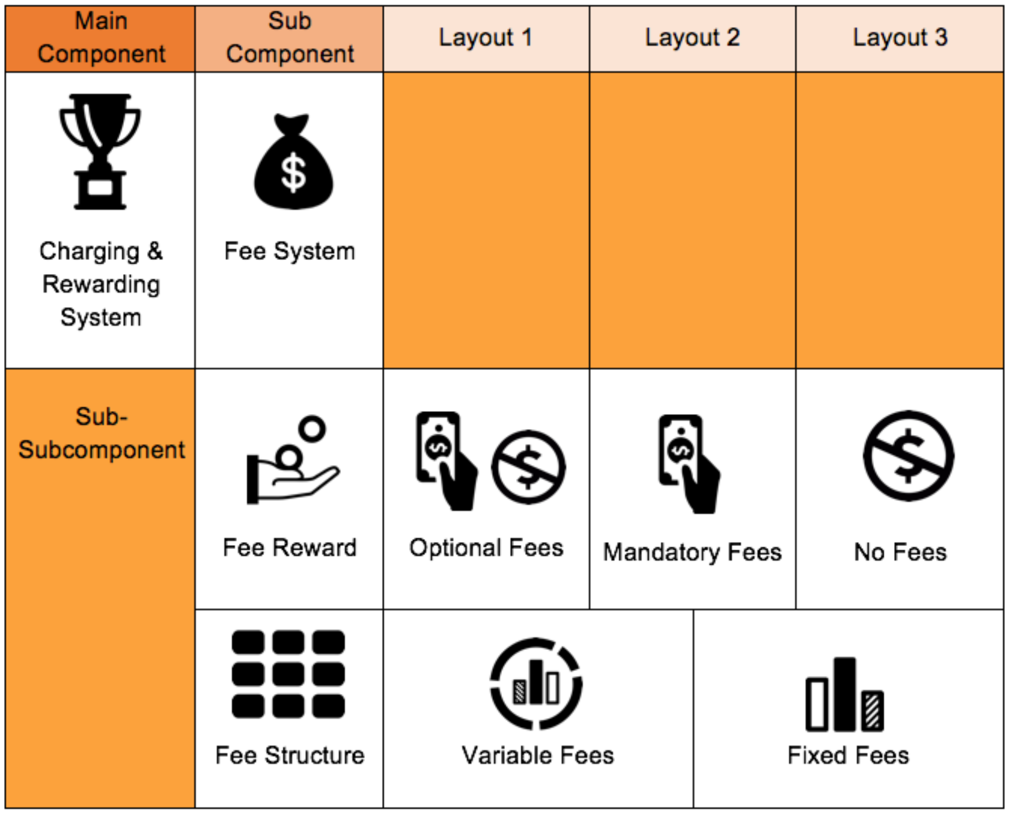


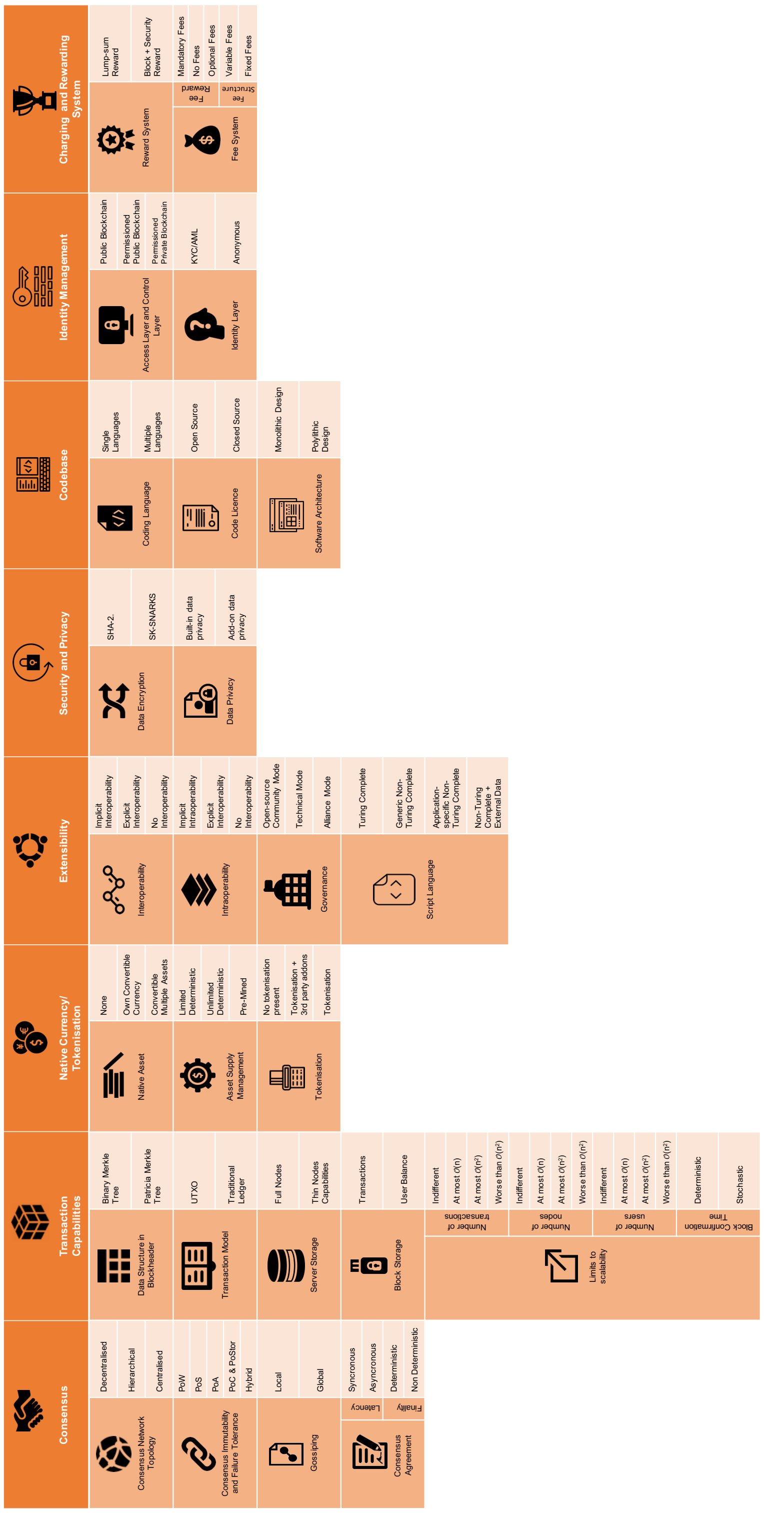
FIGURE 27: Fee System - Ontology Matrix

12 CONCLUSION

In the 21st century, the blockchain technologies will athwart affect all business areas: financial services, IoT, consumer electronics, insurances, energy industry, logistics, transportation, media, communications, entertainment, healthcare, automation, and robotics will be involved. After the advent of Internet, it currently represents the most prominent technology and it will shape the upcoming products and services in every industry field. Since the introduction of Bitcoin in 2009, the awareness of blockchain technologies has considerably increased. During the initial phase, the first mover regarding the adoption of blockchain was the financial industry. This is explained by the fact that blockchain enables cost reduction and increases the efficiency in several business processes (both internal and external) for financial institutions. An example of the big impact of blockchain in the financial industry regard the networks of global payments which involve money transactions in exchange of goods, services or legal obligations between both individuals or economic entities. Beyond payments, blockchain allows real-time settlements, which reduces operational costs for the banks. Furthermore, the immutability of the blockchain reduces the risk of fraud, as a consequence banks can use sophisticated smart contracts to capture digital obligations and to eliminate operational errors. Global payments are just a fraction of the overall use cases in the financial industry. Moreover, many other industries, including the public sector, are now looking at blockchain-enabled solutions for their own processes. This tremendous trend is causing a proliferation of multiple blockchain solutions which often are not interoperable and are built according to different engineering designs. Lately, software architectures, companies and regulators realised the need for standardisation of some of their components. This is becoming a necessary step for the blockchain in order to: 1) gain global adoption and compatibility, 2) create cross industry solutions, 3) provide cost-effective solutions. As always with standarisation processes, their creation must be a necessary equilibrium between different parties. But in this particular case, the open source community that brought forward this disruptive technology and which continuously develops most implementations should play a crucial role, as heralds of the advancement of blockchain. This work is an early stage comparative study of the existing blockchain technologies with the aim to propose a blockchain ontology: a reference architectural model for blockchains and their possible configurations. Based on component-based design, the blockchain ontology decomposes the blockchains into individual functional or logical components and identifies any possible different layout. The blockchain ontology proposes to assist in the exploration of design domains, in the implementation, deployment and performance measurement of different blockchain architectures. See Table 28 in Appendix for a schematic summary of the blockchain ontology matrix.

