Taxonomy of Blockchain Technologies. Principles of Identification and Classification

Paolo Tasca

*Centre for Blockchain Technologies,
University College London, London,
United Kindgdom*

Claudio J. Tessone

*URPP Social Networks,
University of Zurich,
CH-8050 Zürich,
Switzerland.*

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 layouts for the subcomponents are identified and compared between them. Finally, a taxonomy tree summarises the study and provides a navigation tool across different blockchain architectural configurations.

I. INTRODUCTION

A. 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, originally introduced by Satoshi Nakamoto in 2008 (Nakamoto, 2008). 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 (and sometimes even "corrupted") participants can establish agreements on shared states for decentralised and transactional data in a secure way and without the need of a central point of control or regulatory supervision. Blockchain ensures trust among anonymous counterparts in decentralised systems without the need of central supervisor authorities in charge of verifying the correctness of the records in the ledger. Blockchain has been announced as a disruptive technological innovation, but in fact there is no true technical innovation in Bitcoin and blockchain. All components had already been developed before the Bitcoin paper by Nakamoto in 2009. From a historic perspective, the technology has its roots in Ralph C. Merkle's elaborations, who proposed the Merkle Tree the use of concatenated hashes in a tree for digital signatures in the 1970s. Hashing has been used since the 1950s for cryptography for information security, digital signatures and message-integrity verification. A decade later, Leslie Lamport proposed using the a hash chain for a secure login. The first crypto currency for electronic cash was described at the dawn of the web in 1990. Further evolutions and refinements of the hash chain concept were introduced in a paper by Neil Haller on the S/KEY application of a hash chain for Unix login, in 1994. Adam Back proposed hashcash in 2002, but the first eletronic currency based on blockchain with the PoW concept has been proposed by Satoshi Nakamoto's disruptive paper (Nakamoto, 2008) While blockchain is still in its emergent technological phase, it is fast evolving with the potential to see 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 (R., 2016). 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 features of money with those of a payment system (Tasca, 2016), 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 than 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.

B. Problem Statement and Research Method

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 run 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 those of the applications sitting on top of them. To summarise, current variations of blockchain software architectures pose greatest concerns from different perspectives according to heterogeneity:

1. Heterogeneity is a problem according to the future developments of blockchain technologies, because it will prevent the developments, adoption and stimulation of innovation.

2. Heterogeneity will prevent consistency in drafting laws and policies related to the regulation of blockchain/DLT technologies.

3. Heterogeneity will increase ambiguity in the application of consumer protection laws and regulations.

4. Heterogeneity will decrease the clarity on how the workforce may be affected by blockchain/DLT technology.

5. Heterogeneity will decrease the clarity in academic research and sharpen concepts that underpin the development of new applications and solutions.

6. Heterogeneity will prevent the development of the specification and use of solutions using blockchain and DLT for ISO, IEC and other SDOs.

7. Heterogeneity will increase the complexity in the understanding of blockchain/DLT for NGOs and how this technology may be applied in the relevant sectors to achieve social and economic goals

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. Standards can emerge naturally because of market adoption (industry driven) or because imposed by institutes and organisations. In the first group we may include initiatives like the Accord Project [1], the ChinaLedger [2] or R3 [3]. In the second group we may refer to the initiative conducted by the International Organization for Standardization (ISO) with the establishment of the technical committee ISO/TC 307 on Blockchain and distributed ledger technologies. Several working groups with different topics to discuss have been settled. In particular, the ISO/TC 307/WG1 working group is engaged with the reference architecture, taxonomy and ontology. Overall, a long-term standardisation of the blockchain reference architecture will benefit every industry. Thus, a standard for software reference architecture is necessary in order to enable a level playing field where every industry player and community member 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, a multi-party approach is necessary 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, the lack of standards could bring risks related to privacy, security, governance, interoperability and risk to users and market participants, which can appear as blockchain related cyber crimes. 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 (Standards Australia, 2016). Given the above problem statement the goal of this research is to conduct a review of the blockchain literature. This will be a preparatory work in order to identify and logically group different blockchain (main and sub) components and their layouts. In order to achieve our goal we propose a blockchain taxonomy. Taxonomy comes from the term "taxon" which means a group of organisms. In our case, taxonomy encompasses the identification, description, nomenclature, and hierarchical classification of blockchain components. This is different from an ontology which would be more focused on the study of the types, properties, and interrelationships of the

[1] https://www.accordproject.org/
[2] http://www.chinaledger.com/
[3] https://www.r3.com/

components and events that characterize a blockchain system. The methodological approach is composed of the following steps:

1. Analysis across 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 an online database [4]. This brings together a vocabulary of key blockchain terms to provide readers with a foundation upon which understanding the classification and taxonomy developed in the rest of the analysis. The identification of the blockchain components is the crucial part of this analysis. 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. See Table I in Appendix and (Tasca, 2015) for more information.

2. Framework setting. After the comparative study, a hierarchical taxonomy (a tree structure of classifications for a given set 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, as the technology keeps evolving, the layouts are increasing over time. Thus, for the sake of simplicity, we limit our study to two or three main layouts per each *sub* or *sub-sub* component.

C. Results

The result of the component-level analysis is a universal blockchain taxonomy tree that groups (in a hierarchical structure) the major components, identifies their functional relation and possible design patterns.

In general, it is difficult to evaluate whether a taxonomy or an ontology is good or bad, especially if the domain is a moving target like the blockchain. Taxonomies and ontologies are generally developed to limit complexity and organise information but all serve different purposes and generally evolve over time (see e.g. the evolution of the famous Linnaean taxonomy in biology). Thus, our taxonomy simply aims to contribute to set the foundations for classifying different kinds of blockchain components. Without claiming to represent the ultimate structure, the proposed taxonomy could 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;

[4] http//arstweb.clayton.edu/interlex

2. be propeadeutic to the development of blockchain standards with the aim to increase the adoption at a 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) (Marian, 2015).

II. BACKGROUND ON BLOCKCHAIN TECHNOLOGIES

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, we take a bird-eye view and describe the blockchain by looking at its key driving principles such as data decentralisation, transparency, security, immutability and privacy (Aste *et al.*, 2017).

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 owns what). The decentralised consensus on transactions governs 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 holds equal rights and ability to access the ledger. The records are thus transparent and traceable. Moreover, 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. Blockchain is a shared, tamper-proof replicated ledger where records are irreversible and cannot be forged thanks to one-way cryptographic hash functions. Although security is a relative concept, we can say that blockchains are relatively secure because users can transfer data only if they posses a private key. Private keys are used to generate a signature for each blockchain transaction a user sends out. This signature is used to confirm that the transaction has come from the user, and also prevents the transaction from being altered by anyone once it has been issued.

