Cambridge
**Centre
for Alternative
Finance**

**UNIVERSITY OF
CAMBRIDGE**
Judge Business School

DISTRIBUTED LEDGER TECHNOLOGY SYSTEMS
A Conceptual Framework

Michel Rauchs   Andrew Glidden   Brian Gordon   Gina Pieters
Martino Recanatini   François Rostand   Kathryn Vagneur   Bryan Zhang

TABLE OF CONTENTS

FOREWORD

Terms such as cryptocurrency, blockchain, and distributed ledger technology (DLT) have gradually entered our daily lexicon, featured prominently in news and media, and fuelled discussion and debate among communities, industry practitioners and policymakers. Nevertheless, there is no rigorously defined set of terminologies or commonly acceptable taxonomy available. As a result, people are often talking past each other, and these terms are often misconstrued, misused, and misinterpreted.

Without undertaking a systematic and holistic approach, attention and analysis can be narrowly devoted to fractions, parts, and the surface of the phenomenon, rather than the whole. Consequently, people 'can't see the forest for the trees' and they are more susceptible to bias, misunderstanding, inflated claims, or conflicted views.

Therefore, a more thorough and reflective research to conceptualise and examine DLT as a functioning system with key layers, components, processes, and interactions with other systems (if applicable) is needed. By adopting a 'systems perspective', hopefully we can begin the journey to not only see 'trees' and the 'forest,' but to develop a more nuanced understanding of the complex and living 'DLT ecosystem'.

Building on the successes of our Centre's Global Cryptocurrency and Blockchain Benchmarking Studies - and aware of our own limitations and the challenges of the task - we reached out to assemble a team of researchers and contributors from diverse backgrounds. In order to conceptualise DLT systems and reach some form of 'consensus' on definitions and taxonomies, it is essential for our own research process to be open, collaborative, and self-critical. As we see it, this resulting study is a beginning and a catalyst to invite more input, discussion, and debate, as the landscape of DLT itself continues its swift evolution.

In this study, DLT systems were purposefully 'deconstructed' and then 'reconstructed' using a 'systems perspective' and an analytical framework that envisions all DLT systems as constructed of three layers: Protocol, Network, and Data. It articulates how these core layers interact with each other through processes and flows, as well as their conditional dependency and hierarchy within the system. The analysis demonstrates how varying the 'configuration' of these layers and their components will result in 'DLT systems' (and by extension the records and assets within them) that function and behave very differently. It also illustrates how DLT systems might interact with each other within the wider ecosystem, how 'centralisation' and 'decentralisation' should be understood as falling along a spectrum rather than binary, and the necessity for making a distinction between 'native' and 'non-native' recordkeeping.

We are very grateful for the contribution of all of our research team members and the opportunity to do our small part to further our collective understanding of DLT systems.

Bryan Zhang
Executive Director and Co-Founder,
Cambridge Centre for Alternative Finance

RESEARCH TEAM

PROJECT LEAD

Michel Rauchs: Michel is the Lead in Cryptocurrency and Blockchain at the Cambridge Centre for Alternative Finance. He co-authored two benchmarking reports that present an empirical analysis of the cryptocurrency and blockchain ecosystems.

✉ m.rauchs@jbs.cam.ac.uk   🐦 @mrauchs

CO-AUTHORS (ALPHABETICALLY-ORDERED)

Andrew Glidden: Andrew is the Head of Legal Research within Blockchain@BerkeleyLaw. His research interests include corporate governance and finance, financial regulation, and protocol design.

✉ asglidden@berkeley.edu   🐦 @asglidden

Brian Gordon: Brian is a Visiting Scholar at the David Eccles School of Business at the University of Utah and a Research Fellow at the University of California, Merced. His research focuses on strategy, entrepreneurship, and innovation.

✉ brianrgordon@gmail.com   🐦 @GordonBrianR

Gina Pieters: Gina is a Lecturer at the Department of Economics at the University of Chicago and a Research Fellow at the Cambridge Centre for Alternative Finance. Her research examines the economic implications and behaviour of cryptocurrencies across different currencies and monetary systems.

✉ gcpieters@uchicago.edu   🐦 @ProfPieters

Martino Recanatini: Martino is a Visiting Student at the Cambridge Centre for Alternative Finance and he is pursuing a Master's degree in Finance and Banking at the Politecnica delle Marche University. He is writing a master's thesis that addresses the impact of DLT systems on securities post-trading services.

✉ m.recanatini@jbs.cam.ac.uk   🐦 @marecanatini

François Rostand: François is a Visiting Student at the Cambridge Centre for Alternative Finance and is pursuing a Master of Philosophy in Chemical Engineering at the University of Cambridge.

✉ fr339@cam.ac.uk

Kathryn Vagneur: Kathryn is a Research Affiliate at the Cambridge Centre for Alternative Finance. Her work has explored governance, management control, resilience and vulnerabilities in regulatory systems and blockchain projects.

✉ kvagneur.phd91@london.edu

Bryan Zhang: Bryan is the Executive Director and a Co-Founder of the Cambridge Centre for Alternative Finance. He has co-authored more than 15 reports on alternative finance.

✉ b.zhang@jbs.cam.ac.uk   🐦 @BryanZhangZ

CONTRIBUTORS

Oliver Beige: Oliver is an industrial engineer and economist (PhD UC Berkeley) who applies innovation economics to early technology R&D. He has worked for companies including SAP and Mercedes-Benz, and now consults on the intersection between blockchain and enterprise systems.

✉ beige@cal.berkeley.edu   🐦 @ecoinomia

Jill Carlson: Jill works as an advisor to and investor in cryptocurrency protocol projects. She has also consulted on the topic for institutions including the IMF.

✉ jillruthcarlson@gmail.com   🐦 @_jillruth

Nic Carter: Nic is a partner at Castle Island Ventures, a Boston-based blockchain-focused venture fund, and the cofounder of Coinmetrics.io, an open data repository for public blockchains. He wrote a master's thesis on cryptoasset governance structures at the University of Edinburgh.

✉ nic@castleisland.vc   🐦 @nic__carter

Michèle Finck: Michèle is a Senior Research Fellow at the Max Planck Institute for Innovation and Competition and a Lecturer in EU law at Keble College, University of Oxford. She is the author of 'Blockchain Regulation and Governance in Europe' (Cambridge University Press 2018) and is advising a number of institutions on the intersection between law and blockchain technology.

✉ michele.finck@ip.mpg.de   🐦 @finck_m

Larry Sukernik: Larry works at Digital Currency Group (DCG) investing in startups and digital assets. Previously, Larry worked at Ernst & Young, where he spent 3 years developing the cryptocurrency and blockchain practice.

✉ larry@dcg.co   🐦 @lsukernik

Angela Walch: Angela is an Associate Professor at St. Mary's University School of Law and a Research Fellow at the UCL Centre for Blockchain Technologies. Her work on blockchain technologies has focused on governance, language, and operational risks.

✉ awalch@stmarytx.edu   🐦 @angela_walch

DISCLOSURES

ACKNOWLEDGEMENTS

EXECUTIVE SUMMARY

The DLT ecosystem is plagued with the use of incomplete and inconsistent definitions and a lack of standardised terminology, creating a needlessly complicated landscape for everyone from experienced policymakers and developers to individuals venturing into the field for the first time. This study sets out to contribute to international discussions to create a shared, common language around DLT systems to clarify terminology and concepts. We provide a formal definition of DLT systems and a list of key characteristics that distinguish them from alternative systems.

We then introduce a conceptual framework that serves as a multi-dimensional tool for examining and comparing existing DLT systems, which we believe will be useful for a wide range of readers and purposes: from businesses and institutions developing DLT-based applications, investors funding DLT ventures, to academics, regulators and policymakers who wish to have a better and more nuanced understanding of DLT systems. The framework breaks down a DLT system into a set of interconnected layers, components, and processes. It is the combination and interactions of these small and rather simple processes that together form a complex and dynamic system.

We show that layers follow a hierarchy: the protocol layer dominates both the network layer and the data layer in that it can overrule any decisions taken at those layers. Typical roles and actors within a DLT system are grouped together into four categories. We discuss how roles and actors can be distributed across layers, which becomes crucial when examining the power structure around the system. We highlight that in DLT systems decentralisation is not a binary property, but a continuous variable resulting from interplay of the system components, hierarchies, and power structures at each layer.

Our framework presents a non-exhaustive list of potential configurations for various processes at each layer and component. It then shows how different design choices (i.e. different configurations) lead to particular outcomes that shape the properties and characteristics of the system. This exercise requires the application of different lenses developed in the framework for analysing each process. It also shows that trade-offs are inherent to DLT systems and move along a spectrum according to specific security assumptions, threat models, and trust relationships. There is no inherent 'right' or 'wrong': use case requirements and objectives should drive the discussion around acceptable trade-offs to choose from.

We further note that DLT systems generally do not operate in isolation, but in concert with a variety of external systems: only transfers of endogenous resources internal to the system are automatically executed by the DLT system itself without the involvement of external agents. This is particularly relevant when the records produced by the DLT system reference exogenous objects, events, or facts external to the DLT system in question (e.g. items tracked in a supply chain; physical assets

held in custody). These exogenous objects require gateways that connect the system to the external world and are reliant on external agents and an existing legal structure to enforce decisions outside of the boundaries of the DLT system.

The study also demonstrates that choices made in the design, architecture, and governance of each DLT system can result in significant differences with regard to system properties and characteristics. We discuss the concept of provisional settlement and illustrate the life cycle of transactions within different architectures. In addition, we explore how record producers are incentivised by distinguishing between systems whose security model relies on intrinsic economic incentives (i.e. requiring a native asset for compensation) and systems that are secured through access controls and contractual obligations between record producers.

We clear up misconceptions about the form taken by the shared data record structure and classify the data as transactions, logs, records, journals, and ledgers according to the extent the data has been processed by the DLT system network. Importantly, we use the term 'ledger' to mean the set of records which are held in common by a substantial proportion of network participants.

Finally, we conduct a comparative analysis by applying the conceptual framework to six case studies (Bitcoin, Ethereum, Ripple, Alastria, Verified.me, and an anonymised DLT system referred to as "Project X") and introduce a DLT systems landscape map that positions a dozen of DLT systems. We observe that open systems with permissionless participation in transaction processing primarily record transfers of ownership of endogenous resources, whereas the majority of closed systems with more fine-grained permission levels typically reference objects external to the system and depend on gateways and external enforcement. We demonstrate that open systems range from fully centralised to reasonably decentralised as a whole, while closed systems currently tend to be centralised for a variety of reasons, with plans to gradually distribute control over time.

SECTION 1: INTRODUCTION

RATIONALE

The concept of distributed ledger technology (DLT) existed before Bitcoin and blockchain technology. *The Byzantine Generals Problem* theorised by Lamport et al. (1982) described how 'computer systems must handle [...] conflicting information' in an adversarial environment.[1] Subsequent research led to the emergence of the first algorithm for 'highly available systems that tolerate Byzantine faults' with little increase in latency (Castro & Liskov, 2002).[2] The earliest identified occurrences of the concept of a 'blockchain' can be traced back to Haber & Stornetta (1991)[3] and Bayer et al. (1992)[4] who introduced the notion of a chain of cryptographically-linked data blocks to efficiently and securely timestamp digital data in distributed systems using cryptographic hashing functions and Merkle trees.

However, these developments attracted little attention in contrast with recent enthusiasm around cryptocurrencies and blockchain technologies more generally. This new interest has attracted significant investment, resulting in the rapid evolution of DLT system types and applications, many of which have little in common with Bitcoin and its numerous copycats.

DLT systems conceptually emerged in 1982, while the earliest occurance of the 'blockchain' concept can be traced back to 1991

What Is DLT?

Distributed ledger technology (DLT) has established itself as an umbrella term to designate multi-party systems that operate in an environment with no central operator or authority, despite parties who may be unreliable or malicious ('adversarial environment'). Blockchain technology is often considered a specific subset of the broader DLT universe that uses a particular data structure consisting of a chain of hash-linked blocks of data.

Concomitant with the expansion and evolution in types and uses of DLT has been the widespread use of language and terminology which is frequently fuzzy, imprecise, and inconsistent across different projects. This report was motivated by a recognition that, left unsolved, this disorderly use of language and conceptual terminology could hinder development within the DLT sector, and may present society and industry with legal uncertainty and financial risks which are as yet unrecognised.

Currently, much of the general interest is focused on cryptographically-secured digital assets and other digital tokens that can be issued and transferred on DLT systems. Before the properties of these assets can be analysed, however, it is critical to have a robust understanding of the underlying infrastructure, and how specific design decisions impact the nature of the recorded data.

OBJECTIVES

This report seeks to establish a conceptual framework and terminology that can be applied with ease across DLT systems that predate cryptocurrencies such as Bitcoin, and the many DLT systems which have been inspired by or followed Bitcoin. It also seeks to distinguish these newer technologies from 'traditional' databases and other systems. The purpose of the framework is to provide a multidimensional tool for examining and comparing existing DLT systems and their

traits and features. It also can serve as an analytical tool useful when examining proposals for new DLT systems.

This framework for DLT analysis has been designed to be *generic* so it should be applicable to every type of DLT, and *modular*, so that new layers, components, processes, and configurations can be added independently without affecting the core of the framework.

METHODOLOGY

We have taken a 'systems perspective' because it allows us to describe how a collection of parts work together to create a functional whole rather than presenting them as a set of disconnected parts. This enables assessment of the behaviors of such a system in the context of its environment. While the concept of a system itself is a more general notion that indicates separation of some part of the universe from the rest, the idea of a systems perspective is to use a non-

reductionist approach to the task of describing the properties of the system itself. Further, we have sought to consider these systems in the context of their environments or ecosystems, and not as isolated entities. Thus, one can examine the interactions and relationships between a DLT system and its environment.

This analytical approach draws from *Systems Theory* which has developed in parallel streams of research, each with its own unique

orientation, that began to emerge in the 1940s with the first published work by Ludwig von Bertalanffy (1949). He articulated the notion of a *general systems theory*[5], which is multidisciplinary in nature and examines the general science of 'wholeness' as systems. Ervin Laszlo (1972) proposed an organisation of knowledge in terms of systems, systemic properties and inter-system relationships which he termed *'systems philosophy'*.[6] Walter Buckley (1967)[7] and James Grier Miller (1978)[8] further refined Bertalanffy's general systems theory as a theoretical framework and methodology that can be applied in physical, biological as well as social sciences. Especially notable was Miller's concept of 'living systems' which stipulates that systems can have hierarchical levels and subsystem layers, maintained by flows of information, energy and matter.

We aim to conceptualise a DLT system in this vein as a set of interconnected and hierarchical components and their interacting processes. Rather than a simple collection of parts, it is the 'configuration' of hierarchical components - and their interrelations and interactions - that determines the functionality and characteristics of a particular DLT system.

REPORT STRUCTURE

The remainder of the report is structured as follows:

Section 2 provides a review of the existing literature, summarises theoretical concepts and frameworks, and outlines their limitations. It then establishes a formal definition of a DLT system and highlights the necessary criteria it must meet, and defines several key terms.

Section 3 presents a high-level overview of the proposed tool by introducing the various elements of the conceptual framework.

Section 4 investigates the dependencies between layers within a particular DLT system as well as the interactions and relationships with external systems.

Section 5 offers a deep dive into each element of the conceptual framework by outlining potential configurations and their effects on the system illustrated by examples of existing DLT systems.

Section 6 applies the framework as a tool to Bitcoin and compares it to other case studies that have chosen alternative design decisions.

Section 7 summarises the present report and offers recommendations as to how the conceptual framework might be extended and what it can be applied to.