Acknowledgements: PT, TT and CJT thank Harvey R. Campbell for his invaluable comments on a previous version of this paper. PT acknowledges the University College London for financial support through the EPSRC program EP/P031730/1. CJT acknowledges the University of Zurich for financial support through the University Research Priority Programme on Social Networks.

A OVERALL MATRIX



Category	Subcategory		Options
Consensus	Consensus Network Topology		Decentralised; Hierarchical; Centralised
	Consensus Immutability and Failure Tolerance		PoW; PoS; PoA; PoC & PoStor; Hybrid
	Gossiping		Local; Global
	Consensus Agreement	Latency	Syncronous; Asyncronous
		Finality	Deterministic; Non Deterministic
Transaction Capabilities	Data Structure in Blockheader		Binary Merkle Tree; Patricia Merkle Tree
	Transaction Model		UTXO; Traditional Ledger
	Server Storage		Full Nodes; Thin Nodes Capabilities
	Block Storage		Transactions; User Balance
	Limits to scalability	Number of transactions	Indifferent; At most $O(n)$; At most $O(n^2)$; Worse than $O(n^2)$
		Number of nodes	Indifferent; At most $O(n)$; At most $O(n^2)$; Worse than $O(n^2)$
		Number of users	Indifferent; At most $O(n)$; At most $O(n^2)$; Worse than $O(n^2)$
		Block Confirmation Time	Deterministic; Stochastic
Native Currency/ Tokenisation	Native Asset		None; Own Convertible Currency; Convertible Multiple Assets
	Asset Supply Management		Limited Deterministic; Unlimited Deterministic; Pre-Mined
	Tokenisation		No tokenisation present; Tokenisation + 3rd party addons; Tokenisation
Extensibility	Interoperability		Implicit Interoperability; Explicit Interoperability; No Interoperability
	Intraoperability		Implicit Intraoperability; Explicit Interoperability; No Interoperability
	Governance		Open-source Community Mode; Technical Mode; Alliance Mode
	Script Language		Turing Complete; Generic Non-Turing Complete; Application-specific Non-Turing Complete; Non-Turing Complete + External Data
Security and Privacy	Data Encryption		SHA-2; SK-SNARKS
	Data Privacy		Built-in data privacy; Add-on data privacy
Codebase	Coding Language		Single Languages; Multiple Languages
	Code Licence		Open Source; Closed Source
	Software Architecture		Monolithic Design; Polylithic Design
Identity Management	Access Layer and Control Layer		Public Blockchain; Permissioned Public Blockchain; Permissioned Private Blockchain
	Identity Layer		KYC/AML; Anonymous
Charging and Rewarding System	Reward System		Lump-sum Reward; Block + Security Reward
	Fee System	Fee Reward	Mandatory Fees; No Fees; Optional Fees
		Fee Structure	Variable Fees; Fixed Fees

FIGURE 28: Blockchain Ontology Matrix

References

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[2] WEF Financial Services. The future of financial infrastructure. An ambitious look at how blockchain can reshape financial services. Technical report, WEF, 2016.

[3] Paolo Tasca. The dual nature of bitcoin as payment network and money. In *Cash on Trial, ed. Christian Beer, Ernest Gnan, and Urs W. Birchler. Proceedings of the SUERF (Société Universitaire Européenne de Recherches Financières)*, 2016.

[4] Standards Australia. Roadmap for Blockchain Standards. Technical report, Standards Australia, 2016.

[5] Paolo Tasca. Digital currencies: Principles, trends, opportunities, and risks. `http:dx.doi.org/10.2139/ssrn.2657598//`, 2015.

[6] Paolo Tasca Tomaso, Aste and Tiziana di Matteo. Blockchain technologies: foreseeable impact on industry and society. *preprint IEEE Computer*, 2017.

[7] Christopher D Clack, Vikram A Bakshi, and Lee Braine. Smart contract templates: essential requirements and design options. *arXiv preprint arXiv:1612.04496*, 2016.

[8] Daniel Cawrey. Why New Forms of Spam Could Bloat Bitcoin's Block Chain. `http://www.coindesk.com/new-forms-spam-bloat-bitcoins-block-chain/`, 2014. (Date last accessed: 13-Jan-2017).

[9] Ken Shirriff. Hidden Surprises in the Bitcoin Blockchain and How They are Stored: Nelson Mandela, Wikileaks, Photos, and Python Software. `http://www.righto.com/2014/02/ascii-bernanke-wikileaks-photographs.html#ref9`, 2015. (Date last accessed: 11-May-2015).