Immutability. Blockchains function under the principle of non-repudiation and irreversibility of records. Blockchains are immutable because once data has been recorded in the ledger, it cannot be secretly altered ex-post without letting the network know it (data is tamper-resistant). In the blockchain context immutability is preserved thanks to the use of hashes (a type of a mathematical function which turns any type of input data into a fingerprint of fixed size, that data called a hash. If the input data changes even slightly, the hash changes in an unpredictable way) and often of blocks. Each block includes the previous blocks hash as part of its data, creating a chain of blocks. Immutability is relative and relates to how hard the history of transactions is to change. 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". For public proof-of-work blockchains such as Bitcoin, the immutability is related to the cost of implementing the so-called 51% attack. For private blockchains, the block-adding mechanism tends to be a little different, and instead of relying on expensive proof-of-work, the blockchain is only valid and accepted if the blocks are signed by a defined set of participants. This means that, in order to recreate the chain, one would need to know private keys from the other block-adders. A complete discussion of threats about immutability of the transaction history can be found in (Barber *et al.*, 2012). On the other hand, from a governance perspective, this solution is never fully realised. The several examples where the Bitcoin community had reverted Bitcoin blocks based on community decisions. The division between Ethereum and Ethereum classic, and later between Bitcoin and Bitcoin Cash and Bitcoin Gold are not purely anecdotal evidence: they are strong indicators of the importance that the governing body - even if informal - ends up having on the information eventually stored in the blockchain (Walch, 2017).

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 reconciled 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 (Clack *et al.*, 2016)) with 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 (Cawrey, 2014). For example, metadata can be used 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.

III. TAXONOMY OF BLOCKCHAINS

The diversity of blockchain research and development provides an opportunity for cross-fertilisation of ideas and creativity, but it can also result in fragmentation of the field and duplication of efforts. One solution is to establish standardised architectures 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 (Xu *et al.*, 2017), and the problem of consistently engineering large, complex blockchain systems remains largely unsolved. We approach this problem by proposing a component-based blockchain taxonomy starting from a coarse–grained connector–component analysis. The taxonomy compartmentalises the blockchain connectors/components and establishes the relationships between them in a hierarchical manner. We adopt a reverse-engineering approach to unbundle the blockchains and divide them into *main* (coarse-grained) components. Each main component is then split into more (fine-grained) *sub* and *sub-sub* components (where necessary). For each of these sub (and/or sub-sub) components, different *layouts* (models) are identified and compared. By deriving the logical relation between (main, 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.

Similarly to other fields like electronics or mechanics (Otto and Wood, 1998), the software engineering approach used to derive the taxonomy threats blockchains as the result of gluing together prefabricated, well-defined, yet interdependent components. Although equivalent components provide similar services and functions, they can be of different importance and type, and the interconnection may work in different ways. 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.

IV. CONSENSUS

The first identified main component is *Consensus.* It relates to the set of rules and mechanics that allows to maintain and update the ledger and to guarantee the trustworthiness of the records in it, i.e., their reliability, authenticity and accuracy (Bonneau *et al.*, 2015). *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-resistence and tamper-proof (Mattila, 2016). The set of rules and mechanics compose the framework of the validation process that is necessary to overcome security issues during the validation. Figure 1 illustrates the subcomponents forming the component Consensus:

1 Consensus Network Topology

2 Consensus Immutability and Failure Tolerance

3 Gossiping

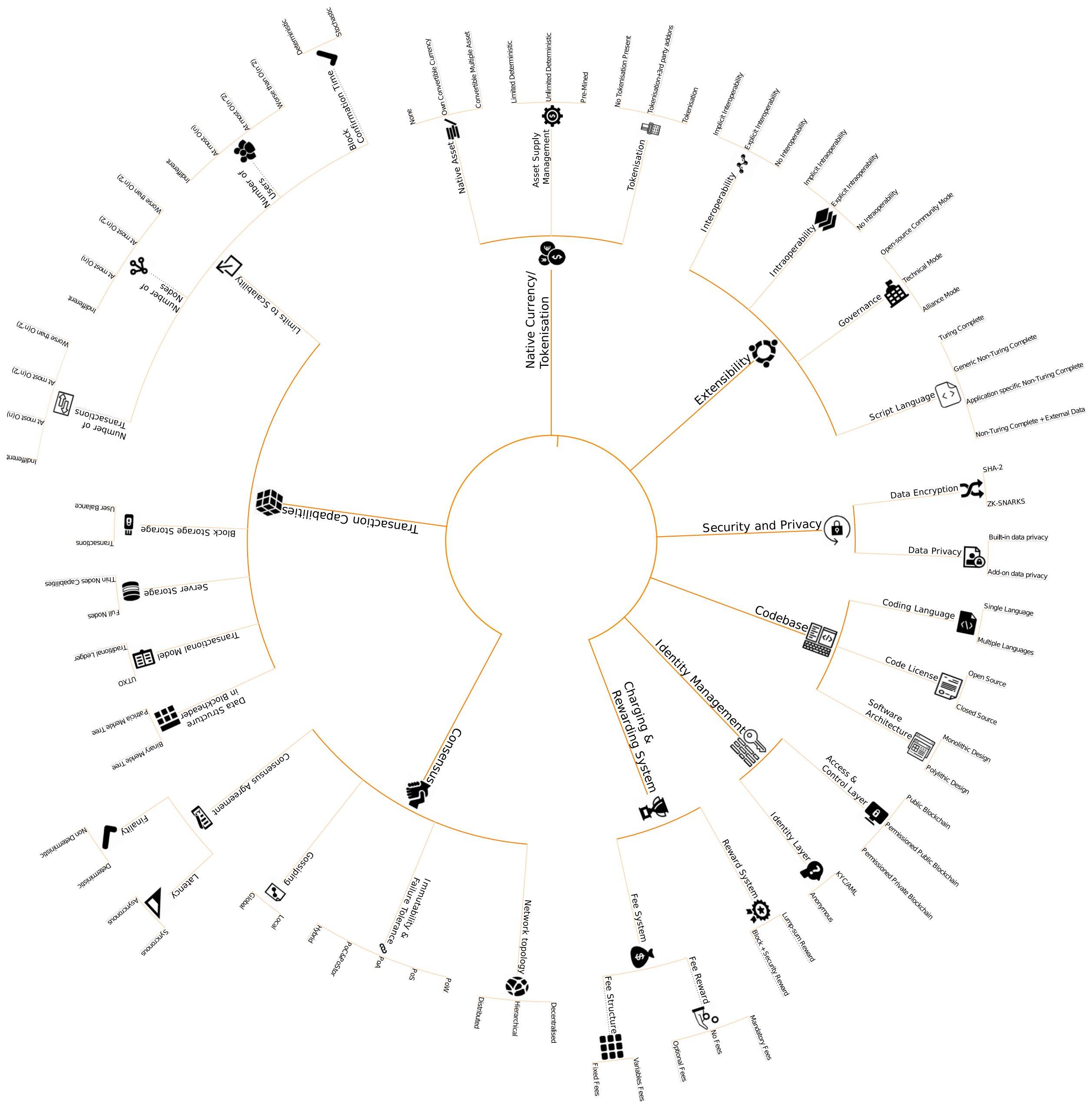

FIG. 1 Blockchain Taxonomy Tree: A representation of the taxonomic decomposition of blockchain-based technologies.