Appendix A presents the full framework in table form; Appendix B summarises the comparative analysis between the six case studies (Bitcoin, Ethereum, Ripple, Alastria, Verified.Me, and 'Project X'[9]); and Appendix C features a glossary of the most commonly used terms.

SECTION 2: DLT SYSTEMS - SETTING THE SCENE

2.1 DLT SYSTEMS IN THE LITERATURE

2.1.1 Definitions

There exist many different definitions of distributed ledger technology (DLT) systems in the literature, and many publications on the subject set out their own unique definition in their preamble. Some definitions are narrow, while others are very broad; some are contradictory. Consequently, a coherent definition for DLT has not yet developed.

For instance, the World Bank (2017) describes DLT systems as 'a specific implementation of the broader category of 'shared ledgers', which are simply defined as a shared record of data across different parties'[10].

Pinna & Ruttenberg (2016) from the European Central Bank (ECB) describe DLT as a technology that 'allow[s] their users to store and access information relating to a given set of assets and their holders in a shared database of either transactions or account balances. This information is distributed among users, who could then use it to settle their transfers of, e.g. securities and cash, without needing to rely on a trusted central validation system'[11]. Davidson et al. (2016) consider a DLT system a 'distributed,

cryptographically secure, and crypto-economically incentivised consensus engine'[12].

In contrast, the Bank of England (2017) provides a set of key architectural characteristics that define DLT systems: 'A DLT is a distributed database, in the sense that each node has a synchronized copy of the data, but departs from the traditional distributed database architectures in three important ways: (i) decentralisation; (ii) reliability in trust-less environments; (iii) cryptographic encryption'. The Bank of England summarises its definition as: 'a database architecture which enables the keeping and sharing of records in a distributed and decentralised way, while ensuring its integrity through the use of consensus-based validation protocols and cryptographic signatures'[13].

Similarly, Tasca & Tessone (2018) list a set of key features that seem unique to DLT systems: 'A DLT system is a community consensus-based distributed ledger where the storage of data is not based on chains of blocks whose principles are (a.) decentralisation of consensus, (b.) transparency, (c.) security and immutability'[14].

Other definitions refer exclusively to 'blockchain technology' and do not differentiate between DLT and 'blockchain'. For instance, Cong & He (2018) define a blockchain as a 'distributed database that autonomously maintains a continuously growing list of public records in unit of 'blocks', secured from tampering and revision'[15], while Atzori (2015) describes it as an 'irreversible and tamper-proof public records repository for documents, contracts, properties, and assets [that] can be used to embed information and instructions, with a wide range of applications'[16].

As shown by these examples, there is no genuine and universal definition for what is referred to as a DLT system. Adding to the challenge is that on the one hand, definitions are sometimes too specific, technical and inaccessible to general audiences; while on the other hand, some are too simplistic and broad so that no meaningful difference to more traditional database architectures can be observed. Either way, a lack of common terminology has resulted in misconceptions and the widespread formation of unrealistic expectations as to what this technology can achieve.

2.1.2 Existing Frameworks

Ontologies - descriptions of things that exist, and how they can be grouped together according to similarities and differences - allow people to converge towards a common terminology in specific ecosystems. Therefore, the project team has carried out an analysis of ontologies previously proposed to understand the suggested categorisations of DLT ecosystems provided by academics, professionals and others who have written on this topic. We summarise some of

these frameworks below and discuss their shortcomings in Section 2.1.3.

Okada et al. (2017) proposes a classification of blockchain based on two dimensions: a) the existence of an authority and b) the incentive to participate.[17]

Lemieux (2017) analyses blockchains through the lens of archival science, the theory underpinning record keeping and preservation of authentic records. This work frames blockchains in terms of types of record keeping systems, namely 'mirror type', 'digital record type', and 'tokenised type', and examines each type in relation to a formal archival theoretic evaluation framework.[18]

Platt (2017) presents a simple yet powerful two-dimensional framework that categorises DLT systems according to (a) their data diffusion model (global vs. local) and (b) on-chain functionality (stateful vs. stateless).[19]

De Kruijff & Weigand (2017) attempt a solution to the lack of formalisation in the enterprise blockchain literature. Kruijff uses an enterprise ontology to distinguish between the datalogical, infological, and essential layer levels of blockchain transactions and smart-contracts.[20]

Xu et al. (2017) have develop a 'layer approach' to the current framework. This work aims to assess the impact of the blockchain design decisions on the software architecture. The proposed taxonomy is intended to help with architectural (software) considerations about the performance and quality of blockchain-based systems.[21]

Glaser (2017) uses a clear terminology, contributing to a common basis for

communication and connects the terminology to digital market models in order to determine every component's market implication[22]. His idea was also based on considerations arising in Glaser & Bezzenberger (2015) that aims to provide an early tool for classifying peer-to-peer transfer systems and decentralised consensus systems.[23]

Lastly, Tasca & Tessone (2018) attempted to add an overall perspective of DLT systems on top of previous definitions. This advanced ontology is quite comprehensive and detailed for the classification of blockchain technologies.[24]

2.1.3 Limitations Of Prior Work

There has been a plurality of definitions proposed for distributed ledger technologies, each varying in detail, which make it difficult to extrapolate from specific definitions into a general and modular framework capable of describing and classifying different types of DLT systems.

Debate is further hampered by a lack of attention to the definitional clarity of DLT system components in prior works. For example, decentralisation is often treated as a binary feature of DLT systems, instead of a continuous variable resulting from the interplay of the various layers and nested subsystems within them. This is partially due to examples in the current literature which do not break down the system into different components and examine the relationships, dependencies, and interactions between these different elements.

In order to overcome these limitations, this study aims to provide a working definition of DLT systems and takes a holistic approach, building up from the process level to develop a generic and durable tool. The resulting conceptual framework can be used for various purposes, including the assessment of an existing system, a comparative analysis of multiple systems, and the development of new systems.

2.2 WHAT ARE DLT SYSTEMS?

Section 2.1 has highlighted the multitude of conflicting definitions as to what constitutes a 'blockchain' or a 'distributed ledger'. Unclear terminology and fuzzy boundaries have resulted in 'DLT' evolving into into an umbrella term used to designate a variety of loosely related concepts (which include, among others, blockchains).

One interpretation of the DLT concept is its most narrow (and historically-grounded) definition: an append-only chain of cryptographically-linked 'blocks' of data,

maintained and updated by a decentralised network, with network nodes encouraged by economic incentives to engage non-strategically[25] to maintain and secure the system so that the data - organised in a specific structure often referred to as 'global ledger' - is robust to adversarial interference, double-spend, censure, counterfeit, collusion, tampering, or other types of malicious actions.

Such a narrow definition, however, excludes many existing and potential future applications of distributed ledger technologies. It also

excludes cases where an enterprise applies the term DLT in a context which is so broad that the line between it and more traditional distributed systems becomes blurred and many of the core elements of the narrow definition are missing or degraded.

In order to resolve this issue, we propose to balance the two ends of the spectrum by taking an alternative approach that focuses on the essential minimum requirements of a DLT system (i.e. the necessary and sufficient conditions), as opposed to articulating the full set of properties that a DLT system might ideally possess. We consider DLT systems as a type or subset of distributed systems, which exhibit a set of specific characteristics that distinguishes them from more traditional distributed systems.

DLT systems are designed to be capable of operating in an adversarial environment

What Is An Adversarial Environment?

An adversarial environment is characterised by the presence of malicious actors within a system or network, who undermine the system by using it in ways it was not intended for. The prototypical adversary in a DLT system is an entity that attempts to exploit the consensus rules to transfer assets without authorisation, censor others' transactions, or otherwise disrupt the network. Adversaries may operate inside or outside the system.[26]

In essence, a DLT system is a 'consensus machine': a multi-party system in which participants reach agreement over a set of shared data and its validity, in the absence of a central coordinator. What separates DLT systems from traditional distributed databases are features rooted in designs capable of supporting data and maintaining data integrity in an adversarial environment.

DLT systems are multi-party 'consensus machines'

DLT systems can tolerate, within limits, the presence of both malicious actors actively attempting to attack the system and unreliable-yet-honest actors.[27] This tolerance extends only to the recording and processing of data; parties who wish to transact together may be able to rely on the performance of the system, but must still generally trust their counterparties.[28] For this reason, DLT systems can be characterised as a disintermediating technology that 'delegates trust to the endpoints' (i.e. the end users) of the system.[29]

These characteristics are dependent on, and specific to, the architecture and design of the system, as well as its operating environment;

these are not the result of some 'natural law' or 'immutable requirement'. Similarly, tolerance to adversaries does not imply that all DLT systems necessarily operate in adversarial environments, or that they provide an invincible defense against adversarial attack.[30]

Figure 1 offers an illustration of the fundamental differences between a traditional database system operated by a single entity, a traditional distributed database and a distributed ledger system. While each system takes inputs from various sources, control over how data is stored, processed, and executed varies from one type to another.

Figure 1: From Centralised Databases To Distributed Ledgers

	1 INPUT	**2 STORAGE**	**3 PROCESSING**	**4 OUTPUT**	**CONTROL**
Centralised Database	Multiple nodes	Single node	Single node	Single node	Single entity
Traditional Distributed Database	Multiple nodes	Multiple nodes	Multiple nodes	Multiple nodes	Single entity
Distributed Ledger Technology	Multiple nodes	Multiple nodes	Multiple nodes	Multiple nodes	Multiple entity

Single entity control   Multiple entity control   Single node   Multiple nodes

Note: a traditional distributed database consists of multiple nodes that collectively store and process data, however, the nodes are generally controlled by the same entity as opposed to DLT systems where there are multiple controllers.

A DLT system is a system of electronic records that enables independent entities to establish a consensus around a shared 'ledger' - without relying on a central coordinator to provide the authoritative version of the records

A DLT system needs to be **capable of ensuring** the following properties, either in the existing system or with minimal changes to the system.

a. **Shared recordkeeping:** enable multiple parties to collectively create, maintain, and update a shared set of authoritative[31] records (*the 'ledger'*).

b. **Multi-party consensus:** enable all parties to come to agreement on a shared set of records

   i. If permissionless, without relying on a single party or side-agreements, and in the absence of ex ante trusted relationships between parties; and

   ii. If permissioned, through multiple record producers who have been approved and bound by some form of contract or other agreement.

c. **Independent validation:** enable each participant to independently verify the state of their transactions and integrity of the system.

d. **Tamper evidence:** allow each participant to detect non-consensual changes applied to records trivially.

e. **Tamper resistance:** make it hard for a single party to unilaterally change past records (i.e. transaction history).

We therefore propose the following **formal definition:**

A DLT system is a system of electronic records that

   i. enables a network of independent participants to establish a consensus around

   ii. the authoritative ordering of cryptographically-validated ('signed') transactions.[32] These records are made

   iii. persistent[34] by replicating the data across multiple nodes,[35] and

   iv. tamper-evident[36] by linking them by cryptographic hashes.[36]

   v. The shared result of the reconciliation/consensus process - the 'ledger' - serves as the authoritative version for these records.[37]

The goal of a DLT system is thus to produce a set of authoritative records that are validated and executed via a multi-party consensus process that involves the participation of multiple separate entities - all in the absence of a central authority. Users create and broadcast unconfirmed transactions (i.e. proposals to make a new ledger entry), which get bundled together into records by record producers, and added to the ledger. The instructions contained in the now-confirmed transactions are then automatically executed by all auditors.

2.3 CLARIFYING TERMINOLOGY

The 'Ledger' Concept

There is significant overlap and similarity among many of the terms used to describe the components of DLT systems. This often results in ambiguous or conflicting use of terminology. Consider, for example, the term 'ledger'. Not only does the DLT system literature assign 'ledger' a different meaning from the one used in disciplines like accounting or finance, but the

DLT literature itself uses the term to describe two very different ideas: (i) the set of data held by an individual network node, and (ii) the set of data held in common by the majority of nodes.

Within this project we define the terms log, journal, record and ledger[38] according to the extent transaction data has been accepted, processed, and validated by the network as a whole (**Figure 2**).

Key Concepts

- **Transaction:** any proposed change to the ledger; despite the connotation, a transaction need not be economic (value-transferring) in nature.

- **Log:** an unordered set of valid[39] transactions held by a node, which have not yet been incorporated into a formal record subject to network consensus rules (i.e. 'unconfirmed' transactions).

- **Record:** transaction data which has been subject to network consensus rules.
  Note: A 'candidate record' is a record that has not yet been propagated to the network.

- **Journal:** the set of records held by a node, although not necessarily consistent with the consensus of other nodes. Journals are partial, provisional, and heterogeneous: they may or may not contain all the same records.

- **Ledger:** the authoritative set of records collectively held by a significant proportion of network participants at any point in time, such that records are unlikely to be erased or amended (i.e. 'final').[40]
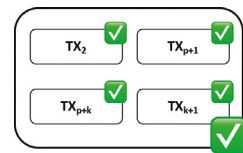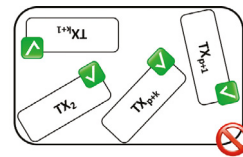
Using Bitcoin as an example, a *transaction* can be a transfer of an asset from one address to another; a node's *log* is its mempool (i.e. the collection of unconfirmed transactions the local node has received from connected nodes, which have not yet been processed into records);[41] a *record* would be a confirmed block; a node's *journal* is its individual, locally-stored copy of the blockchain,[42] which may be incomplete or contain data unknown to the rest of the network; and the *ledger* would be the authoritative set of blocks which are, by consensus, considered 'final' – i.e. which have a vanishingly low probability of being overwritten by a more-worked subchain.[43]

Figure 2: From Transactions To Records

Globally agreed-upon authoritative set of records.

Set of records held by each individual node, although not necessarily consistent with the consensus of other nodes.

Transaction data arbitrarily selected by Record Producers and broadcast to the network to be subject to consensus rules.

Set of unconfirmed transactions held by each individual node: transactions are not yet incorporated into a formal record subject to network consensus rules.

Proposed changes to the ledger (i.e. ledger entries).

LEDGER
JOURNAL
RECORDS
LOG
UNCONFIRMED TRANSACTIONS
PROCESSING

① UNCONFIRMED TRANSACTIONS	② LOG (MEMPOOL)	③ RECORD	④ JOURNAL	⑤ LEDGER
End users create transactions and broadcast them to the network though various means. These transactions are waiting for confirmation.	Each fully-validating node stores unconfirmed transactions in its log ('mempool'). Logs may differ from one node to another.	Each record producer now arbitrarily selects a set of unconfirmed transactions from its log and creates a candidate record. After performing the necessary steps specified by the protocol to make the candidate record valid, it broadcasts the record to connected nodes.	Each node will verify the received candidate record: if it complies with protocol rules, the node will add the record to its own instance of the ledger - the journal. Journal states may differ from one node to another.	The ledger represents the globally agreed-upon authoritative set of records that constitutes the state of the system. It results from the convergence of synchronised individual journals.
End users (via nodes, SPV, API)	Individual node level	Individual node level	Individual node level	Global system level

PROCESSING

Each node in the network has its own, potentially imperfect 'copy' of the ledger (i.e. a journal). This means not only that some of the data held by the node is provisional and partial, but that it may not always reflect the complete set of structured, authoritative records as determined by the consensus mechanism set out by the protocol.

The goal of a DLT system is to keep these individual instances (journals) of the structured record in sync, leading to a convergence towards a single accepted set of authoritative records (the ledger). This enables a group of separate parties that do not necessarily trust each other to reach agreement over a shared set of data without having to rely on a central authority. Conceptually, the 'ledger' should be regarded as a latent, abstract construct that is generated by the DLT system as whole through the constant efforts of synchronising the individual copies maintained by each full participant (**Figure 3**).[44]

The core of all DLT systems is the organisation and processing of shared data resulting in the ledger. A functional DLT system creates and maintains a ledger in spite of unreliable participants or adversaries.