[10] Xiwei Xu, Ingo Weber, Mark Staples, Liming Zhu, Jan Bosch, Len Bass, Cesare Pautasso, and Paul Rimba. A taxonomy of blockchain-based systems for architecture design. In *ICSA'17: IEEE International Conference on Software Architecture*, Gothenburg, Sweden, April 2017.

[11] Juri Mattila. The blockchain phenomenon. *The Blockchain Phenomenon(Berkeley Roundtable of the International Economy, 2016, edn.)*, 2016.

[12] Aaron Wright and Primavera De Filippi. Decentralized blockchain technology and the rise of lex cryptographia, 2015.

[13] Ripple. `http://www.ripple.com`, 2015. (Date last accessed: 01-June-2015).

[14] Kevin Driscoll, Brendan Hall, Håkan Sivencrona, and Phil Zumsteg. Byzantine fault tolerance, from theory to reality. In *International Conference on Computer Safety, Reliability, and Security*, pages 235–248. Springer, 2003.

[15] M. Tsukerman. Proof of stake versus proof of work, 2015.

[16] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *International Conference on Financial Cryptography and Data Security*, pages 436–454. Springer, 2014.

[17] Karl J O'Dwyer and David Malone. Bitcoin mining and its energy footprint, 2014.

[18] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. Proofs of space. In *Annual Cryptology Conference*, pages 585–605. Springer, 2015.

[19] Sunoo Park, Krzysztof Pietrzak, Joël Alwen, Georg Fuchsbauer, and Peter Gazi. Spacecoin: A cryptocurrency based on proofs of space. Technical report, IACR Cryptology ePrint Archive, 2015: 528, 2015.

[20] Ray Patterson. Alternatives for proof of work, part 2: Proof of activity, proof of burn, proof of capacity, and byzantine generals. `http://https://bytecoin.org/blog/proof-of-activity-proof-of-burn-proof-of-capacity/`, 2015. (Date last accessed: 25-April-2017).

[21] Marko Vukolić. The quest for scalable blockchain fabric: Proof-of-work vs. bft replication. In *International Workshop on Open Problems in Network Security*, pages 112–125. Springer, 2015.

[22] Christian Decker and Roger Wattenhofer. Information propagation in the bitcoin network. In *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on*, pages 1–10. IEEE, 2013.

[23] Donald R Morrison. Patricia—practical algorithm to retrieve information coded in alphanumeric. *Journal of the ACM (JACM)*, 15(4):514–534, 1968.

[24] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 151, 2014.

[25] Ethereum. `https://ethereum.org/`, 2015. (Date last accessed: 10-July-2015).

[26] Dr. Wood G. Etherum: A secure decentralised generalised transaction ledger, yellow paper, 2014.

[27] Guide. `https://bitcoin.org/en/developer-guide#block-chain-overview`, 2015. (Date last accessed: 25-April-2017).

[28] Claudio J. Tessone and Paolo Tasca. A parsimonious model for blockchain consensus: Scalability and consensus collapse, 2017.

[29] Ittay Eyal. The miner's dilemma. In *Security and Privacy (SP), 2015 IEEE Symposium on*, pages 89–103. IEEE, 2015.

[30] Claudio J. Tessone and David Garcia. Bitcoin: The centralisation of a dencetralised economy, 2017.

[31] Misha Tsukerman. The block is hot: A survey of the state of bitcoin regulation and suggestions for the future. *Berkeley Tech. LJ*, 30:1127, 2015.

[32] George F Hurlburt and Irena Bojanova. Bitcoin: Benefit or curse? *IT Professional*, 16(3):10–15, 2014.

[33] Vitalik Buterin et al. A next-generation smart contract and decentralized application platform. *white paper*, 2014.

[34] Michael Crosby, Pradan Pattanayak, Sanjeev Verma, and Vignesh Kalyanaraman. Blockchain technology: Beyond bitcoin. *Applied Innovation*, 2:6–10, 2016.

[35] Campbell R Harvey. Cryptofinance. *Browser Download This Paper*, 2016.

[36] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Scalable zero knowledge via cycles of elliptic curves. In *International Cryptology Conference*, pages 276–294. Springer, 2014.

[37] Paolo Tasca, Shaowen Liu, and Adam Hayes. The evolution of the bitcoin economy: Extracting and analyzing the network of payment relationships, 2016.

[38] Developer guide mining. `https://bitcoin.org/en/developer-guide#mining`, 2015. (Date last accessed: 25-April-2017).