4 Consensus Agreement

4.1 Latency

4.2 Finality

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

A. Consensus Network Topology

Consensus Network Topology describes the type of interconnection between the nodes and the type of information flow between them for transaction and/or for the purpose of validation. For efficiency reasons, systems have historically 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, this 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 of our techno- and socio-economic systems (one example is the modern electronic payment system). To avoid the single point of failure, these centralised can be extended into hierarchical constructs which exhibit larger scalability and more redundancy, while keeping the communication efficient. Alternative to those centralised topologies, decentralised solutions have been proposed. Since the dawn of the Internet, technical systems have evinced a transition towards decentralised arrangements (Wright and De Filippi, 2015) where all the nodes are equivalent to any 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 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 this is not the only determinant. Also other factors like the reward mechanism (see Section XI) heavily influence the validation (Bonneau *et al.*, 2015). As a matter of an example, Bitcoin has a decentralised validation process, which is still be accompanied by ever increasing concentration of computational power devoted to the proof-of-work by 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% to 80% range (Tasca, 2015).

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 (Nakamoto, 2008) connected in a random way as provided by super-nodes. This network illustrates a decentralised *Consensus Network Topology*. Obviously, this layout is independent from 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 irregularity of the role the nodes have. For example, in Ripple (Ripple, 2015) 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 (Ripple, 2015). This kind of solution yields (or can be extrapolated to) a hierarchical network topology. In the community of developers these hierarchical topologies are also 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 (or 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.

B. Consensus Immutability and Failure Tolerance

In general, the failure tolerance of a distributed system shall be defined with respect to three interrelated issues: faults (e.g., Byzantine faults), errors and failures. See e.g., (Driscoll *et al.*, 2003; Castro *et al.*, 1999) for Byzantine fault tolerance in distributed systems. There are different types of failures and generally it is costly to implement a fault tolerant system. Practically, it is not possible to devise an infallible, reliable system . For a literature review and deeper analysis of fault tolerant distributed systems, we refer the reader to (Cristian, 1991) and (Fischer, 1983). A blockchain, as special case of a distributed system, is fault tolerant when it shows the ability to continue functioning. i.e., it must grant reliability, validity and security of the information stored in the ledger. Indeed, blockchains represent a decentralised solution to the problem of information storage which require no central database but many duplicates such that each server holds 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. Therefore, a blockchain system is in the need of an efficient consensus mechanism to ensure that every node has its original version of the full transaction history which is kept consistent with the other peers over time. In this vein, the immutability of achieved consensus differs with respect to the resources required to keep large network security. In the past years, the evolution of blockchain technologies has been accompained with the development of different mechanisms that help to keep reliable, valid and secure the information contained in the ledger. All together, the mechanisms for *Consensus Immutability* together with the subcomponents of *Consensus Agreement* determine the failure tolerance of the blockchains. As of this writing, we identify six main layouts for *Consensus Immutability and Failure Tolerance*:

1. **Proof-of-Work**. The most widely used cryptocurrency, Bitcoin, uses Proof-of-Work (PoW) to ensure the immutability of the transaction records. In this setup, computing devices, usually called *miners*, connected to a peer-to-peer network perform the task of validating the transactions proposed for addition to the complete record of existing - valid - transactions. The generation of a block that

can be appended to the blockchain - rendering in this way valid all transactions there included - requires finding the solution of inverting a cryptographic function, which can only be done by brute force. In PoW, the probability that a miner mines a new block depends on the ratio between the computational power he devotes to this task and the total instantaneous computational power by all miners connected to the network. Specifically, miners must find a solution to an one-way hash function by computing new hash values based on the combination of the previous hash values contained in the message, the new transactions in the block they create and a nonce. The solution is such that the new hash value will start with a given number of zeros $\leq$ target. As for this writing, the mining process needs several requirements to be successful (Tsukerman, 2015). These include specialised hardware which is needed to perform the computational tasks and ever increasing amounts of electricity to power the hardware. These computations are run by dedicated machines (ASICs) which are very expensive and are resource-intensive as contribute to a large electricity footprint for cryptocurrency miners (O'Dwyer and Malone). Due to this scheme, in the last years miners agglomerate around mining pools (Lewenberg *et al.*, 2015). Therefore, 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. In late 2017, every second almost five quadrillion SHA256 computations were performed in the Bitcoin mining process. Regretfully, these computations do not have any practical or scientific relevance apart from ensuring that the process of block creation is costly, but others' blocks validity are simple to verify for peers. Interestingly, when adversaries coordinate, it is sufficient that they hold only the 25% of the total computing power to mount an attack (Eyal and Sirer, 2014). In this layout, there exists the risk of monopoly mining, induced by large coordination of miners in a single mining pool, which continuously increases the expected payoff of others if they join said mining pool. In this hypothetic situation, said maining pool can censor specific transactions and dictate what transactions are accepted and which ones not. 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 an equivalent goal. It is worth mentioning that some cryptocurrencies (e.g. Primecoin) have tried to make the PoW a task that serves a useful aim (in that case, searching for long chains of prime numbers, or Cuningham series).

2. **Proof-of-Stake**. PoS links the block generation to the proof of ownership of a certain amount of digital assets (e.g., digital currencies) linked to the blockchain. The probability that a *prover* is selected to verify the next block is larger the larger is the share of assets this prover has within the system. The underlying assumption is that users with a large share of the system wealth are more likely to provide trustworthy information with respect of the verification process, and are therefore

to be considered trusted validator (Mattila, 2016). Two alternative PoS methods have been devised. The first one is based on randomised block selection (used in e.g., NXT and BlackCoin); it uses a calculation searching for the lowest hash together with the stake size; it is therefore somewhat deterministic and each node can independently determine the likelihood of being selected in a future round. An alternative scheme is the *coin-age*-based selection (used by e.g., Peercoin, being actually the first one to be implemented in real world) which combines randomisation with coin-age (a number derived from multipliying the amount of the assets held by the prover and the length of time it has been helding them). 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 can be addressed by Delegated Proof-of-Stake (DPoS), a generic term describing an evolution of the basic PoS consensus (utilised in, e.g., BitShares, Casper by Ethereum, Tendermint) where blocks are forged by a predetermined users delegated by the user who has the actual stake. These forgers 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 preselected real entities (i.e., the *authorities*) are allowed to control the content that is added to the public registry. Those nodes will receive a set of private keys that will be used to "sign" the new blocks, acting as *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: control of mining frequency, distribution of mining load (and opportunity) between the various signers and; maintenance of the the list of signers such to be robust from malicious attacks even in presence of dynamic mutation of the trusted signers.