Figure 3: Depicting The 'Ledger' Concept
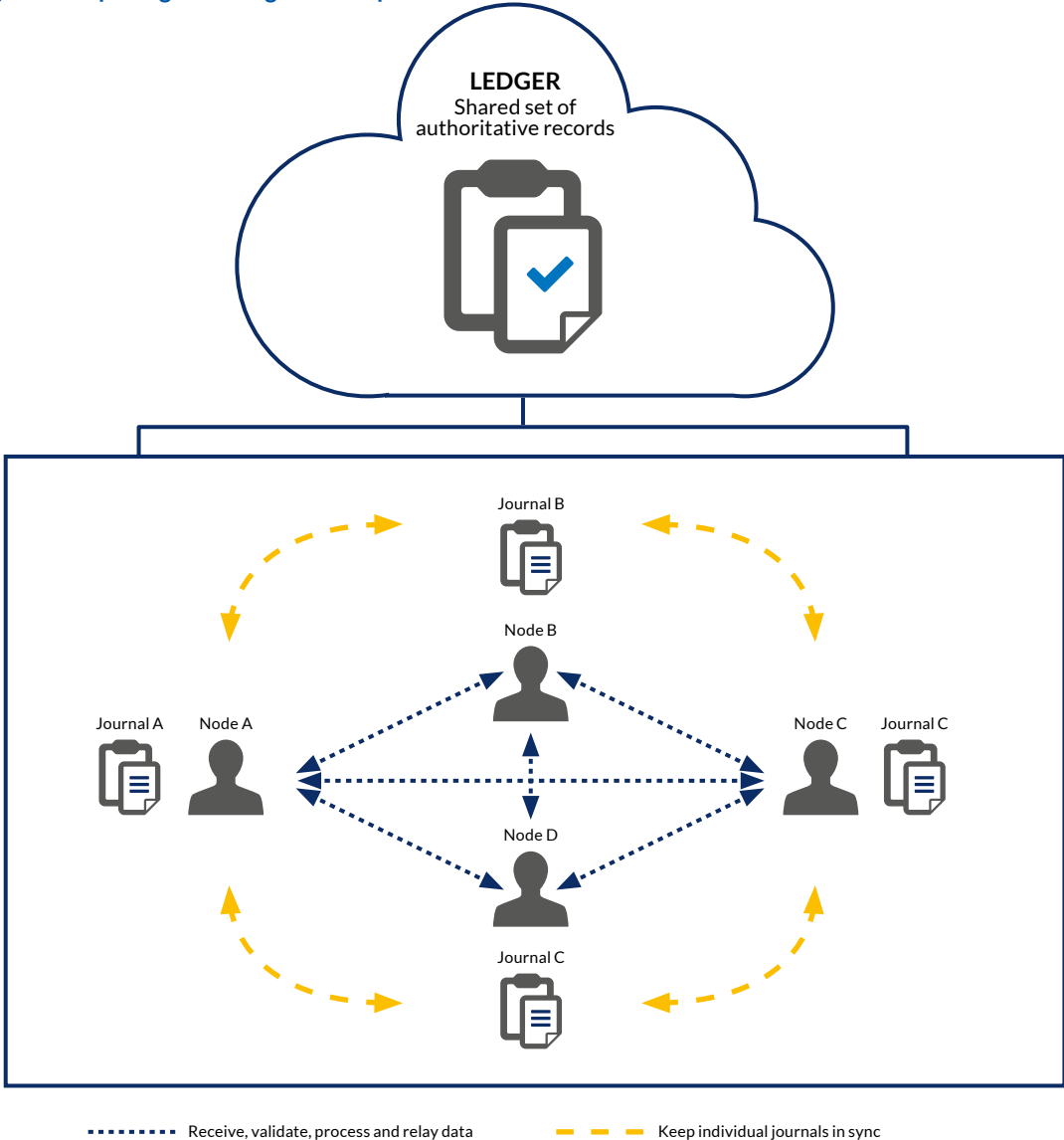
LEDGER
Shared set of authoritative records

Journal B

Node B

Journal A   Node A

Node C   Journal C

Node D

Journal C

·········· Receive, validate, process and relay data   – – – Keep individual journals in sync

The 'Private Key' Concept

Users create transactions - or, technically speaking, state transitions in the form of ledger entries - by putting raw data into a standardised format, adding a cryptographic signature to the transaction for authentication purposes, and then broadcasting it to other nodes in the network. The signature, produced by a *private key*, represents the users' permission for the DLT system to request a ledger entry reflecting the transaction.[45] A valid signature provides the cryptographic assurance to the DLT system that the transaction initiator has the authorisation to enact a corresponding ledger entry.

Private keys can be stolen if not properly secured, allowing the thieves to engage in transactions

indistinguishable from those of the true owner

It is important to note that a valid signature does not automatically provide proof that the *owner* of the corresponding private key has produced the signature. Instead, it provides a guarantee that a *holder* of the private key has initiated the transaction. The use of a private key provides a strong presumption that a transaction was authorised. However, private keys can be stolen by attackers if they are not properly secured. Storing private keys securely can be a cumbersome task; key management is notoriously difficult and requires a certain level of technical proficiency, which is why it is often outsourced to third-party custodial services.[46]

2.4 ACTORS

A DLT system is composed of actors that perform various roles. In this context, an **actor** is any entity or individual that is either directly or indirectly interacting with a DLT system. Actors can be grouped together into four key categories according to the role they play in the system (**Figure 4**).

One entity can take the roles of multiple actors simultaneously and operate on more than one layer. Similarly, a specific role can be performed by multiple actors at the same time.

DEVELOPERS

Core Protocol

Client

Application

External Systems

ADMINISTRATORS

Foundation

Company

Consortia

Open-Source Community

GATEWAYS

Gatekeeper

Oracle

Custodian

Issuer

Exchange

PARTICIPANTS

Auditor

Record Producer

Lightweight Client

End-User
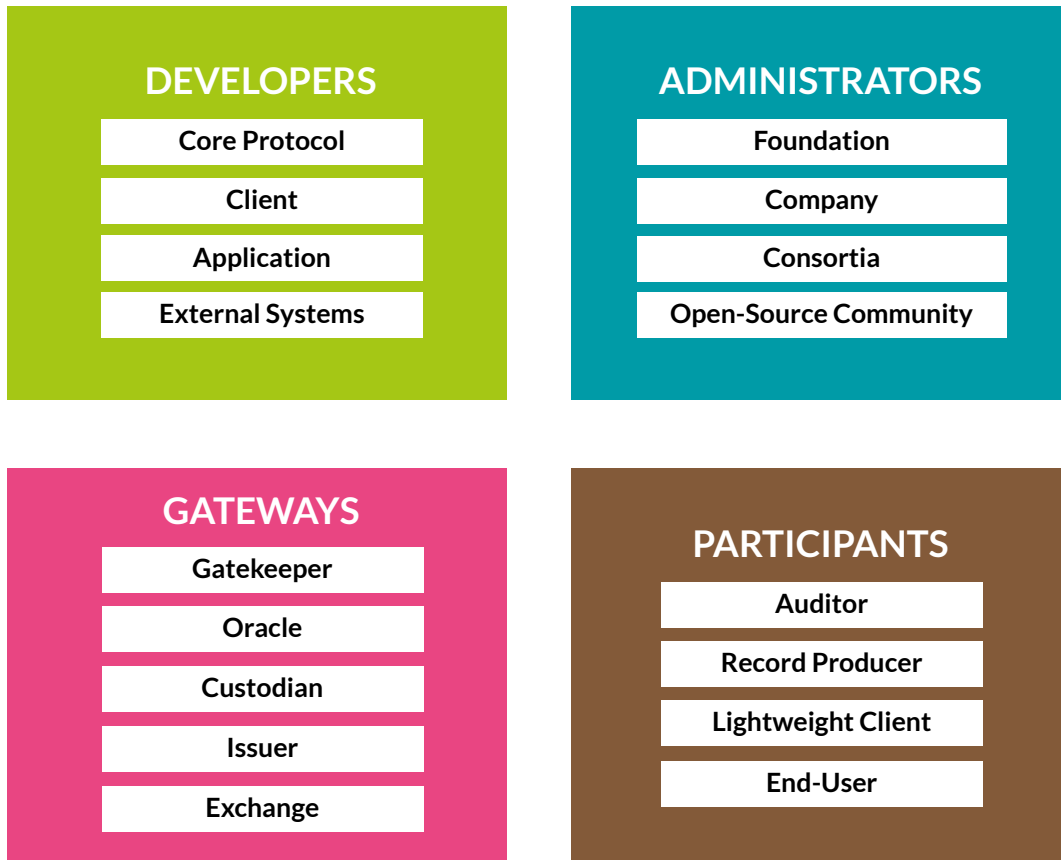
2.4.1 Developers

Developers write and review code that underlies the technological building blocks of a DLT system and its connected system(s). Developers may be professionally employed or participating as volunteer contributors.

- **Protocol:** maintaining the core protocol codebase (or an alternative implementation).
- **Client:** building the DLT client[47] that provides an interface to the DLT system.
- **Application:** designing applications that run on top of the DLT system platform.
- **External systems:** creating infrastructure to enable protocols to function or interact with each other.

Checks And Balances

Ideally, a system of checks and balances should arise from the composition of actors and roles that ensures that no single party or cartel can take over the system unilaterally. This in turns ensures the tamper-resistant characteristic of DLT systems.

2.4.2 Administrators

Administrators control access to the core codebase repository and can decide to add, remove and amend code to change system rules. Administrators are often considerably involved in the governance process and may have absolute control over it.

The nature and role of an administrator can vary greatly from one system to another. For instance, closed and permissioned DLT systems may have a dedicated entity taking the role of administrator, whereas open, permissionless systems often have a loosely connected set of 'administrators' in the form of volunteer core developers rather than a formal administrator. In the latter case, these developers do not actually directly control the codebase; rather, they propose changes which are 'ratified' by users independently (by choosing to incorporate the proposals in the software they run).

2.4.3 Gateways

Gateways provide interfaces to the system by acting as a bridge between the system and the external world.

- **Gatekeeper(s):** granting participants access to the system.
- **Oracles:** transmitting external data to the system.
- **Custodians:** holding assets in custody.
- **Exchanges:** facilitating purchase/sale of digital assets.
- **Issuers:** issuing or redeeming tokens representing the assets recorded in the system.

2.4.4 Participants

The network consists of interconnected participants that communicate by passing messages among each other.

- **Auditors:** checking submitted transactions and records for validity, reporting invalid records to the network, and relaying valid transactions and records. Ability to perform an independent audit of the system state. Often called *full/fully-validating nodes.*[48]
- **Record Producers:** producing and submitting sets of candidate records for potential inclusion into the ledger. Often called *miners* or *validators.*[49]
- **Lightweight Clients:** querying auditors for data regarding specific transactions; do not fully validate the system.
- **End-users:** indirect users of the system who require a gateway to access the system (e.g. custodial wallet service).

Actors in a DLT system can take multiple roles and operate on more than one system layer. For instance, an entity can take multiple roles just as one role can be performed by multiple entities. Every DLT system has a different composition of actors, roles and entities; the distribution and repartition of roles across layers, components, and processes shapes the properties of the system.

SECTION 3: INTRODUCING THE FRAMEWORK

This section begins to examine the necessary and sufficient elements which comprise a DLT system. The aim is to provide flexibility in the analysis and classification of DLT systems.

As shown in **Figure 5**, a DLT system can be divided into three interdependent **core layers**:

1. **Protocol:** set of software-defined rules that determine how the system operates

2. **Network:** interconnected actors and processes that implement the protocol

3. **Data:** information flowing through the system that carries a specific meaning in relationship to the design and functions the system is intended to play for users

Layers → Components → Processes

Figure 5: DLT System Anatomy

Protocol Layer				
	Genesis Component	Inter-System Dependencies	Codebase Creation	Rule Initiation
	Alteration Component	Protocol Governance	Protocol Change	

Network Layer				
	Communications Component	Network Access	Data Broadcast	Transaction Initiation
	Transaction Processing Component	Record Proposal	Conflict Resolution Rule	Incentivised Transaction Processing
	Validation Component	Transaction Validation	Record Validation	Transaction Finality

Data Layer				
	Operations Component	Input	Programmatically-executed Transactions	Locus of Execution
	Journal Component	Reference		

Each **layer** is composed of one or more components involved in the creation or operation of a DLT system. A **component** is a logical set of related processes necessary for the functioning of the system. A **process** is a series of actions carried out by **actors** to achieve a specific objective or series of objectives involved in the successful operation of a component.

The full framework can be found in table format in **Appendix A**.

3.1 PROTOCOL LAYER

The protocol layer is the foundation of the entire DLT system: it defines the set of formal rules that governs the system and codifies its architectural design. The protocol can be considered a set of 'constitutional' arrangements agreed upon by all system participants. The protocol contains two components:

Figure 6: Protocol Layer

PROTOCOL LAYER

Genesis Component
- How the system is linked to other external systems?
- How is the protocol generated?
- Where are rules defined?

Alteration Component
- How are decision-making and implementation processes created?
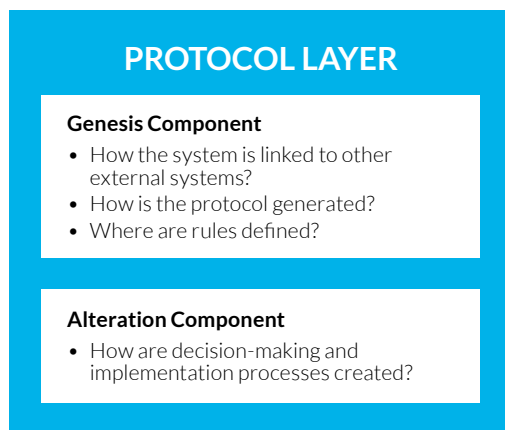
Genesis Component:
Defines the processes of the DLT system at the time of network launch. It consists of the initial codebase and architecture specifying the rules of engagement within the system, including the first ('genesis') record.

Alteration Component:
Sets out how the protocol evolves over time. It includes a governance aspect (i.e. how collective decisions are made) as well as an implementation consideration (i.e. how the result of those decisions are incorporated). The alteration component need not be an explicit part of the protocol; indeed, most DLT systems move governance and related issues 'off-chain'.[50]

'On-chain' Versus 'Off-chain'

The term '**off-chain**' refers to anything that occurs outside of the formal boundaries of a DLT system. This is opposite of '**on-chain**' which refers to anything that occurs within the boundaries of the DLT system.

3.2 NETWORK LAYER

The network layer is comprised of interconnected actors that collectively store, share, and process data. The network layer is the practical implementation of the protocol rules, describing how participants access the system, how data is shared within the network, how the ledger is updated, and how participants verify the validity of transactions and records. It contains three core components:

Figure 7: Network Layer

NETWORK LAYER

Communications Component
- How can the network be accessed?
- How is data shared?