4. **Proof-of-Capacity/Proof-of-Space and Proof-of-Storage**. PoC or PoSpace and PoStorage are implementations of the popular idea of "space as resource". Here the focus is not on the CPU cycles but on the amount of actual memory (non-volatile) space 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 devote 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. These schemes are more fair and green than 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, just to cite a few. PoC consists of an initialisation and subsequent execution between a prover P and a verifier V (Dziembowski *et al.*, 2015). 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. The PoC is in general defined by three quantities: (N_0, N_1, T); then, the miner shows that she either: 1) had access to at least N_0 storage between the initialisation and execution phases and at least N_1 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 proposed by (Park *et al.*, 2015) 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 (Patterson, 2015).

5. **Proof-of-Burn**. In Proof-of-Burn (PoB) miners must prove that they burned some digital assets. They do so by sending them (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 (Bonneau *et al.*, 2015). 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 developer'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.

C. 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*.

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 (Ripple, 2015) 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 sent 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. Upon connection of a new node, the submit a randomly chosen list of peers to the entrant one. The logical network topology is intended to be largely unstructured, similar to the Erdos-Renyi network (Barabsi and Psfai, 2016). Such topology lacks a concept of vicinity or local neighbourhood, and therefore the *Gossiping* process can be termed *global.*

D. Consensus Agreement

The *consensus agreement* defines the set of rules under which records (like sets of transactions or any other atomic piece of information) 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 on storing verified, trustworthy, information even in the presence of f malicious nodes or in presence of malicious participants launching sybil attacks (Douceur, 2002). In this regard, it is very important to understand how the nodes communicate between them.

1. Latency

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

1. **Synchronous Communication.** Systems which set upper bounds on "process speed interval" and "communication delay" such that every message arrives within a certain known, predefined, time-interval (Δ) are called *synchronous.* This does not preclude the possibility of having message delays due to exogenous network latency, but the delay is bounded and any message that takes longer than

Δ is discarded. Lax 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" plus 2 hours. Another example of blockchain adopting *synchronous communication* is Ripple through the use of clocks. Specifically, Ripple's "LastLedgerSequence" parameter asserts 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), its main disadvantage is that response times are unpredictable and it is harder to design applications based on them. Synereo is an example of blockchain using the asynchronous communication protocol.

2. Finality

Finality describes whether information intended to be stored in a blockchain (or, as a matter of fact, in any system) can be safely considered *perpetually* stored once the recording is performed. For a distributed system like blockchain-based ones this is very challenging to achieve, and it is certainly not one of the underlying design principles. In a system where the new blocks diffuse through gossiping, and because of rules such as the precedence of the longest chains, even if consensus is achieved globally, *a priori* nothing prevents a set of new nodes entering the system and overriding the previous consensus by offering a longer versions of the hostory. We identify two possible layouts for *Finality*:

1. **Non-Deterministic**. In this case, *consensus agreement* "eventually settles". Non-deterministic are randomised or inherently 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) to minimise the probability of forks. Moreover, the propagation of blocks through the network has characteristic delays (Decker and Wattenhofer, 2013) and even in presence of only honest nodes the fork probability cannot be ruled out simply because different nodes may find competing blocks of the same height before the one found first reaches the complete network . This cannot be prevented even if there is in place a concurrency control mechanism, which which attempts to correct results for simultaneous operations. Therefore, overall, the protocol is non deterministic. Thus, even though the widespread heuristic "wait until 6 confirmed blocks are appended to the chain" reduces the likelihood that a transaction is overridden afterwards, it does not eliminate completely 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 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 (Lamport *et al.*, 1982) achieve deterministic consensus. A prime example of an implementation featuring deterministic finality is Stellar. Another case of deterministic is for private blockchains where new blocks follow a predefined set of rules.

V. 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. centralised payment systems, like 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. Figure 1 illustrates the subcomponents forming the component Transaction Capabilities:

1 Data Structure in the Blockheader

2 Transaction Model

3 Server Storage

4 Block Storage

5 Limits to Scalability

  5.1 Transactions

  5.2 Users

  5.3 Nodes

  5.4 Confirmation Time

A. 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 purposes; on the other, it contains additional information for different application layers or blockchain technology platforms. The *data structure in the blockheader* describes the capabilities of the system to store transaction information. The original application of Merkle proof was implemented in Bitcoin, as described in (Nakamoto, 2008). We identify two possible layouts for Data Structure in the blockheader:

1. **Binary Merkle Tree.** Bitcoin uses the Binary Merkle tree (Merkle, 1988) 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 (Morrison, 1968)) allows activities like inserting, editing or deleting information referring to the balance and nonce of accounts, which enables faster and more flexible validation of transactions than the one *merkle tree model* (Wood, 2014). However, with respect to the applications, it has the important advantage of allowing for verification of specific branches of the tree. Ethereum (Ethereum, 2015) 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 (Wood, 2014). Importantly, this technology allows even blocks outside 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 (Wood, 2014).

B. Transaction Model

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. The challenge of the *transaction model* is to prevent data that ought not to be trusted by the parties connected to the system - e.g. those originated in behaviour, like double spending - to enter into the ledger As of this writing, it is possible to identify two possible layouts for *Transaction Model* in the widely used blockchain-based systems:

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 (Guide, 2015). This transactions mechanism is available in blockchain technologies like Bitcoin.

2. **Traditional Ledger.** In comparison to the *UTXO model*, different implementations of blockchain systems - like *Stellar* and *Ripple* use a more traditional ledger model to record the transactions

recorded in the system. In particular, Stellar lists every single transaction in the *Stellar distributed ledger history*. Also, *Ripple* uses the traditional ledger transaction model to register increments/decrements of balance and clear all account balances. In *Ethereum* some transactions are used to execute actions in smart contracts defined in specific atomic records in the blockchain. Those transactions can be seen as order executions of stakeholders which perform the actions out of said smart contracts.

C. 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 (Xu *et al.*, 2018). 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.

D. Block Storage

Which information is stored in the blockchain determines the scalability of the system across some dimensions. More crucially, it also allows to understand how concomitant information from users are abstracted within the system. 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 emitter(s) and receiver(s) of a specific transaction. This kind of approach is preserved in more exotic applications of blockchain-like technologies, like IOTA, which relies on the storage of an directed-acyclic-graph to store every single transaction. This kind of approach works not only for cryptocurrencies applications, but it is underlying all transfer-of-property-like applications.

2. **User balance.** In systems like Ripple, the decentralised storage also contains information about the user balance in the specific assets. This approach may limit the storage needs of the system, but at the same time reduces accountability and the possibility to roll back transactions.

E. 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 may refer to to the number of nodes connected to the network, the number of users of the service, the set of network connections and/or amount of network traffic, the number of transactions, etc. It is worth remarking that these ingredients are intertwined in the real world (Tessone and Tasca, 2017), and - upon usage and continuous development - the limiting factor of a particular blockchain system may vary over time. In a rapidly evolving technology such as blockchain is nowadays, these limits are often changed by the development teams behind some of these systems. An example is the implementation (or not) of SegWit2x and Lightning (as technology for micropayment channels) (Decker and Wattenhofer, 2015) as ways to alleviate the limitation in the number of transactions that Bitcoin can process with respect to its current implementation.

The importance of this component is due to its influence on the final scaling of the system. Scaling is a property that specifies how the growth will influence its overall performance. As an example, how the total network traffic induced by unverified transactions grow with the number of network nodes. If every node has a small - limited - number of connections, then the total network 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 - for a given implementation - network traffic is the most crucial limiting factor, different logical topologies will have different scalability. Acknowledging that a categorical definition is a crude simplification, we will focus here on the most limiting element for each system. We identify four possible layouts as *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 given time window is limited. Therefore the layout “Number of Transactions” refers - regardless of the information stored in the blockchain(Eyal, 2015) - to the specific implementations, where 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

that limits its scalability. A similar problem occurs in Ethereum where the system will be constrained by the number of DAOs, individuals, 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 states 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 can still appear irrespective 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 (Tessone and Garcia, 2017), 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.

4. *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 (Catalini and Gans, 2017) when the relevant global characteristics of a system is independent of the number of specific class; the other 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.

5. **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.

VI. NATIVE CURRENCY/TOKENISATION

So far, cryptocurrencies and other transfer of property records are the most common usage of the blockchain technology. In cryptocurrencies, system participants who contribute to the verification process - if selected by some rule to issue a new block into the blockchain - are awarded the possibility to issue a transaction without issuer (so called “coinbase”) to themselves. On the one hand, this is a customary way of introducing new assets into the system. On the other, it introduces an incentive for users to participate of the verification process which leads to an increased trustworthiness on the system.

The aforementioned incentive scheme (Sompolinsky and Zohar, 2018) is to be provided in a token, whose value is assigned (Catalini and Gans, 2017) precisely because of the cost associated with its production (Garcia *et al.*, 2014). Initially, solutions like Bitcoin have created its own (and single) asset class (*the bitcoin*) that can be transacted within the system. This particular solution is not the only one possible with the primary example of Ethereum, where beyond the natural Ether native token, via smart contracts arbitrary new tokens can be created and their property exchanged. Further, the native currency possibilities present for example in Ripple (Tsukerman, 2015) and tokenisation enable different use cases of the blockchain technology like asset-transfers via tokens, exchanges, etc. All this is just the beginning of cryptoeconomics: it is of uttermost importance how these assets are supplied into the system, because this affects the way users are incentivised to participate in the validation process. Figure 1 illustrates the subcomponents forming the component Native Currency/Tokenisation:

1 Native Asset

2 Tokenisation

3 Asset Supply Management

A. Native Asset

Some systems implemented using blockchain technologies have underlying a native asset (which are normally called *cryptocurrency*) which is a digital token whose owners assign a value and allow to run the daily activities on the platforms or communities. Whether these cryptocurrenties ought to be considered fiat or commodity currencies (Grinberg, 2012; Selgin, 2015; Luther, 2018), and whether they may eventually be massively adopted replacing traditional ones (Luther, 2016). 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 Cryptocurrency.** Most implementations of cryptocurrencies only deal with transfer of property of its own tokens within the system. Bitcoin or Litecoin are examples of technologies with single asset compatibility (Buterin *et al.*, 2014). 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 enrol into smart contracts. Further, solutions like coloured-coins (Rosenfeld, 2012).

3. **Convertible Multiple Assets.** Other technologies like Counterparty, Ardor o 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 those native to the platform. This approach

of multiple, convertible, currencies has the advantage of allowing for exchange markets be directly reflected into the system.

B. 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(Catalini and Gans, 2017) enables a range of possible use cases for the blockchain technologies outside the purely financial world(Tsukerman, 2015; roh, 2017; adh, 2017; Conley *et al.*, 2017).

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 Colour-Coin (Rosenfeld, 2012) 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. The most paradigmatic example is Ethereum, where the creation of a new Token is produced by means of the creation of a smart contract. Thanks to this flexibility, and extreme extension possibility of such platform, the conditions for creation of new tokens is countless.

C. Asset Supply Management

The process of the digital asset (usually referred to as *cryptocurrency*) 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 (Tessone and Garcia, 2017).

We identify three 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.

VII. 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 related technology. Figure 1 illustrates the subcomponents forming the component Extensibility:

1 Interoperability

2 Intraoperability

3 Governance

4 Script Language

A. 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(Dilley *et al.*, 2016). 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.

B. 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(CHEN *et al.*, 2017). 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.

C. 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 (Atzori, 2015; Davidson *et al.*, 2016; Wright and De Filippi, 2015). Regulatory rules are by definition not technical in nature and therefore outside the scope of this taxonomy. We focus instead on techical rules which are particularly interesting for their feedback loop with the proposed technological solutions. We identify three possible layouts *Governance*:

- **Open-source Community.** 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.

- **Technical**. 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 Microsoft) 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 (R., 2016) 2,500 patents on this topic have been filed from 2014 to 2016.

- **Alliance**. 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.

D. 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 which aim at prevent specific behaviours of code execution, like undefined termination.

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, affects 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 (Kim and Laskowski, 2018).

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. 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 completeness for 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 (Atzei *et al.*, 2017). 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 allows obtaining data from sources that gather data which is exogenous to the blockchain.

3. **Application-specific Non-Turing Complete**. There are some non-Turing complete languages that are more expressive than the generic ones and purposely designed for certain cases. By restricting the language to be only 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".

VIII. SECURITY AND PRIVACY

The recent evolution and new implementations of blockchain systems bring risks, both technical and operational, associated with security and privacy. Thus, we group together security and privacy as two interrelated faces of the same problem. Similarly, ISO TC 307 has created a dedicated Working Group on "Security and Privacy" (iso).

Security of blockchain systems is a matter of significant concern. Crypto currencies, the most widely deployed application of blockchain systems, have suffered from cyber attacks which became possible because of sensitive data mismanagement and the flawed design of the systems (Lin and Liao, 2017). Without going into the detailed distinction between "risks", "threats", "attack surfaces" and "vulnerabilities", security of blockchain systems concerns: 1) Information mismanagement (alternation, deletition, distruction, disclosure etc.); 2) Implementation vulnerabilities (including cryptomechanisms implementation vulnerabilities, run-time leakage of information etc.); 3) Cryptographic mechanisms mismanagement (including use of weak algorithms, key disclosure); 4) User privileges mismanagement. For a recent comprehensive survey specifically targeting to the security and privacy aspects of Bitcoin and its related concepts we refer the readers to (Conti *et al.*, 2017). With regard to privacy we refer to the "freedom from intrusion into the private life or affairs of an individual when that intrusion results from undue or illegal gathering and use of data about that individual" (ISO 25237 and other ISO standards). These privacy principles apply to any ICT system containing or processing PII, including blockchain systems. Figure 1 illustrates the subcomponents forming the component Security & Privacy:

1 Data Encryption

2 Data Privacy

A. Data Encryption

By *Data Encryption* we refer to cryptographic primitives. To ensure authenticity, integrity property and order of events, cryptographic primitive (cryptographic algorithms) are used. For example, Bitcoin blockchain uses ECDSA digital signature scheme for authenticity and integrity, and SHA-2 hash function for integrity and order of event. Hash functions are also commonly used as a part of Proof-of-Work consensus mechanism. We identify two major 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 (Crosby *et al.*, 2016; Harvey, 2016) 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(Meiklejohn and Orlandi, 2015).