Transaction Processing Component
- How are transactions processed?
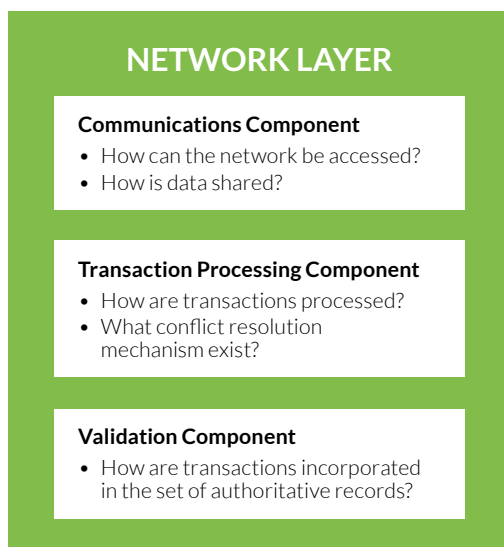- What conflict resolution mechanism exist?

Validation Component
- How are transactions incorporated in the set of authoritative records?

Communications Component:
Specifies which actors can become participants and access the network (*open* vs. *closed*), how data is shared (*public* vs. *private*) and who has the authorisation to initiate transactions (*unrestricted* vs. *restricted*).

Transaction Processing Component:
A set of processes that specifies the mechanism of updating the shared set of authoritative records: (i) which participants have the right to update the the shared set of authoritative records (*permissionless* vs. *permissioned*) and (ii) how participants reach agreement over implementing these updates.

Validation Component:
Sets out the actions undertaken by each auditor to verify whether transactions and records conform to protocol rules, i.e. are valid and non-conflicting. This is a crucial aspect of a DLT system that provides nodes with the ability to verify independently what occurs within the system.

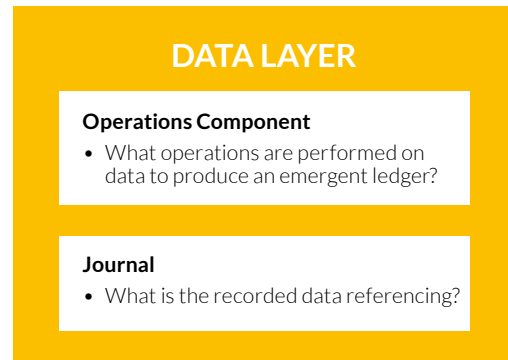There is a popular belief that records stored on a DLT system are 'immutable' and can never be reversed. However, that is not necessarily the case: DLT systems provide different degrees of transaction finality depending on the system design. This means that a confirmed (and executed) transaction may be subject to reversal. Section 5.2.3 provides a more detailed overview of the transaction finality process.

3.3 DATA LAYER

The data layer refers to the information processed and stored by the DLT system in the form of records. The data layer is at the core of the functionality the system delivers. A DLT system exists for the express purpose of creating a shared data structure – the ledger – that has a set of crucial features, the most important of which are usually persistence, transparency, standardisation[51], and censorship resistance. Within a set of information states, functions, property rights, and relations defined by a DLT system protocol, this ledger provides an authoritative version of records at a moment in time that is both shared amongst the users of the system and updated over time as users engage with one another *via* the system.

The data layer consists of two components:

Figure 8 - Data Layer

DATA LAYER

Operations Component
- What operations are performed on data to produce an emergent ledger?

Journal
- What is the recorded data referencing?

Operations Component:
The processes which govern how (and which) data is used in the creation of new records, modification of existing records, and the execution of code. This may also include 'smart contracts'.

Journal Component:
Concerns the *content* of the stored records (i.e. what data within records is being referenced, or *'what is in the blocks?'*).

Censorship Resistance

Censorship resistance is a term commonly used in the context of DLT which generally refers to the inability of a single party or cartel to unilaterally perform any of the following:

1. Change rules of the system
2. Block or censor transactions
3. Seize accounts and/or freeze balances

Programmatically-executed Transactions (Smart Contracts)

Programmatically-executed transactions (PETs) are computer scripts that, when triggered by a particular message, are executed by the system. When the code is capable of operating as all parties intend, the deterministic nature of the execution reduces the level of trust required for individual participants to interact with each other.

For example, these scripts can replace fiduciary relationships, such as custody and escrow, with code. These are often called 'smart contracts', but are not autonomous or adaptive ('smart'), nor contracts in a legal sense. Rather, they can be evidence of a contract, or a technological means of implementing a contract or agreement.

Reference/Value Linking

The nature of the records, and the value(s) to which they point, are important aspects of the journal component. Records may reference an internal object (e.g. a native token such as bitcoin/BTC or ether/ETH) or something external to the system (e.g. a physical item tracked across a supply chain).

DLT systems can only enforce records that reference endogenous (internal) objects

The distinction between endogenous (internal) and exogenous (external) objects is crucial in illustrating the boundaries of a DLT system:

it only has the ability to automatically and independently enforce transactions that point to internal resources endogenous to the system. As soon as the records reference exogenous objects, enforcement becomes dependent on external agents.

In such cases, enforcement relies on existing legal and socio-economic structures or other arrangements outside of the DLT system. Some architectures (e.g. Bitcoin) are incapable of conforming to the decisions of external agents (such as courts) without the cooperation of the participants who have control over the specific subset of assets at issue - a concept referred to as 'sovereignty'. Native, endogenous and exogenous objects are discussed in detail in Section 4.2.2.

3.4 PUTTING IT ALL TOGETHER

The proposed conceptual framework breaks a DLT system down intro three essential layers:
- The protocol layer defines, manages, and updates the global ruleset that governs the system;
- The network layer implements the ruleset and performs the steps required to reach system-wide consensus; and
- The data layer specifies the nature and meaning of the data over which agreement is reached.

Figure 9 summarises the components and processes pertaining to each layer of a functioning DLT system.

Figure 9 - DLT Systems Framework Overview

PROTOCOL LAYER

The protocol layer defines, manages and updates the global ruleset that governs the system.

GENESIS COMPONENT

INTER-SYSTEM DEPENDENCIES
PROCESS 01

Defines the boundaries of the system that is being investigated. It determines whether the system can persist on its own (e.g. self-sufficient) or whether it is dependent on another system to properly function (e.g. dependent).

PROCESS 02
CODEBASE CREATION

The act of developing an adequate codebase as the foundation of the DLT system. The codebase can be based on an existing framework or written from scratch.

RULE INITIATION
PROCESS 03

Refers to defining the ruleset that the DLT system will operate upon. This process can be performed by different actors and is specific to a particular DLT system.

ALTERATION COMPONENT

PROCESS 04
PROTOCOL GOVERNANCE

Refers to the set of decision-making processes required to alter he protocol in an orderly and legitimate manner.

PROTOCOL CHANGE
PROCESS 05

Refers to the means of implementing the proposed changes o the protocol rules

NETWORK LAYER

The network comes as a direct result from the implementation of the protocol rules. The network consists of an interconnected group of actors and processes that adhere to a technology standard (protocol) and actively participate in the exchange of data and information.

COMMUNICATIONS COMPONENT

NETWORK ACCESS
PROCESS 01

Determines the access points to the DLT system. Access to the system can be restricted or unrestricted.

PROCESS 02
DATA BROADCAST

Refers to the transmitting and relaying data across the network to connected nodes

TX INITIATION
PROCESS 03

Contains a set of instructions that will be executed once the transaction has been added to the ledger. Generating a transaction can either be unrestricted or restricted.

TRANSACTION PROCESSING COMPONENT

PROCESS 04
RECORD PROPOSAL

Refers to a record producer selecting a set of unconfirmed transactions and bundling them together to form a candidate record (e.g. in the form of a candidate block)

CONFLICT RESOLUTION RULE
PROCESS 05

Determines how disputes regarding competing or conflicting versions of valid records are being resolved and depends on the consensus mechanism in use.

PROCESS 06
INCENTIVISED TX PROCESSING

Refers to the explicit and implicit incentives present in the system to encourage record producers to engage in transaction processing by creating and proposing records.
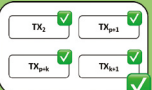
VALIDATION COMPONENT

PROCESS 07
TRANSACTION VALIDATION

Consists in verifying whether an individual transaction complies with the protocol rules before relaying it to other actors.

PROCESS 08
RECORD VALIDATION

Verifies whether a candidate record proposed by a record producer is valid according to protocol rules.

PROCESS 09
TRANSACTION FINALITY

Refers to the transition period between provisional settlement and permanent settlement for confirmed records.

DATA LAYER

Together, the protocol layer and network layer underwrite the construction of the data layer which is assembled over time as transactions are written into the ledger by the activities of participants using the DLT system.

OPERATIONS COMPONENT

PROCESS 01
INPUT

Refers to the source or method of acquiring data for the DLT system

PROCESS 02
PROGRAMMATI-CALLY-EXECUTABLE TRANSACTIONS

Changes in the data layer as a result of code-directed events that are conditional on the occurrence of some state of affairs that is captured on the ledger.

PROCESS 03
LOCUS OF EXECUTION

Determines where computations such as automated executables (e.g. smart contracts) are being executed (on- vs. off-chain).

JOURNAL COMPONENT

PROCESS 04
REFERENCE

Refers to what the data stored in the system is pointing at (internal, external, and/or self-referential

SECTION 4: SYSTEM INTERACTIONS

4.1 WITHIN THE SYSTEM BOUNDARIES

4.1.1 Layer Interdependencies

DLT systems consist of three layers that are interdependent in the sense that the 'lower' layers of the system make the 'higher' levels possible. The ordering is not spatial, but rather reflects conceptual and functional dependencies (see **Figure 2** in Section 2.3).

The protocol layer defines the rule set governing the operations of the network of interconnected participants. The protocol-governed network layer, in turn, hosts the data layer that records the time-ordered entries and modifications to the ledger.

A protocol is *just* a piece of software which by itself is inert. A protocol is 'brought to life' when it is implemented by a network. A network is a system of independent servers and storage that participate in protocol-defined operations. Unlike many traditional IT architectures, where the servers and storage are all owned, operated, and maintained by single corporate or government entity, a DLT network involves a collection of heterogeneous participants who do not necessarily know or trust one another *ex ante* but who contribute resources to the network in exchange for value gained from participating in the DLT system.

The protocol and network layers, in turn, enable the construction and maintenance of the data layer: a shared database created by multi-party consensus and having special properties such as tamper resistance (see the five key attributes of a DLT, listed in Section 2.2).

4.1.2 Layer Hierarchy

It is important to assess the relationship between the layers, and understand how these impact each other, when considering the resilience, robustness and tamper resistance of a DLT system (**Figure 10**).

Figure 10: Layer Impact Hierarchy

Transaction
processing can
censor/reverse data

Changes to protocol rules
can overrule data semantics

Changes to
protocol rules can
overrule transaction
processing decisions

DATA

NETWORK

PROTOCOL
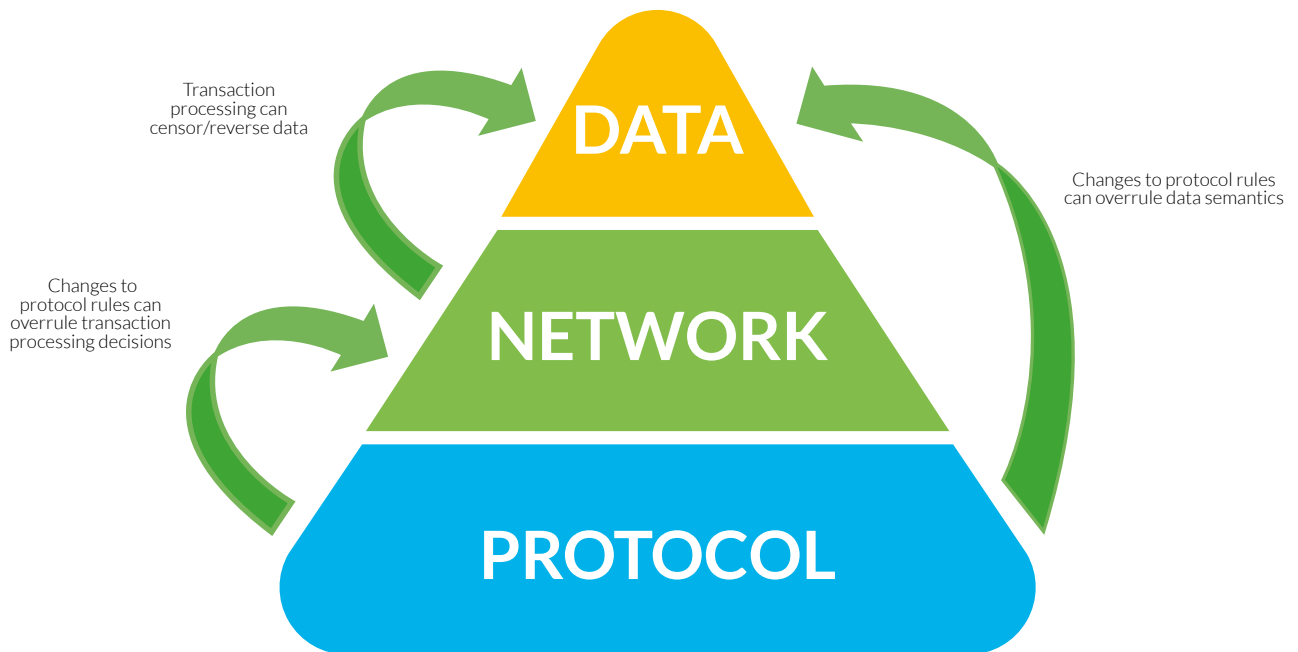
The network can impact the data layer as it processes transactions: colluding record producers can decide to censor arbitrary data by ignoring and refusing to relay corresponding transactions (i.e. not adding them to records). This means that despite the data layer being ostensibly permissionless (allowing anyone to build applications on top of the DLT system), it runs the risk of being censored or manipulated by colluding record producers.

The protocol layer can impact both the network and the data layer. Since the protocol specifies the rules under which the system operates, a change in rules can override decisions taken by record producers at the network layer during the transaction processing process. Moreover, modifying protocol rules can change the semantics of processed data and override previous configurations at the data layer.[52]

Actions and decisions at the network and data layers can always be overridden by the protocol layer

It follows that whoever has control over the protocol layer has the ability to influence directly both the network and the data layer. Decisions taken at the network layer *generally* only impact the data layer, but in certain cases either layer can be used to coordinate protocol changes across a network (e.g. Bitcoin's BIP signaling process; Decred's on-chain governance voting model).[53] This means that a system truly resilient to external interference needs to have sufficient decentralisation at both the protocol layer and the network layer in order to avoid single-party censorship and control. For example, particular blocks or transactions can be 'blacklisted' at the protocol level. A decentralised network layer on top of a centralised protocol layer is always

susceptible to arbitrary rule changes that override consensus decisions taken by record producers.[54]

4.1.3 Trade-offs: There Is No 'One Size Fits All'

Different objectives require different design choices. Design configurations at one layer of a DLT system can impact other layers or components and lead to different system characteristics, imposing a trade-off of costs and benefits. Every system makes these trade-offs in accordance with their objectives and their security, trust, and threat models. A system may favour a specific property, but that choice will inevitably come at the expense of another. For instance, the presence of trust in the system (e.g. identified, regulated entities in a closed DLT system) allows for a more flexible design approach than a DLT system built to minimise the trust requirement between participants (e.g. Bitcoin).

Early DLT systems put particular emphasis on keeping all aspects of their system 'decentralised', so as to improve the networks' censorship resistance. This came at significant cost: inefficient redundancy, inherent scaling limitations, low throughput, slow confirmation speed, high energy costs, and poor user experience, to name a few. Subsequent DLT systems have sought to address some of these issues, but these design choices come at the expense of other system properties, or an increase in the system's centralisation.[55]

Each design decision involves a complex set of trade-offs

With current technology, trade-offs most frequently revolve around the same set of properties (e.g. decentralisation, validation speed, security, actor incentivisation, complexity, throughput, trust requirements, network size). The *decentralisation/performance* trade-off has been the most discussed: generally, the more centralised the DLT system, the faster, cheaper, and more efficiently it runs.