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 (Ben-Sasson *et al.*, 2014). With the hashed message and the encrypted one, is sufficient as a proof to generate the validation. This anonymises much more the individual information.

B. 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 (Tasca *et al.*, 2016). 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 exists a strong interrelation between *Data Privacy* and *Data Encryption*. 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 ZeroCash uses built-in zero-knowledge cryptography to encrypts the payment information in the transactions (Sasson *et al.*, 2014). Although ZeroCash 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)(Wit, 2017) uses 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 (N-to-N) system characterised by encrypted transactions where only the parties involved in the transaction have access to the data (Hearn, 2016). 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 inputs, 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 (Bitcoin Wiki, 2015) (Bitcoin wiki, 2015). 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* (Noether *et al.*, 2016) and *stealth addesses* (Möser and Böhme, 2017) which hide the recipient of a transaction and can be used by any blockchain. Ring signatures - firstly introduced by (Rivest *et al.*, 2001) - 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 wwho can unlock the address by using the nonce and the secret key generated earlier. Monero (https://getmonero.org/) is an example of blockchain that aims to achieve privacy through the use of traceable ring signatures and stealth addresses.

IX. 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 could undergo. Therefore the main component Codebase is essential to align and increase the efficiency of blockchain related IT architectures. Figure 1 illustrates the subcomponents forming the component Codebase:

1 Coding Language

2 Code License

3 Software Architecture

A. Coding Language

Coding language illustrates the interconnectivity of programming languages of the blockchain technologies. 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, implementations occurred (much less popular than the original codebase) 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 maintains JavaScript, Java, and Go-based SDKs

for communicating with Horizon. There are also community-maintained SDKs for Ruby, Python, and C-Sharp.

B. Code License

The Code License illustrates the possibility of changes to the source code of the underlying technology. We identify thee possible layouts for *Code License*:

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. Just as an example, most the blockchains running on the Ethereum Enterprise Alliance, rather than on the public Ethereum blockchain, use closed source codes. In this case, 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.

C. 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 give. 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.

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.

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 Fabric follows a polylithic design as it is composed of interchangeable modules representing different components of blockchain technology.

X. IDENTITY MANAGEMENT

The main component *Identity Management* ensures secure access to sensitive data to establish a suitable governance model for the blockchain. This is a complex matter, as different levels of authority, accountability and responsibility are attached to different type of participants (e.g. users, administrators, developers, validators, etc). Generally, the set of rules are defined and enforced through mechanisms intrinsic to the system itself (on-chain governance). The subcomponents also eventually determine the concept of digital identity that users end up having within the systems. Figure 1 illustrates the subcomponents forming the component Identity Management:

1 Access and Control Layer

2 Identity Layer

A. 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 answer 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 way in order to better serve either a public or a private interest of 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.

Although different variations are possible, the authority to perform transactions on a blockchain generally belongs to one of the following main models of the *Access and control Layer*(Guegan, 2017):

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. An example of blockchain in this area is 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) with intangible (contracts and intellectual property) assets between enterprises.

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).

B. Identity Layer

The onboarding and offboarding of nodes / entities to the blockchain networks is handled differently by the various software solutions. By identification we mean the capability to identify an entity uniquely in a given context. Digital identity can be defined as a set of identifying attributes for an entity that together enable the unique identification of the entity in a context (UID). A vital part of any identity system (and most information systems) is that a UID is managed throughout the entitys lifecycle to protect it from negligence and fraud, and to preserve the UIDs uniqueness. A UID can then be assigned to the identity and used to link or bind the entity to the claimed identity and to any digital credential (software or hardware) issued to the entity. This digital credential acts a trusted proxy for the physical or logical entity and is used to support a wide range of personal and trust-related functions such as authentication, encryption, digital signatures, application logins and physical access control.

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*. We identify two possible layouts:

1. **KYC/AML.** Compliant blockchains have the ability to validate organisations and their attribute data from authoritative sources to ensure the quality of data written to the blockchain and linked to identifiers in the blockchain. An example is Stellar that sets 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 its 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 fulfil.

2. **Anonymous.** In general the common misunderstanding of the anonymity level within Bitcoin networks is that the majority of the users do not distinguish between anonymity and the pseudo-anonymity. In light of this, the Bitcoin protocol has no identity layer to identify the users. Those circumstances could benefit misuse of Bitcoins and money laundering activities through this blockchain network, but to control approaches to anonymity in Bitcoin and other cryptocurrencies(Maurer, 2016). Regal Reid and Martin Harrigan (Reid and Harrigan, 2013) have been able to demonstrate that several pseudonymous addresses can be linked to one single user. See also (Tasca *et al.*, 2016) for a Bitcoin transaction-path driven users identification method.

XI. 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. Figure 1 illustrates the subcomponents forming the component Charging and Rewarding System:

1 Reward System

2 Fee System

    2.1 Fee Reward

    2.2 Fee Structure

A. Reward System

This subcomponent illustrates the rewarding mechanisms automatically put in place and triggered by the systems in order to compensate active members contributing to data storage or transaction validation and verification. 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 verification is only 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, the blockchain rewarding system includes, besides the block reward, a reward for including in the validation forked blocks that are still valid. The design idea is to incentivise cross-validation of transactions (crucial in a setting where validation can be arbitrarily costly)(Timmerman *et al.*, 2017).

B. 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.*

1. Fee reward

Fees Reward describes the nature of the fees that the users are required to contribute when using a blockchain. The fees system has been shown to play an important role in the way verifiers do (Möser and Böhme, 2015) and may (Carlsten, 2016) behave. This kind of design-time consideration ought not to be neglected, as it is usually the case. We identify three possible layouts for *Fees Reward*:

1. **Optional Fees.** In Bitcoin and related technologies (Guide, 2015) 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 Fabric is a blockchain solution for businesses, which combines a permissioned network and an identity layer without any transaction fees.

2. Fee structure

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

XII. CONCLUSION

In the 21st century, the blockchain technologies will athwart affect all business areas: financial services (Alvseike and Iversen, 2017; Scott, 2016; Evans, 2015; Quintana Diaz, 2014), IoT (Boudguiga *et al.*, 2017; Dorri *et al.*, 2017), consumer electronics (Andrews *et al.*, 2017), insurances (Mainelli and von Gunten, 2014; ste, 2016), energy industry, logistics (Badzar, 2016; Hackius and Petersen, 2017), transportation, media (Kotobi and Bilén, 2017), communications (Plant *et al.*, 2017), (Chakravorty and Rong, 2017), entertainment, healthcare (Kuo *et al.*, 2017), 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 architectures 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 standardisation 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.

Based on the review of the current literature on blockchain technologies, our work is an early stage analysis across existing software architectures with the aim to propose a taxonomy: a reference architectural model for blockchains and their possible configurations. Based on component-based design, the blockchain taxonomy decomposes the blockchains into individual functional or logical components and identifies any possible different layout. The blockchain taxonomy proposes to assist in the exploration of design domains, in the implementation, deployment and performance measurement of different blockchain architectures. Figure 1 illustrates the blockchain taxonomy tree resulting from our analysis.

Our work sheds light on the current proliferation of non-interoperable blockchain platforms and on the need (for) and current discussions (about) blockchain standards. Although our work contributes to the ongoing efforts on setting blockchain standards, we do not conclude by saying that we need a set of standards *now*. This process generally takes several years in order to produce concrete solutions (even 10 years for complex subjects). Therefore, we think that our taxonomy represents a timely honest intellectual exercise to be used as preliminary supporting material for all those interested in reducing blockchain complexity. At the same time, we are aware that our taxonomy tree, although hopefully very useful, is very preliminary and likely the first version of subsequent more complex evolutions.

a. Acknowledgements: PT and CJT thank Harvey R. Campbell for his invaluable comments on a previous version of this paper. They also thank Alessandro Recchia and Thayabaran Thanabalasingham for their support and contribution. 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.

REFERENCES

Nakamoto S., "Bitcoin: A peer-to-peer electronic cash system," (2008).

R. J. M., in *World Economic Forum* (2016).

Tasca P., "The dual nature of bitcoin as payment network and money," (SUERF Conference Proceedings 2016/1, 2016) Chap. VI.

Standards Australia, *Roadmap for Blockchain Standards*, Tech. Rep. (Standards Australia, 2016).

Tasca P., (2015).

Marian O., U. Chi. L. Rev. Dialogue, **82**, 53 (2015).

Aste T., P. Tasca, and T. di Matteo, Computer, **50**, 18 (2017).

Barber S., X. Boyen, E. Shi, and E. Uzun, in *International Conference on Financial Cryptography and Data Security* (Springer, 2012) pp. 399–414.

Walch A., (2017).

Clack C. D., V. A. Bakshi, and L. Braine, arXiv preprint arXiv:1612.04496 (2016).

Cawrey D., "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).

Xu X., I. Weber, M. Staples, L. Zhu, J. Bosch, L. Bass, C. Pautasso, and P. Rimba, in *Software Architecture (ICSA), 2017 IEEE International Conference on* (IEEE, 2017) pp. 243–252.

Otto K. N. and K. L. Wood, Research in Engineering Design, **10**, 226 (1998), ISSN 1435-6066.

Bonneau J., A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten, in *Security and Privacy (SP), 2015 IEEE Symposium on* (IEEE, 2015) pp. 104–121.

Mattila J., *The Blockchain Phenomenon–The Disruptive Potential of Distributed Consensus Architectures*, Tech. Rep. (The Research Institute of the Finnish Economy, 2016).

Wright A. and P. De Filippi, (2015a).

Ripple, "http://www.ripple.com," (2015), (Date last accessed: 01-June-2015).

Driscoll K., B. Hall, H. Sivencrona, and P. Zumsteg, in *International Conference on Computer Safety, Reliability, and Security* (Springer, 2003) pp. 235–248.

Castro M., B. Liskov, *et al.*, in *OSDI*, Vol. 99 (1999) pp. 173–186.

Cristian F., Communications of the ACM, **34**, 56 (1991).

Fischer M. J., in *International Conference on Fundamentals of Computation Theory* (Springer, 1983) pp. 127–140.

Tsukerman M., "Proof of stake versus proof of work," (2015a).

O'Dwyer K. J. and D. Malone, in *25th IET Irish Signals & Systems Conference 2014 and 2014 China-Ireland International Conference on Information and Communications Technologies (ISSC 2014/CIICT 2014)*.

Lewenberg Y., Y. Bachrach, Y. Sompolinsky, A. Zohar, and J. S. Rosenschein, in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '15 (International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2015) pp. 919–927, ISBN

978-1-4503-3413-6.

Eyal I. and E. G. Sirer, in *International Conference on Financial Cryptography and Data Security* (Springer, 2014) pp. 436–454.

Dziembowski S., S. Faust, V. Kolmogorov, and K. Pietrzak, in *Annual Cryptology Conference* (Springer, 2015) pp. 585–605.

Park S., K. Pietrzak, J. Alwen, G. Fuchsbauer, and P. Gazi, *Spacecoin: A cryptocurrency based on proofs of space*, Tech. Rep. (IACR Cryptology ePrint Archive, 2015: 528, 2015).

Patterson R., "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).

Barabsi A.-L. and M. Psfai, *Network science* (Cambridge University Press, Cambridge, 2016) ISBN 9781107076266 1107076269.

Douceur J. R., in *Peer-to-Peer Systems*, edited by P. Druschel, F. Kaashoek, and A. Rowstron (Springer Berlin Heidelberg, Berlin, Heidelberg, 2002) pp. 251–260, ISBN 978-3-540-45748-0.

Decker C. and R. Wattenhofer, in *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on* (IEEE, 2013) pp. 1–10.

Lamport L., R. Shostak, and M. Pease, ACM Transactions on Programming Languages and Systems (TOPLAS), **4**, 382 (1982).

Merkle R. C., in *Advances in Cryptology — CRYPTO '87*, edited by C. Pomerance (Springer Berlin Heidelberg, Berlin, Heidelberg, 1988) pp. 369–378, ISBN 978-3-540-48184-3.

Morrison D. R., Journal of the ACM (JACM), **15**, 514 (1968).

Wood G., Ethereum Project Yellow Paper, **151**, 1 (2014a).

Ethereum, "https://ethereum.org/," (2015), (Date last accessed: 10-July-2015).

Wood G., Ethereum Project Yellow Paper, **151** (2014b).

Guide, "https://bitcoin.org/en/developer-guide-block-chain-overview," (2015a), (Date last accessed: 25-April-2017).

Xu Q., K. M. M. Aung, Y. Zhu, and K. L. Yong, in *New Advances in the Internet of Things* (Springer, 2018) pp. 119–138.

Tessone C. J. and P. Tasca, "A parsimonious model for blockchain consensus: Scalability and consensus collapse," (2017).

Decker C. and R. Wattenhofer, in *Symposium on Self-Stabilizing Systems* (Springer, 2015) pp. 3–18.

Eyal I., in *Security and Privacy (SP), 2015 IEEE Symposium on* (IEEE, 2015) pp. 89–103.

Tessone C. J. and D. Garcia, "Bitcoin: The centralisation of a decetralised economy," (2017).