Use case requirements should dictate design choices and acceptable trade-offs

It is rare for a design choice to strictly dominate another; generally, one cannot get all the benefits without any of the downsides. Hence, one should be aware of the trade-offs involved when analysing specific design decisions, and carefully evaluate whether the resulting trade-offs are acceptable. Ultimately, a DLT system is designed to serve a specific purpose: that purpose should dictate design choices and acceptable trade-offs. **Figure 23** in Section 6.2.5 presents an overview of some common design choices that have an impact on other system properties.

4.1.4 A Note On 'Decentralisation'

'Decentralisation', one of the key buzzwords in the DLT ecosystem, is often mistaken as an end in itself rather than being a means to an end. It is also surprisingly ill-defined given its importance in the many discussions about DLT applications.[56]

A systems theory approach can view decentralisation as the absence of a privileged party, or, conversely, the ability for a participant to choose the parties it trusts or engages with. Under this view, a system is centralised if there exists a distinct entity (or collection of entities), at any layer, with which an actor must interact. The system is fully decentralised if an arbitrary number of entities can be *feasibly* ignored or bypassed. However, this does not mean - nor guarantee - a dilution of power.

One aspect of decentralisation in the context of DLT systems, as defined by Buterin (2017), is that the data structures that are created through user engagement within the platform are distributed across many different machines under the control of participants who do not necessarily know or trust one another.[57] However, this description over-emphasises the replication of data to the exclusion of other critical elements.

Yet another view quantifies decentralization as the number of entities that must be compromised in order to prevent the system (or any subsystem) from operating as intended. In practice, however, measuring this number - or comparing it across different systems - is very difficult.[58]

Across all definitions for 'decentralisation', the recurring theme is whether the system has processes and institutions which allow free and open participation and encourage vibrant debate, rather than relegating decision-making or system management to a fixed set of entities.

'Decentralisation' in a DLT system is not a binary property: it is the accumulation of behaviours at multiple layers

Consequently**,** given that a DLT system consists of multiple processes and subsystems, 'decentralisation' of a DLT system is not a simple binary property. The degree of centralisation reflects the accumulation of interacting decisions and tradeoffs at various layers. In practice, it is more useful to identify the contributing factors to centralisation and decentralisation across a spectrum, as pure decentralisation is a seldom-achieved ideal at both the hardware and software levels.

A DLT system can have different degrees of decentralisation at each of its layers

For instance, the data layer may be decentralised (i.e. permissionless application development) while the network and protocol layer are controlled by a single party. Or the network and data layers could be decentralised yet the protocol layer centralised. Even further, there could be differences *within* a particular layer: for instance, record proposal and network access processes in the network layer could be performed by a single authority, while transaction validation and record validation could be decentralised to a certain extent.
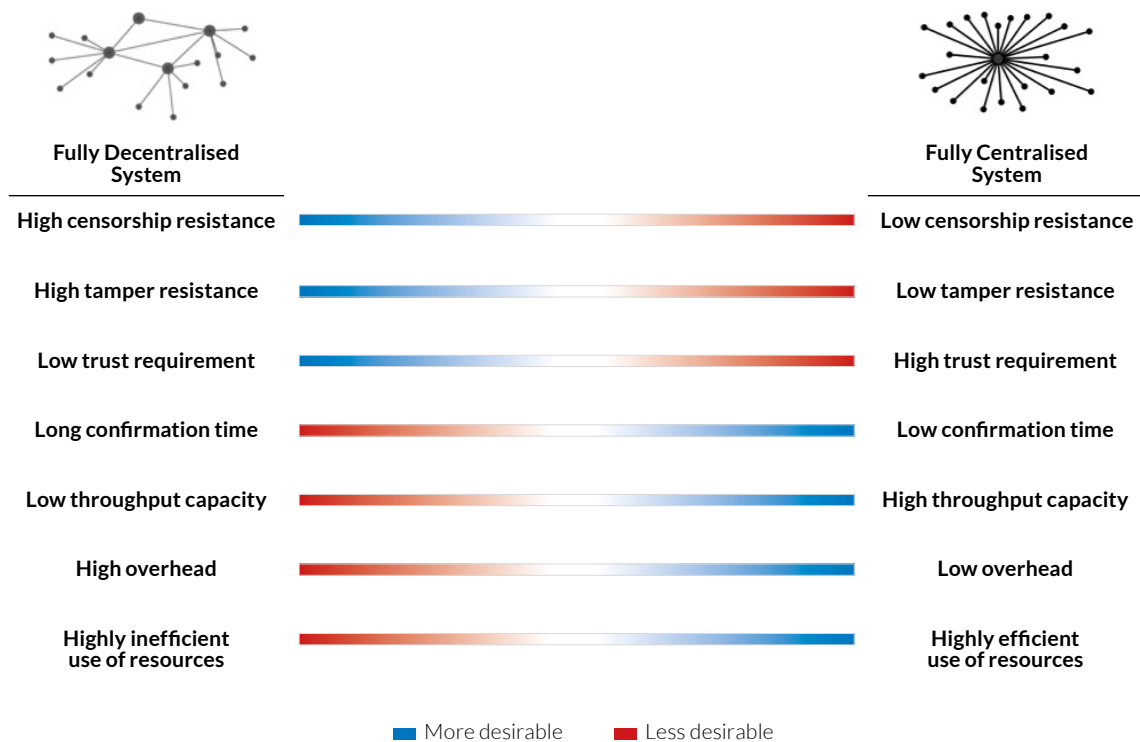
In order to determine the potential source of authority within the system (and ultimately over the stored records), attention has to be paid to the hierarchy. Thus, a DLT system cannot be considered either 'decentralised' or 'centralised' without first assessing where and how the power dynamics can potentially play out within (and between) each layer under various scenarios. These dynamics can be fluid and evolve over time, which further complicates the task of forming a definitive assessment of the system. Most DLT systems have varying degrees of decentralisation at different layers; some systems deliberately choose to centralise certain aspects so as to better meet specific objectives.

Open, public, and permissionless DLT systems such as Bitcoin strive for decentralisation to achieve censorship resistance: no single party can shut down the system, manipulate the ledger, or censor transactions. This also enhances resilience and enables the system as a whole to survive shocks, including the loss of network participants.[62]

It is important to highlight that design choices which centralise a DLT system using current technology impact not only censorship resistance, but other factors such as security, performance (or validation speed), and overhead (complexity of the information): as previously discussed, changing any component of a DLT system imposes trade-offs.

Figure 11 illustrates some of the trade-offs between a decentralised and centralised system. Some DLT systems may be more centralised in certain aspects to emphasise a specific property deemed desirable within the system. Given that there may be instances where the centralisation of a process would be desirable, it is not reasonable - nor feasible in practice - to require that all layers of a system be fully decentralised in order for it to be classified as a DLT.

Figure 11: One Choice At The Expense Of Others

Fully Decentralised System		Fully Centralised System
High censorship resistance		Low censorship resistance
High tamper resistance		Low tamper resistance
Low trust requirement		High trust requirement
Long confirmation time		Low confirmation time
Low throughput capacity		High throughput capacity
High overhead		Low overhead
Highly inefficient use of resources		Highly efficient use of resources

■ More desirable   ■ Less desirable

For instance, prioritising the validation speed of records and transactions may come at the expense of the complexity and the size of the ledger, as the functions (recordkeeping, smart-contracting) and record sizes might be reduced to a minimum. It may also decrease the overall security, or tamper-resistance of the system, if the network is centralised in order to increase validation speed. Similarly, choosing a Proof-of-Stake (PoS)[63] consensus mechanism over Proof-of-Work (PoW)[64] - in order to improve speed and reduce energy consumption - may impact actors' incentives and thus affect the security and tamper-resistance of the system.

Alternatively, aiming at increasing the security of the system may hinder the validation speed, reduce the allowed transaction size due to the space necessary for encrypting transactions, and discourage actor participation, as the costs of running fully-validating nodes might become prohibitively expensive over time. With such an objective, the complexity of the technology might also be limited as a collateral effect, because reducing complexity would help improve speed and record size. Finally, dynamic membership networks that allow anyone to join or leave at will can grow particularly large in network size but will result in higher confirmation times due to latency issues.

All choices regarding centralisation or decentralisation of elements create both costs and benefits for DLT systems

4.2 BEYOND THE SYSTEM BOUNDARIES

4.2.1 Systems Perspective

DLT systems are seldom self-sufficient. Instead, they are often in constant interaction with other systems. **Figure 12** depicts the different types of systems configurations seen in DLT deployments.

Figure 12: A Systems Perspective
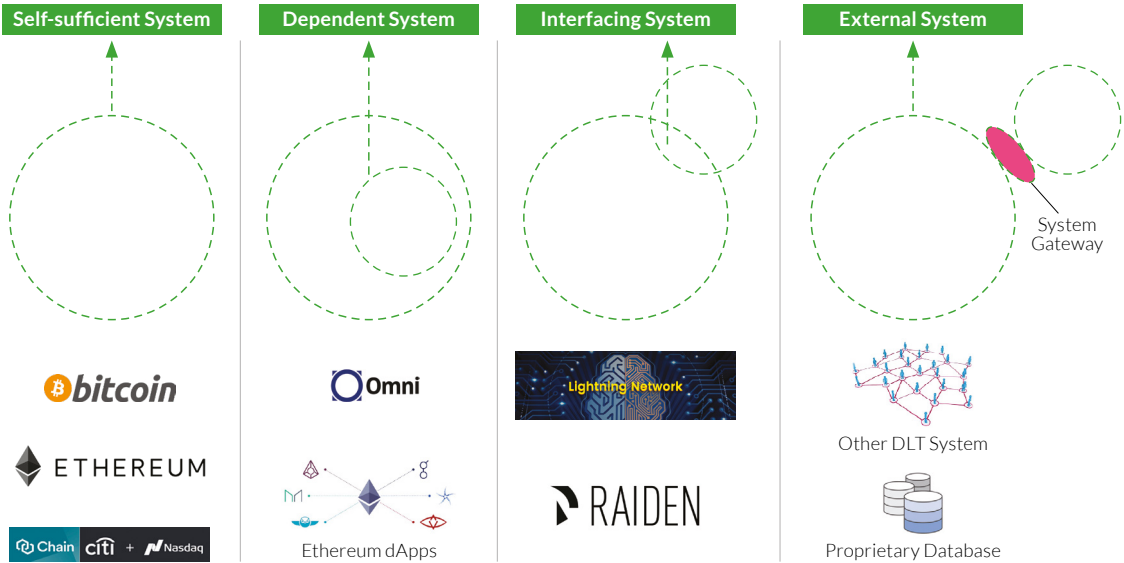
Self-sufficient Systems

A self-sufficient DLT system has all of the components necessary for its continued operation incorporated into its basic architecture, and the system itself is sufficient to enable the core functionality. Such systems do not depend on other systems for their operation, apart from the wider Internet infrastructure (e.g. reliance on TCP/IP or similar protocols and the underlying network infrastructure). Examples are open systems such as the Bitcoin and Ethereum main nets as well as permissioned systems such as the NASDAQ Linq blockchain.

Depending on the nature of the records (e.g. exogenous/external), a system may require inputs from external sources. This requirement alone is insufficient to preclude classifying a DLT system as self-sufficient. For example, a DLT system representing asset transfers in a supply chain should be able to persist and function even if external data is not received, although it will depend on gateways or interfaces to supply data pertaining to the creation or physical transfer of assets. Section 4.2.2 elaborates on the relationship between self-sufficient and external systems.

Dependent Systems

A dependent DLT system must interface with another DLT system in order to function properly. On its own, such a system is not self-sufficient. Examples of dependent systems are Omni and Counterparty which operate on top of Bitcoin as well as the dApps ('*decentralised applications*') running on Ethereum. Omni, for instance, is wholly dependent on Bitcoin, as it is a protocol which tracks assets that exist as arbitrary data within certain Bitcoin transactions.[65] Omni borrows its security and finality properties from Bitcoin while adding semantic content to transactions; it does not exist outside of Bitcoin.

Interfacing Systems

An interfacing DLT system is a system that 'opportunistically' employs core functionality provided by another DLT system but which could easily be reconfigured to use another 'base-layer' DLT system if needed/desired. This means that if one system ceased to exist, the interfacing system would be able to survive for at least some time on its own and may be able to continue operating by exploiting the functions of an alternative 'base layer' DLT. The long-term survival of an interfacing system depends on the continued existence of at least one 'base-layer' DLT system, and a collapse of a base-layer system may cause significant disruption to the interfacing system. Examples include 'layer-2' solutions such as the Lightning Network based on Bitcoin and the Raiden Network based on Ethereum.[66] These systems are commonly designed to improve the scalability and functionality of the base layer, without compromising network decentralisation or security.

External Systems

An external system is any other system that is yoked to, or coupled with, a 'focal' DLT system. The external system is architecturally unrelated to, or distinct from, the focal DLT system. An external system can be connected to the system in question via a gateway (either via a direct or indirect interface). This could be other DLT systems as well as proprietary databases or services (e.g. wallets, exchanges, or applications). An example of a direct system gateway would be an atomic swap protocol, whereas an indirect system gateway would involve a trusted intermediary to transfer tokens from a proprietary database to the system, or between two incompatible DLT systems.

4.2.2 Exogenous And Endogenous References

Records may reference endogenous data and/or exogenous data. Endogenous data is information that comes exclusively from within the core system. Exogenous data refers to data that tracks information about the same entity or a relationship that is external to the DLT system. Exogenous entries may be *representations* of assets (monetary or non-monetary), or other information. An example of endogenous data would be a record of bitcoin units within the Bitcoin system, while an example of exogenous data could be a record tracking luxury handbags on a global supply chain.

Figure 13: Gateways Connect A DLT System To Exogenous Objects
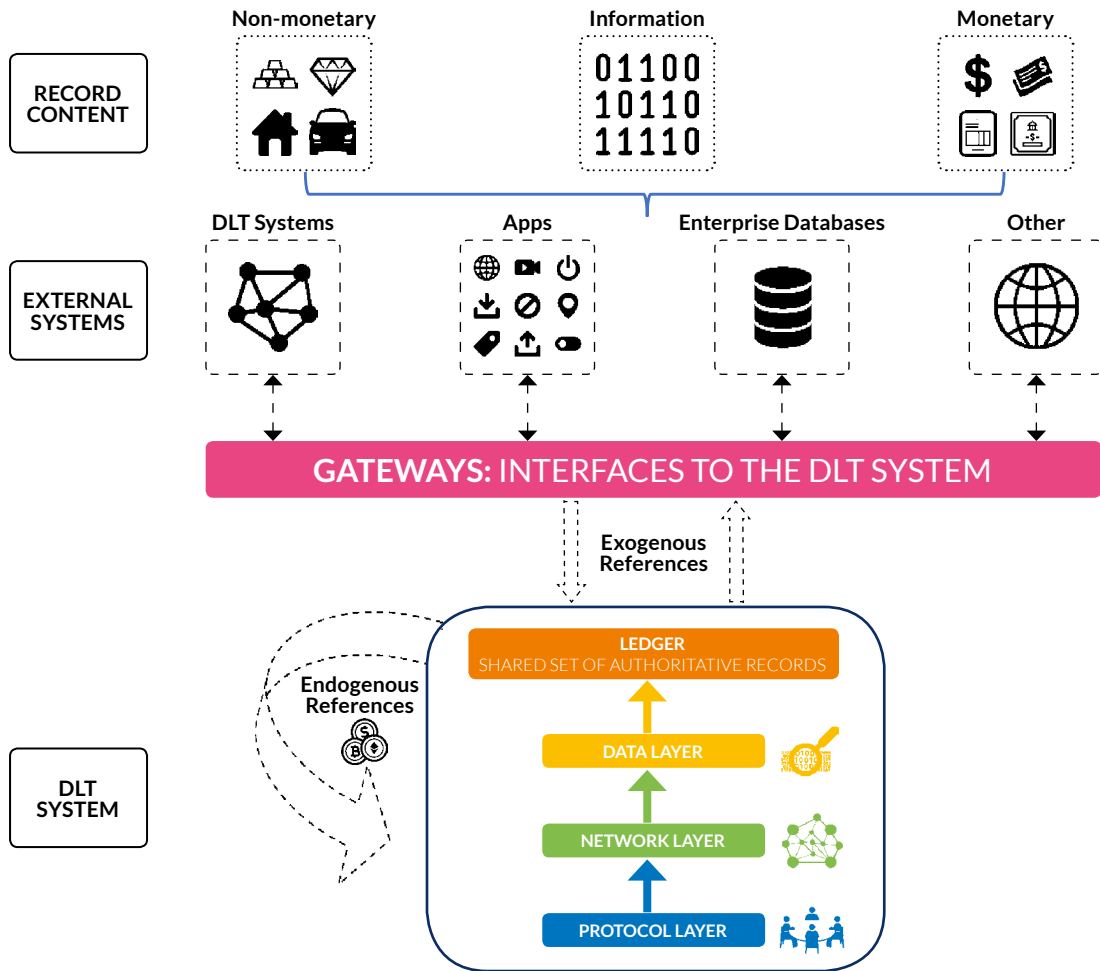
More generally, if the data in a journal only refers to facts about user actions on the platform, or facts about the past history of the DLT system itself, then the reference type is *endogenous*. If, on the other hand, the data refers to some state in the world external to the DLT system or the users interaction with the DLT system, than the reference type is *exogenous*. **Figure 13** provides a representation of the pathways for data interaction between the DLT system and external platforms.