Catalini C. and J. Gans, "Some simple economics of the blockchain. rotman school of management working paper no. 2874598," (2017).

Sompolinsky Y. and A. Zohar, Communications of the ACM, **61**, 46 (2018).

Garcia D., C. J. Tessone, P. Mavrodiev, and N. Perony, Journal of The Royal Society Interface, **11**, 20140623 (2014), ISSN 1742-5689.

Tsukerman M., Berkeley Tech. LJ, **30**, 1127 (2015b).

Grinberg R., Hastings Sci. & Tech. LJ, **4**, 159 (2012).

Selgin G., Journal of Financial Stability, **17**, 92 (2015).

Luther W. J., Journal of Private Enterprise, **33**, 31 (2018), cited By :1.

Luther W. J., Contemporary Economic Policy, **34**, 553 (2016), cited By :8.

Buterin V. *et al.*, white paper (2014).

Rosenfeld M., "Overview of colored coins," (2012), white Paper bitcoin.co.il.

Blockchain-Based Token Sales, Initial Coin Offerings, and the Democratization of Public Capital Markets (2017).

Why do businesses go crypto? An empirical analysis of Initial Coin Offerings (2017).

Conley J. P. *et al.*, *Blockchain and the Economics of Crypto-tokens and Initial Coin Offerings*, Tech. Rep. (Vanderbilt University Department of Economics, 2017).

Dilley J., A. Poelstra, J. Wilkins, M. Piekarska, B. Gorlick, and M. Friedenbach, arXiv preprint arXiv:1612.05491 (2016).

CHEN Z.-d., Y. Zhuo, Z.-b. DUAN, and H. Kai, DEStech Transactions on Computer Science and Engineering (2017).

Atzori M., "Blockchain technology and decentralized governance: Is the state still necessary?" (2015).

Davidson S., P. De Filippi, and J. Potts, (2016).

Wright A. and P. De Filippi, (2015b).

Kim H. and M. Laskowski, arXiv preprint arXiv:1801.02029 (2018).

Atzei N., M. Bartoletti, and T. Cimoli, in *International Conference on Principles of Security and Trust* (Springer, 2017) pp. 164–186.

"Iso/tc 307," (Date last accessed: 12-April-2018).

Lin I.-C. and T.-C. Liao, IJ Network Security, **19**, 653 (2017).

Conti M., C. Lal, S. Ruj, *et al.*, arXiv preprint arXiv:1706.00916 (2017).

Crosby M., P. Pattanayak, S. Verma, and V. Kalyanaraman, Applied Innovation, **2**, 6 (2016).

Harvey C. R., (2016).

Meiklejohn S. and C. Orlandi, in *International Conference on Financial Cryptography and Data Security* (Springer, 2015) pp. 127–141.

Ben-Sasson E., A. Chiesa, E. Tromer, and M. Virza, in *International Cryptology Conference* (Springer, 2014) pp. 276–294.

Tasca P., S. Liu, and A. Hayes, The Journal of Risk Finance, 00 (2016).

Sasson E. B., A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, in *Security and Privacy (SP), 2014 IEEE Symposium on* (IEEE, 2014) pp. 459–474.

Wit J. d., (2017).

Hearn M., Corda Technical White Paper (2016).

Bitcoin Wiki, "Mixing Service," `https://en.bitcoin.it/wiki/Mixing_service` (2015), (Date last accessed: 01-Jan-2018).

Bitcoin wiki, "CoinJoin," `https://en.bitcoin.it/wiki/CoinJoin` (2015), (Date last accessed: 01-June-2017).

Noether S., A. Mackenzie, *et al.*, Ledger, **1**, 1 (2016).

Möser M. and R. Böhme, in *Security and Privacy Workshops (EuroS&PW), 2017 IEEE European Symposium on* (IEEE, 2017) pp. 32–41.

Rivest R., A. Shamir, and Y. Tauman, ASIACRYPT, Lecture Notes in Computer Science, **2248** (2001).

Guegan D., *Public Blockchain versus Private blockchain*, Tech. Rep. (Université Panthéon-Sorbonne (Paris 1), Centre d'Economie de la Sorbonne, 2017).

Maurer F. K., in *GI-Jahrestagung* (2016) pp. 2145–2150.

Reid F. and M. Harrigan, in *Security and privacy in social networks* (Springer, 2013) pp. 197–223.

Timmerman K., M. Thomas, *et al.*, Proctor, The, **37**, 26 (2017).

Möser M. and R. Böhme, in *International Conference on Financial Cryptography and Data Security* (Springer, 2015) pp. 19–33.

Carlsten M., *The Impact of Transaction Fees on Bitcoin Mining Strategies*, Ph.D. thesis, Princeton University (2016).

Guide, "Developer guide mining," `https://bitcoin.org/en/developer-guidemining` (2015b), (Date last accessed: 25-April-2017).

Alvseike R. and G. A. G. Iversen, *Blockchain and the future of money and finance: a qualitative exploratory study of blockchain technology and implications for the monetary and financial system*, Master's thesis (2017).

Scott B., *How can cryptocurrency and blockchain technology play a role in building social and solidarity finance?*, Tech. Rep. (UNRISD Working Paper, 2016).

Evans C., Journal of Islamic Banking and Finance, **3**, 1 (2015).

Quintana Diaz J. M., (2014).

Boudguiga A., N. Bouzerna, L. Granboulan, A. Olivereau, F. Quesnel, A. Roger, and R. Sirdey, in *IEEE Security & Privacy on the Blockchain (IEEE S&B 2017) an IEEE EuroS&P 2017 and Eurocrypt 2017 affiliated workshop* (2017).

Dorri A., S. S. Kanhere, and R. Jurdak, in *Proceedings of the Second International Conference on Internet-of-Things Design and Implementation* (ACM, 2017) pp. 173–178.

Andrews C., D. Broby, G. Paul, and I. Whitfield, (2017).

Mainelli M. and C. von Gunten, Long Finance (2014).

Insurance through Blockchain: A hybrid approach (2016).

Badzar A., (2016).

Hackius N. and M. Petersen, in *Proceedings of the Hamburg International Conference of Logistics (HICL)* (epubli, 2017) pp. 3–18.

Kotobi K. and S. G. Bilén, in *Wireless Telecommunications Symposium (WTS), 2017* (IEEE, 2017) pp. 1–6.

Plant L. *et al.*, Australian Journal of Telecommunications and the Digital Economy, **5**, 15 (2017).

Chakravorty A. and C. Rong, in *Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication* (ACM, 2017) p. 99.

Kuo T.-T., H.-E. Kim, and L. Ohno-Machado, Journal of the American Medical Informatics Association, **24**, 1211 (2017).

Appendix A: Blockchains Analysed for the Taxonomy

Technology	Description
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 cryptographic 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.
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.
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.
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.
2-3 Enigma	Distributed ledger technology, created by the MIT digital currency lab.

TABLE I Blockchain Technologies.