The distinction between exogenous and endogenous data may seem superficial, but it is not. Bitcoins, for example, only exist as data records within the Bitcoin DLT system. The only way to change its state is to change the data record within the Bitcoin system. Handbags, stock prices, or weather readings are all examples of things that exist independent of the DLT system, and whose state can change without altering records within the DLT system that tracks them.

Linking a DLT with an external system requires gateways to act as an interface: oracles bridge the gap between the DLT system and external systems by serving as a source of information. In the case of a supply chain, this could be RFID tags attached to the luxury goods and scanned by machines at each intermittent station. Other external systems (e.g. other DLT systems, apps, proprietary enterprise databases, etc.) may communicate their own recorded information with the original DLT system, providing data that become part of it.

Interfacing with an external source of data requires a gateway; this undermines the ability of a DLT system to automatically and independently enforce decisions

Using the example of a supply chain, a DLT system may properly record the movement of RFID tags, but those devices may not necessarily be attached to (or embedded in) the objects they are taken to represent: one could imagine a shipping crate filled with nothing but RFID tags that could fool the DLT system into accepting a false transaction representing a large transfer of physical assets. Similarly, some number of RFID tags may be defective, and transfers would not necessarily be recorded.

A DLT system only has effective enforcement capabilities (i.e. the ability to automatically execute decisions) with regards to endogenous data (i.e. internal references that exclusively exist within the boundaries of the system). Records referencing exogenous resources, facts, or events are provided by external agents who must be entrusted through non-system means to report honestly and/ or enforce decisions. In the prior supply chain example, parties with a shared interest in properly recording the transaction would need to develop systems to prevent or ameliorate any malfunctions, such as coupling the RFID interface with a physical inspection.

A DLT system can only independently and autonomously enforce decisions that involve endogenous record references
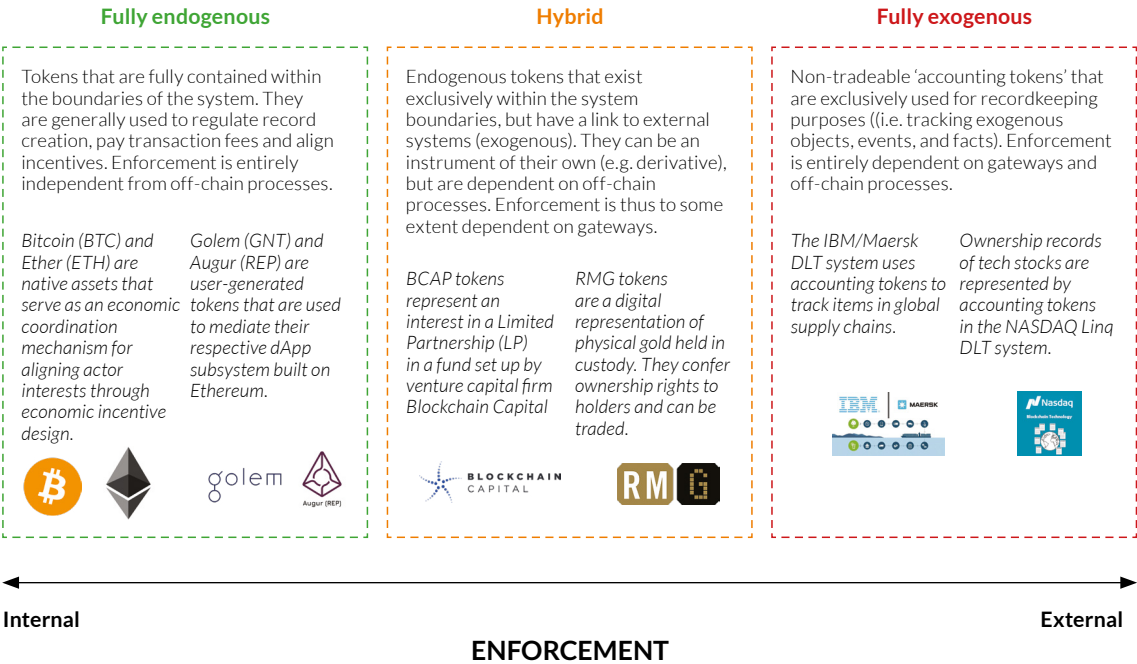
What can be written to the journal are ultimately determined by the protocol. This doesn't mean, however, that the protocol necessarily explicitly lays out all of the data types that can be recorded by the DLT system. For example, a DLT system capable of supporting a Turing-complete smart contract provides its users with the flexibility to define novel data types for the smart contracts that they create.

Finally, records may also reference data that carry aspects of both endogenous and exogenous nature, in which case they are referred to as *'hybrid'*. An example would be a security directly issued on a DLT system (*endogenous* because it exclusively exists within the system boundaries) that is dependent on off-chain cash flows (*exogenous* because it requires a connection to an external system). In the case of hybrid references, it is more difficult to determine the exact enforcement capabilities of the DLT system because the relationship between both aspects may vary

from one record to another. Hybrid references are a fast-developing subfield as corporations are increasingly attempting to convert existing assets on to a DLT system. As such, it may require further gradation in the future.

Figure 14 summarises the three types of references that records in a DLT system can point to. Native assets are fully endogenous as they are entirely contained within the boundaries of the system and do not require a formal connection to the external world. In contrast, fully exogenous records are exclusively referencing external data, which necessitates the existence of a gateway to (a) receive information and (b) enforce decisions outside the DLT system. Exogenous data is meaningless within the system without an attached link bridging to the material world. In contrast, hybrid records reference data which shares both endogenous and exogenous characteristics. As a result, enforcement is to some extent dependent on gateways.

Figure 14: Three Types Of References

Fully endogenous

Tokens that are fully contained within the boundaries of the system. They are generally used to regulate record creation, pay transaction fees and align incentives. Enforcement is entirely independent from off-chain processes.

Bitcoin (BTC) and Ether (ETH) are native assets that serve as an economic coordination mechanism for aligning actor interests through economic incentive design.

Golem (GNT) and Augur (REP) are user-generated tokens that are used to mediate their respective dApp subsystem built on Ethereum.

Hybrid

Endogenous tokens that exist exclusively within the system boundaries, but have a link to external systems (exogenous). They can be an instrument of their own (e.g. derivative), but are dependent on off-chain processes. Enforcement is thus to some extent dependent on gateways.

BCAP tokens represent an interest in a Limited Partnership (LP) in a fund set up by venture capital firm Blockchain Capital

RMG tokens are a digital representation of physical gold held in custody. They confer ownership rights to holders and can be traded.

Fully exogenous

Non-tradeable 'accounting tokens' that are exclusively used for recordkeeping purposes ((i.e. tracking exogenous objects, events, and facts). Enforcement is entirely dependent on gateways and off-chain processes.

The IBM/Maersk DLT system uses accounting tokens to track items in global supply chains.

Ownership records of tech stocks are represented by accounting tokens in the NASDAQ Linq DLT system.

Internal — ENFORCEMENT — External

SECTION 5: A DEEPER DIVE INTO THE FRAMEWORK

In this section we identify commonly adopted configurations of processes within the protocol, network, and data layers of the framework we introduced in Section 3. We apply these configurations to distinguish between specific DLT systems, highlighting differences and similarities, in Section 6. This section assumes that the reader has familiarised themselves with the definitions, concepts, and terminology we introduced in preceding sections.

5.1 PROTOCOL LAYER

The protocol layer defines, manages and updates the global ruleset that governs the system.

5.1.1 Genesis Component

The genesis component of the protocol layer refers to the processes required to undertake and complete before launching the DLT system.

Inter-System Dependencies

Inter-system dependencies defines the boundaries of the system that is being investigated. It determines whether the system can persist on its own (i.e. *self-sufficient*) or whether it is dependent on another system to properly function (e.g. *dependent*). Section 4.2 discusses possible configurations in greater detail.

Table 1: Inter-System Dependencies

System type	Description
Self-sufficient system	Able to operate on its own - not dependent on another system.
Dependent system	Unable to operate on its own - relies on another system to function.
Interfacing system	Able to operate temporarily on its own - long-term survival closely reliant on another system
External system	Self-sufficient system that interacts with a DLT system (generally acting as data source/recipient)

Codebase Creation

Codebase creation is the act of developing an adequate codebase as the foundation of the DLT system. The codebase can be based on an existing framework or written from scratch. Popular existing frameworks are open-source codebases from permissionless DLT systems (primarily Bitcoin and Ethereum)

and permissioned DLT systems such as Hyperledger suite, Corda, Chain, and Multichain. There are also closed-source codebases for proprietary platforms available provided by companies such as Digital Asset, Clearmatics and SETL.

Table 2: Codebase Creation Configurations

Lense	Configurations	Description
Codebase	Existing framework	Many DLT systems share similar codebases that are based on existing frameworks
	New/from scratch	Code is substantially different from existing frameworks, either in terms of purpose, coding language and/or architecture
Openness	Open-source	Can be forked: network can be replicated
	Closed-source	The codebase is developed by a private company or consortium for enterprise or consumer use

Rule Initiation

Rule initiation refers to defining the ruleset upon which the DLT system will operate. This process can be performed by different actors and is specific to a particular DLT system.

Table 3: Rule Initiation Configurations

Lense	Configurations	Description
Administrator	Anonymous	A founder whose real identity remains hidden by operating under a pseudonym.
	Volunteer	A set of people that collaborate on a project on a voluntary basis; often loosely connected with no formal governance structure.
	Consortium	A group of private and/or public institutions that formally join forces to collaboratively develop and manage a project.
	Foundation	A non-profit foundation coordinates and oversees activities; formally registered under a legal structure that may impose fiduciary obligations.
	Company	The project is initiated (and generally managed) by a single corporation or joint venture entity with designated management.
Codebase Maintainer	Open-source community	Everyone has the right to propose changes to the codebase; a formal decision-making process may or may not be in place.
	Company	Control over the codebase is exclusively exercised by a company.
	Consortium	An organised group of stakeholders is in charge of collectively maintaining the codebase.
Means	Formal protocol specification	The protocol is formally specified - generally in the form of a specific documentation - and followed by all client implementations.
	Reference client	The reference client dictates the rules of the protocol - generally in the absence of a formal protocol specification.[67]

5.1.2 Alteration Component

The alteration component refers to the processes in place which enable modification of the protocol rules. Protocol alterations can include the removal of technical errors (bugs), improvement of security and functionality of the system, and extension or restriction of existing protocol rules.

Protocol Governance

Protocol governance refers to the set of decision-making processes which enable alteration of the protocol in an orderly[68] and legitimate[69] manner. This is a subset of broader project governance, which encompasses the full set of processes and norms which guide and define coordination and action, but which may not be embedded formally within the DLT system.[70]

An essential element of any proposed protocol alteration is the means by which it is adopted and ratified - or, in other words, how *legitimacy* is conferred upon the proposal by the network's participants. Because legitimacy in this context is a social concept, we find it appropriate to identify some of the possible 'socio-political' relationships found in DLT systems.

Table 4: Protocol Governance Configurations

Configurations	Description
Anarchic	Protocol change proposals are provided and approved on a cooperative and voluntary basis, due to absence of a central authority.[71] Contentious proposals run the risk of fracturing the network, resulting in a permanent split.
Dictatorship	Decisions over changes on protocol rules are taken by a single entity (e.g. person, company, mining pool).
Hierarchical	Individuals have the ability to propose changes, but recognised leadership such as a Foundation or a committee in control of a key code repository all but ensures protocol changes will rely on the consent of the leaders. Alternatively, the system may be explicitly divided into 'first-class' and 'second-class' users/nodes.
Federation	A group of agents vote on protocol alterations, linked by a horizontal relationship scheme. Members of a Federation need not have equal voice/power, nor even necessarily known to each other.[72]
Plutocratic	Protocol change proposals are voted on, with each vote weighted by the importance of each proposer or voter. In the plutocratic case, substantial weight is given to a minority of voters (e.g. due to high ownership share of the weighting asset) .[73]
Democratic	Protocol change proposals are voted on, with each vote weighted by the importance of each proposer or voter. In the democratic case, a minority of voters do not have substantial weight in vote outcomes.

Protocol governance takes many forms and is often only implicitly defined. DLT systems considered to have anarchic (or loose) governance do not have a foundation, corporation, or 'benevolent dictator' to guide decision-making. These often rely on governance norms, processes, and procedures inherited from the free / open source software community. Examples include informal processes such as discussions among developers on mailing lists and at conferences. In a dictatorial setting, these same processes may exist, but with an acknowledged leader empowered to make unilateral decisions.

In some cases, protocol governance does not fit neatly into only one category. For example, the EOS blockchain operates as a federation of Block Producers, which are selected by user/custodian votes (weighted by token holdings). This arrangement implicitly divides the network into sets of 'first-class' and 'second-class' nodes, giving it elements of hierarchy, federation, and democracy/plutocracy.[74] Accordingly, each category should be seen as *a* mechanism, rather than *the* mechanism, of protocol governance for a given DLT system; each system will exhibit one among countless permutations of these mechanisms, and the relative importance of each mechanism may change over time.

It is also important to realise that the 'anarchic' mode of governance will always exist as a governance mechanism for any open-source project, as distinct from closed-source and proprietary projects. This is because a DLT system based on an open-source codebase operates with the cooperation of its users and record producers. In the face of an attempt to force a protocol change on users, they will always have the option of forking the code to reverse or ignore it. This will result in the creation of a distinct DLT system, albeit one with a shared history up until the moment of divergence (a '*hard fork*' leading to a network split).[75] A consequence of this is that DLT systems can, in some circumstances, be regarded as decentralised with respect to governance, even when there is a single 'core' code repository.[76] In contrast, proprietary systems may not allow for this kind of user-driven 'exit'.[77]

On-chain Governance

On-chain governance refers to the incorporation of protocol governance features within the data layer of the DLT system. The intent is to formalise governance, thereby enhancing legitimacy and avoiding network splits due to contentious or uncoordinated protocol changes. A diverse set of on-chain voting schemes have been developed for DLT systems, ranging from barometers of community sentiment to enforceable referenda. However, on-chain governance features are generally only a supplement to other forms of governance.[78]

Protocol Change

The protocol change process considers the entities who may propose protocol changes, the means by which protocol changes are funded, and how the changes are implemented. Implementation can involve different mechanisms such as providing updates to specific node software, forced upgrades to all nodes running a particular instance of the software, and the blacklisting of nodes running older versions of the software.

Table 5: Protocol Change Configurations

Lense	Configurations	Description
Proposal	Open alteration	Open systems allow anyone to propose changes.
	Filtered alteration	Proposals are conditional on some requirement of the system. For example, Dash and Tezos allow anyone to propose changes, conditional on the approval of token holders. Other systems may require a centralised initial submission in which proposals undergo curation based on merit or strategic objectives.
	Authorised alteration	Corporations or consortia may restrict who can propose changes.
Funding	Altruist	Some protocols rely on volunteer efforts, while others (e.g. Monero) may fund development work through voluntary charitable contributions by token holders or other interested parties.
	Supported development	Foundations, such as the Ethereum Foundation, may fund development work through grants. Although this helps ensure coherence and developer accountability, it may also have a centralising effect on the protocol layer. Additionally, the Foundation itself may be vulnerable to capture by self-interested parties or state actors. Grants may be awarded by following a specific process determined by the Foundation.
	Network-funded development	Development work is supported through the issuance of new tokens. The extent of this issuance may be determined by the network offering a bounty for meeting development objectives, or may be defined by developers themselves, subject to network approval.
	Corporate-sponsored development	Corporations or consortia fund development through the sponsoring organisation(s).
Implementation	Run client software of choice	Participants implement changes individually by choosing to run a specific instance of a client software. No action from an administrator is required, but this may result in network splits from contentious or uncoordinated changes. This mode tends to reduce developer or record producer control of the governance process.
	Pushed to clients	Changes are implemented by pushing updates directly to clients - generally launched by an administrator or an on-chain governance system. This mode tends to prioritise the integrity of the network, but may tend to cede power to developers or record producers.

Different DLT systems may allow for a mix of these mechanisms. For example, Ethereum accepts voluntary contributions from its community as well as from developers supported by grants.

5.2 NETWORK LAYER

The network of a DLT system exists as a direct result of the implementation of the protocol rules. The network consists of an interconnected group of actors and processes that adhere to a technology standard (protocol) and actively participate in the exchange of data and information over integrated communication channels.

5.2.1 Communications Component

Communications refer to the exchange and sharing of data across participants in a DLT system.

Network Access

Network access determines the right of entry to the DLT system; this is the right to connect to the network. Access to the system can be unrestricted, meaning that anyone[79] can freely join, leave, and rejoin the system at any point in time, or it may be restricted by a gatekeeper responsible for granting access rights to specific entities. Open networks generally have dynamic and flexible membership, whereas closed networks may have static/fixed membership.

Generally, auditors get direct access to the network by running fully-validating nodes: they are considered 'first-class' citizens with greater rights, as they are able to broadcast, validate and relay transactions and records. Participants can also get indirect access to the network by either running 'lightweight clients' (also called '*SPV nodes*') that query full nodes for transaction data or by connecting to a particular service via an Application Programming Interface ('*API*') part of a server designed and programmed to receive requests and send responses to other servers or devices.

Table 6: Network Access Configurations

Lense	Configurations	Description
Openness	Open	Unrestricted network access: simply requires downloading and running a software client.
	Semi-open	Access is partially restricted: prospective participants need to apply; generally decided via on-chain voting/approval of existing network participants.
	Closed	Access is restricted to vetted participants. Requires a gatekeeper to onboard new members.
Channels	Full node	Fully performing the functions and tasks available in the system: receives, validates, stores and broadcasts transactions and records in the system; performs independent validation.
	Lightweight node	Client that allows performing basic tasks such as creating transactions - does not fully validate the system state. Requires connecting to a full node for receiving information of the system.
	API access	External end-users connecting to a full node via an Application programming interface (API).

Generally, the more open a system is, the more exposed it is to malicious actors. In particular, these systems are vulnerable to 'Sybil' attacks, where the attacker creates numerous fake identities to increase influence over the network.

A Sybil attack is a class of attack in which a malicious actor gains influence or disguises malfeasance by creating numerous false identities

Because identity is an exogenous (i.e. 'real world') property, a DLT system cannot, by itself, prevent such attacks; it must rely on the intervention of an outside agent (such as a credentialing authority) or Sybil-resistance mechanisms (such as PoW or PoS) to mitigate these attacks.

Data Broadcast

Data broadcast is the process of transmitting and relaying data across the network to connected nodes. Data can be *raw* and unformatted or in a *standardised* format (e.g. in the form of a transaction or record). Data can be broadcast to every node in the DLT system (*universal diffusion*) or only shared with a specific subset of nodes (*multi-channel diffusion*).[80] In the latter case, data diffusion is generally limited to the transaction parties involved in a trade or who depend on specific historic transactions; effectively creating a private sub-network which is often referred to as a '*channel*'. This concept is commonly referred to as *sharding*.

Early DLT systems (e.g. Bitcoin, Litecoin) use the universal data diffusion model, which still remains the dominant broadcast method

In order to meet confidentiality and privacy requirements for enterprises, more recent frameworks have implemented the multi-channel diffusion model (e.g. Hyperledger Fabric, Corda). Others, such as Cosmos, are designed to act as 'hubs' so that independent DLT systems can be linked together as part of an application-based sharding scheme. Although universal data diffusion is technically used within each Cosmos subnetwork, the inter-network system resembles multi-channel diffusion. In either case, multi-channel diffusion prevents nodes from storing and processing data that is of little interest to them, and can theoretically lead to better scaling.[81]

Table 7: Data Broadcast Configurations

Configurations	Description
Universal data diffusion	Data is broadcast to all nodes: convergence towards a single shared set of records (*global consensus*)
Multi-channel data diffusion	Data is only shared between a subset of nodes directly involved in a specific operation (*local consensus*)

An implication of multi-channel data diffusion is that not all network participants need to be involved in reaching consensus over a channel state: only channel participants are required to reach agreement over data stored in that channel ('*local' consensus*). This differs significantly from systems with global data diffusion as every single node is required to come to consensus over the global state of the system ('*global' consensus*); failure to achieve consensus by some subset of nodes may result in the departure of the nodes which do not agree, or a divergent DLT system (*network split*).
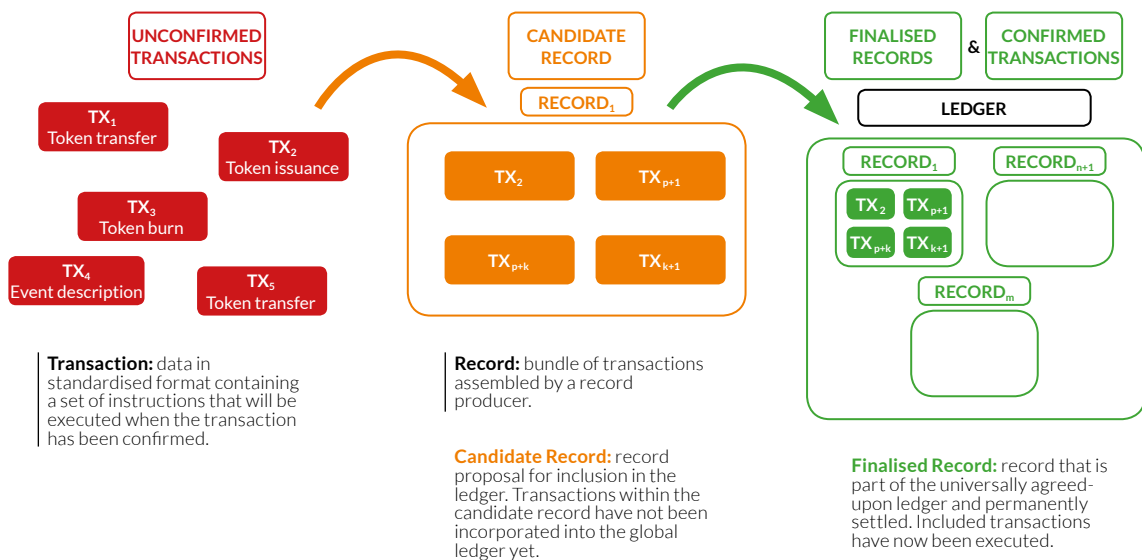
Transaction Initiation

A transaction contains a set of instructions that will be executed once the transaction has been added to the ledger. Generating a transaction can either be *unrestricted* (i.e. open to anyone) or *restricted* to select participants. Transactions are generated by users signing a message in a standardised format using the corresponding private key. There are different interfaces available to end-users for creating and broadcasting transactions to the network (e.g. desktop and mobile wallets).

5.2.2 Transaction Processing Component

Transaction processing describes the set of actions required to add an unconfirmed transaction to the shared set of authoritative records. A transaction is considered (provisionally) settled once added to a record ('confirmed'), which results in the execution of the set of instructions embedded within the transaction. However, a single confirmation[82] is generally insufficient to be relied upon for subsequent transactions; the record must be 'finalised' before the transaction outputs may be relied upon by the system. Finality is discussed in Section 5.2.3.

Figure 15: Conceptualising Transaction Processing In A DLT System

Transaction: data in standardised format containing a set of instructions that will be executed when the transaction has been confirmed.

Record: bundle of transactions assembled by a record producer.

Candidate Record: record proposal for inclusion in the ledger. Transactions within the candidate record have not been incorporated into the global ledger yet.

Finalised Record: record that is part of the universally agreed-upon ledger and permanently settled. Included transactions have now been executed.

Records are subject to the *consensus algorithm* used by the DLT system to reach agreement over the state of the system. This includes a process for determining whether a proposed record is valid, as well as rejecting invalid records (e.g. records which are defective or noncompliant) and choosing among different, yet equally-valid records.

Record Proposal

Record proposal refers to a record producer selecting a set of unconfirmed transactions and bundling them together to form a candidate record. Record proposal can be *permissionless* in that any network participant has the right to produce a new candidate record, or *permissioned* in the sense that only a specific subset of participants are allowed to generate a candidate record. Note that this only refers to network participants, i.e. actors that have already been approved to join the system.

Table 8: Record Proposal Configurations

Configuration	Description
Permissionless	Any network participant has the ability to create a candidate record
Permissioned	Record creation is restricted to a subset of participants

As the records are subject to network consensus, they must adhere to the protocol rules. At a basic level, they must be formatted correctly and contain no invalid or conflicting transactions. Additionally, each record must include a reference/pointer to a previous record, and, if appropriate, a PoW or other Sybil resistance technique.

Consensus algorithms can be classified according to their difficulty level (in energy consumption or financial terms). Algorithms with unlimited difficulty are uncapped in the resources they may require to reach consensus. For instance, in the case of Bitcoin's PoW computation, the difficulty of finding a valid solution increases as additional hashing power is added to the system.[83] In contrast, other algorithms (e.g. Practical Byzantine Fault Tolerance/BFT) do not consume a significant amount of resources and have limited difficulty.

In open systems, a mechanism for resisting Sybil attacks needs to be incorporated into the consensus algorithm. Permissioned and closed systems generally do not require this component, as Sybil attacks are prevented by carefully vetting entities before granting them permission to join the network and produce records.

Early open DLT systems exclusively used PoW as a Sybil prevention mechanism. PoW makes it computationally difficult (i.e. costly and time-consuming) to produce new records but easy for others to verify them. In contrast, emerging PoS-based systems use staking of endogenous resources (e.g. native asset) in order to choose the next record producer.[84] PoW systems are resource-intensive but are robust as long as the number of participants is large and sufficiently distributed.In contrast, PoS systems are less resource-intensive than PoW systems, but are also generally

vulnerable to 'nothing-at-stake' and 'grinding' attacks, among others.[85]

Closed DLT systems generally have static membership and thus a complete overview of all participants in the network should be possible. They often use mechanisms such as Round-Robin schemes or algorithms such as Practical Byzantine Fault Tolerance (PBFT), Paxos, or Raft in which nodes temporarily elect one node to be a leader (i.e. record producer).

Consensus algorithms and Sybil resistance mechanisms are an active area of research. Further information on the variety of consensus mechanisms used in DLT system can be found in Seibold & Samman (2016).[86]

Conflict Resolution Rule

The conflict resolution rule determines how disputes regarding competing or conflicting versions of valid records are being resolved and depends on the consensus algorithm in use. For instance, Bitcoin resolves a temporary split caused by two competing valid blocks at equal height by choosing the block on the branch that carries most cumulative PoW (*longest-chain rule*).[87] Tezos adapts the longest-chain rule for PoS, defining the 'weight' of a block as the number of 'endorsements' it receives from randomly-selected record producers. Alternative resolution rules involve *unanimous agreement* of all record producers or passing a certain *quorum threshold*.

As with all design decisions, each consensus algorithm reflects a set of trade-offs

The 51% Attack

An attack in which an entity or cartel with a majority of the 'votes' (e.g. computing power) in a DLT system produces records faster than the rest of the network. Eventually, these records are revealed to the network, causing the records of 'honest' nodes to be replaced due to the conflict resolution rule. The 51% attack is the classic attack against DLT systems. PoW-based systems are especially vulnerable to such attacks; a similar attack in PoS-based systems is called the 'long-range' attack. It should be noted that in some cases, DLT systems are vulnerable to attacks carried out by less than 51% of voting power (e.g. *selfish mining*, which is theoretically feasible with only one-third of voting power).

Incentivised Transaction Processing

Incentivised transaction processing refers to the explicit and implicit incentives present in the system to encourage record producers to engage in transaction processing by creating and proposing records. These incentives can be of different nature (e.g. monetary, legal,

social) and can be expressed directly by protocol rules (e.g. block rewards in native asset) or by external factors (e.g. contractual agreements established between participants). Many DLT systems use a combination (**Table 9**).

This distinction matters when categorising DLT systems. Open systems such as Bitcoin

tend to be secured via economic incentive designs that make use of an endogenous network resource (native asset) as an economic coordination mechanism to align incentives. Dependent systems may use the native token of the system they are dependent upon. In contrast, closed networks with known and vetted participants generally rely on pre-established authority relations through mutual contractual obligations.

Table 9: Incentivised Transaction Processing Configurations

	Intrinsic	Extrinsic
Monetary[xx]	Block rewards (subsidy + transaction fees)	Paid services (fees)
Non-monetary	Required for transaction creation[xx]	Contractual obligations, reputation, etc.

5.2.3 Validation Component

Validation refers to the set of processes required to ensure that actors independently arrive at the same conclusion with regard to the authoritative set of records. This includes verifying the validity of unconfirmed transactions, verifying record proposals, and auditing the state of the system. This component is a crucial differentiator from non-DLT systems in that it provides participants with the ability to independently audit the system.

Transaction Validation

Transaction validation consists of verifying whether an individual transaction complies with the protocol rules before relaying it to other actors. This involves verifying that the transaction is properly formatted, has a valid signature, and does not conflict with any other transaction. In certain systems, transactions may be subject to encumbrances (such as a prohibition on transfers until a certain time or condition is met). Such encumbrances are often integral to the operation of programmatically-executed transactions ('smart contracts').

Record Validation

Record validation is verifying whether a candidate record proposed by a record producer is valid according to protocol rules. If the proposed record is deemed valid by an auditor, it is added to the journal and relayed to other nodes. While the exact process differs from one system to another, it generally involves verifying the validity and uniqueness of each transaction contained in the record, as well as checking whether the conditions specified by the record proposal process are met (e.g. verifying that a valid PoW is attached).[88]

The combination of transaction and record validation performed by auditors provides the ability to independently compute the entire state of the system from genesis (*full audit*).
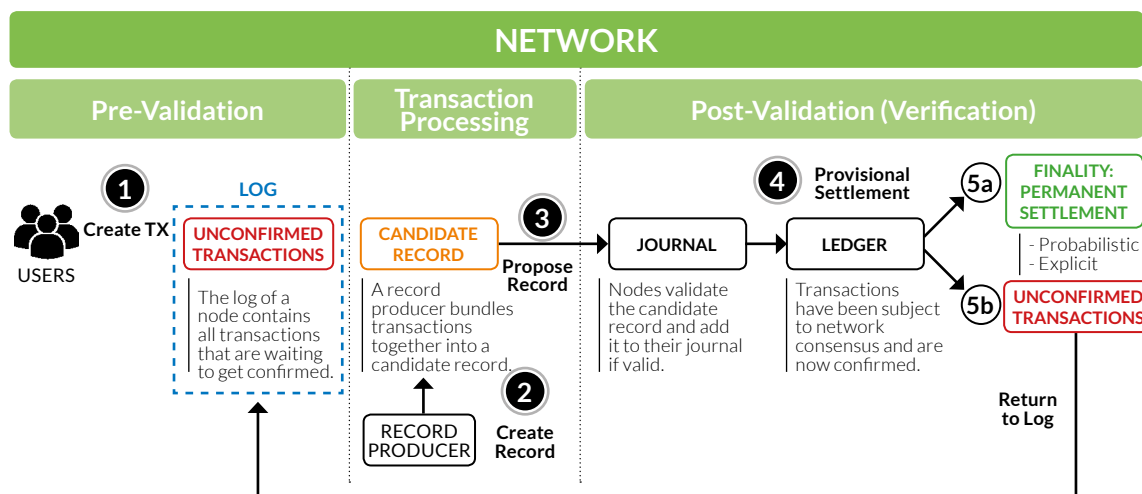
Transaction Finality

Contrary to popular belief, a confirmed transaction or record is not necessarily irreversible. Transaction finality determines when a confirmed record can be considered *final* (i.e. not reversible). Finality can be *probabilistic* (e.g. PoW-based systems that

are computationally impractical to revert) or *explicit* (e.g. systems that incorporate 'checkpoints' that must appear in every transaction history). Finalised records are also called *permanently settled*. Records that have been produced, but which are feasible to revert, are *provisionally settled*. Provisionally-settled records become permanently settled after the transition period between record creation and finality.

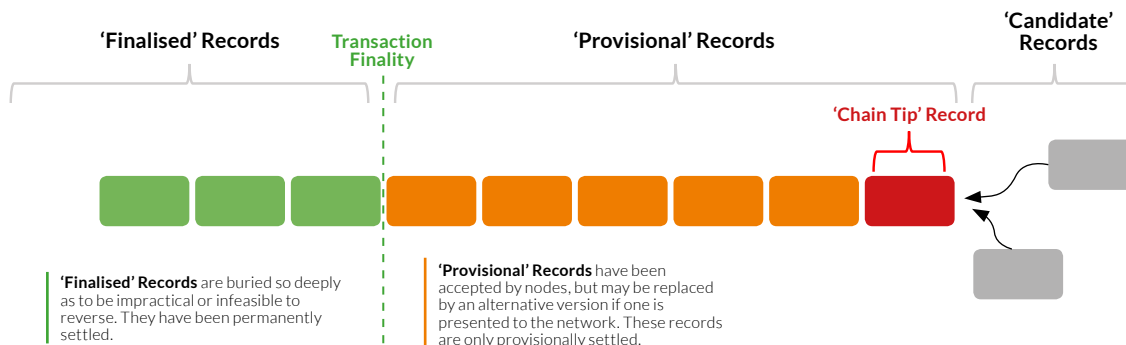**Figure 16: Transaction Finality In DLT Systems**

Figure 16 offers a schematic description of the steps involved in the settlement process. First, a user creates a transaction and broadcasts it to the network. Each auditor verifies whether the transaction complies with the protocol rules. If it is considered valid, the node adds the transaction to its log (also called 'mempool'), a virtual environment that stores unconfirmed transactions waiting to be added to the shared set of authoritative records.

In the transaction processing phase, record producers will arbitrarily select unconfirmed transactions from their mempool and bundle them together into a candidate record. They will then perform the steps required by the consensus mechanism to propose this

candidate record to the network. Nodes will review the received candidate record and its content; if it passes the validity checks, the record is added to the node's journal. Individual journals eventually converge towards a single common 'ledger' as the transactions are confirmed and executed.

However, confirmed records could be abandoned (*orphaned*) for the sake of an alternative, competing record: this means that during the provisional settlement phase, confirmed transactions can get reversed - in which case they are returned to the log as unconfirmed transactions, waiting to be included in the next record. The duration of the provisional settlement phase depends on the system design and set-up. Some systems implement nearly immediate finality, whereas others have 'probabilistic' finality in the sense that, theoretically, records can always be reversed. In practice, however, the likelihood of such a 'reorganisation' decreases rapidly with each additional record added to the ledger, because the financial costs attached to PoW mining can become prohibitive as an attacker attempts to reach 'deeper' into the ledger. As long as records are in the provisional settlement phase, they should not be considered 'final'.[89]

Typically, users refrain from interacting with data that is only provisionally settled, because the possibility of reversion creates a risk that assets can be double-spent. The provisional settlement period represents a safety factor for nodes, helping ensure that transactions are fully incorporated in the ledger (as opposed to the node's local journal) before users rely on their outputs - thus preventing double-spending attacks.[90]

Some systems also implement 'checkpoints' to limit the possibility of 'long-range' attacks. In a long-range attack, a block producer creates a competing subchain without revealing records to the network, and then reveals all these private records simultaneously to cause other nodes to orphan long-accepted records. A 'checkpoint' is a block that an honest node will never orphan. As a result, the checkpoint limits the 'reach' of a long-range attack. However, checkpoints also create a theoretical risk of a permanent network split under certain conditions, such as 'eclipse' attacks.[91]

Finally, it should be noted that changes to protocol rules have the power to alter the form of the ledger, which can directly impact transaction finality.[92]

Table 10: Transaction Finality Configurations

Finality	Probabilistic	Explicit
Provisional	In theory always; practically, a time window determined by network conditions	Short time window determined by protocol
Finalised	In theory never; practically, after a certain block depth	After a specific block depth determined by protocol

5.3 DATA LAYER

The protocol layer determines how a DLT system will function and how it will operate. The network layer implements the protocol layer. Together, the protocol layer and the network layer form the basis for the data layer which is assembled over time as transactions are written into the ledger by the activities of participants using the DLT system.

5.3.1 Operations Component

The operations components of the Data Layer include all of the processes through which the journal – and, derivatively, the ledger – is co-created and transformed by users as they interact with the system.

Input

The input process refers to the source or method of acquiring data for the DLT system. As discussed in Section 4.2.2, data sources may be internal or external, which may reflect users actively interfacing with the system, or a change in state driven by an internal system process, or an externally-driven process (e.g. a transaction initiated by an external or interfacing system), or smart contract.

More generally, we define internal sources of input as any record or transaction that is created by, or the direct result of, a user interfacing with a DLT system 'on the platform' (i.e. on-chain). External sources of input, on the other hand, are the result of input from off-chain systems that interface with the DLT system but that are, in principle, separable from the core platform (i.e. they are a dependent or interfaced system in the framework outlined in **Figure 12** in Section 4.2.1). Hybrid sources such as 'generalised state channels' allow users to run programs outside the DLT system and relay the state to the system at any time; development of these techniques is, however, still in its infancy.

Table 11: Input Configurations

Types	Configurations	Description
Internal sources	Transactions	A set of cryptographically-authenticated instructions to modify the state of the ledger.
	Records	Bundle of transactions that have been added to the shared set of authoritative records (*global ledger*).
	Automated executables	Programs that exist inside the system (or on another DLT system that interfaces with the focal system) which are allowed to trigger phenomena once a predetermined condition is verified.
External sources	Sensors	Physical devices that are able to broadcast specific information to selected systems (e.g. RFID chips).
	Information providers	Entities that collect and organise data which are allowed to interact with selected systems (e.g. a price API).
Hybrid sources	Generalised state channels	A transaction type that allows users to run programs outside the DLT system, with each state transition representing a private 'counterfactual'. At any time, the final state can be relayed to the DLT system.

Programmatically-executable Transactions

Not all changes to the data layer are the direct result of internal or external inputs. Some changes to the data layer are the result of code-directed events that are conditional on the occurrence of some state of affairs that is captured on the ledger. A prime example here are changes that are initiated by programmatically-executed transactions (i.e. 'smart contracts'). When encoded conditions are met, a smart contract automatically executes and, as a result, a number of downstream events occur – some of which may include changes to the ledger itself. DLT systems that support the design and execution of a wide range of programmatically-executed transactions are generally referred to as *stateful*. Users can build and run complex expressive smart contracts directly at the system level: the supported computer language is flexible and general-purpose to theoretically allow the modeling of any imaginable program.

Other DLT systems only support a limited range of programmatically-executed transactions at the base layer: they are based on a simple script language that generally features a limited series of OP_Codes enabling the design of relatively simple, specific-purpose programs. These systems are usually referred to as *stateless*.[93]

Ethereum (Solidity), Tezos (Michelson) and EOS (WebAssembly) are three stateful DLT systems that are equipped with a Turing-complete language to design complex smart contracts. They are often referred to as 'smart contract platforms'. In contrast, DLT systems such as Bitcoin and Monero merely provide support for a simple scripting language that allows limited type of operations.

Table 12: Programmatically-executed Transactions Configurations

Type	Description
Stateless	Fixed-function machine at the base system layer: allows for the execution of limited computations.
Stateful	General-purpose computations executed on-chain by network participants via an integrated virtual machine.

Locus of Execution

The locus of execution determines where computations such as programmatically-executed transactions are being executed. Generally, the locus of execution can either be *on-chain* (i.e. internal) or *off-chain* (i.e. external).

On-chain computations are executed internally in each auditors' own environment ('execution engine'). This environment can range from a simple fixed-purpose machine to a more complex general-purpose virtual machine (e.g. Ethereum Virtual Machine/ EVM) that provides a rich Turing-complete environment. On-chain smart contracts are executed by every auditor in the system and are thus often referred to as 'self-executing' or 'self-performing'.[94]

Off-chain computations are executed in environments that are external to the system (in an external or interfaced system). While off-chain computations are initiated by events and processes on the DLT system, the execution is 'outside-the-system' in the sense that relevant work is not being performed at the core DLT system layer. In this case, DLT systems can be understood as serving the function of a settlement layer for the external

or interfacing systems that run the core transaction logic.

In some systems, execution may occur in hybrid *side-chains*. For example, Ethereum's Plasma network alleviates some computational burdens on nodes through parallelisation. Similarly, Cosmos acts as a 'hub' that regards each independent system it is connected to as a 'side-chain' from the perspective of the larger inter-network it coordinates.

In 'stateless' DLT systems - characterised by limited expressiveness - more complex business logic generally tends to get pushed to external or interfacing systems where it will be executed in a different runtime environment. While this layered approach limits on-chain capabilities, it can also provide benefits such as reducing the 'attack surface' of the base layer, potentially providing increased privacy and confidentiality as well as the possibility of better scaling prospects and enabling low-latency applications that would otherwise be constrained by network delays.[95]

Table 13: Locus Of Execution Configurations

Types	Configurations	Description
On-chain	Fixed-purpose machine	Limited to a narrow set of operations.
	General-purpose virtual machine	Capable of performing an open-ended range of operations.
Off-chain	Coordinator	The DLT system's primary purpose is to initiate and control off-chain computations.
	Automated Arbiter	The DLT system's core function is to settle the outcomes of an automated executable, which is otherwise executed by off-chain parties or systems.
Side-chain	Subnetwork	Side-chains usually operate according to the same 'on-chain' architecture of their respective DLT systems, but distribute computational loads to subnetworks to improve overall system scaling.

5.3.2 Journal Component

Reference

A ledger emerges over time as users interact with a DLT system. The ledger, however, is an abstraction. Input processes and automated executables do not directly operate on the ledger *per se*, but rather the journal. The specific kinds of information and/or data structures that are held by a node are always specific to the DLT system. A DLT system focused on digital payments, for example, needs to hold information about the assets held by individual users. A DLT system that enables smart contracts, on the other hand, has to be able to hold the customised code implementing the smart contract on the platform.

Types of Reference

There are four different kinds of reference data: endogenous variables, exogenous variables, hybrid variables, and self-referential data.

Endogenous (internal reference) refers to data that tracks information about variables that are native to the system. In Bitcoin, one endogenous reference variable, for example, is used to track the number of bitcoins the user has at any particular time. This internal variable is updated as the user sends and receives bitcoins to/from other accounts.

Exogenous (external reference) refers to data that tracks information about variables that exist outside of the system. A *hybrid* reference refers to data that shares both endogenous and exogenous characteristics. These three types of references are also discussed in Section 4.2.2.

There is a fourth reference type that is neither endogenous nor exogenous: This neutral or null data type is a *self-referential* reference. For example, a smart contract is simply a piece of code that can execute when certain conditions are met. While a smart contact may require information about external and internal system variables, the code itself has no intrinsic reference to anything outside of itself (a 'null reference').

Table 14: Types Of References And Value Linking

Type	Description
Endogenous	Refers to data or digital assets that exclusively exist within the boundaries of the system and do not require a connection to external systems. Decisions can be automatically enforced by the system as the data and/or assets are intrinsic to the system. For example, native assets such as ETH and associated dApp tokens are endogenous references of the Ethereum system.
Exogenous	Records referencing data that is exclusively extrinsic to the system and thus requires gateways for connecting to the external world and enforcing transactions. Recordkeeping-only systems are an example of this type in that they only record events or facts occurring externally.
Hybrid	Digital assets that share both endogenous (i.e. exclusively exist within the boundaries of the system) and exogenous characteristics (i.e. have some link to the external world). Hybrid can also refer to systems that support both endogenous and exogenous references.
Self-referential	Pieces of code (e.g. smart contracts) that do not reference endogenous or exogenous variables, although they may require information about internal or external variables.

SECTION 6: APPLYING THE FRAMEWORK - CASE STUDIES

In this section, we use Bitcoin as a case study to show how the framework can be applied to analyse and characterise a DLT system. We will then proceed to compare other notable DLT systems and examine where they differ.

All DLT systems presented in this section are *self-sufficient systems*; we omit discussion of dependent, external, and interfacing systems (e.g. ERC20 tokens, the Lightning Network).

6.1 BITCOIN

Bitcoin was introduced conceptually in October 2008 and launched in January 2009. The rationale behind Bitcoin was to create a digital value transfer and storage system with rapid settlement that would not rely on trusted third-parties.

Protocol

Table 15: Bitcoin: Protocol Layer

Layer	Component	Process	Configuration
Protocol	Genesis	Inter-System Dependencies	Self-sufficient system: not dependent on an external system.
		Codebase Creation	Codebase is built from scratch and open-source.
		Rule Initiation	Reference client ('Bitcoin Core') specifies rules; alternative implementations follow the same ruleset.
	Alteration	Protocol Governance	Anarchic: coordinated via the Bitcoin Improvement Proposal (BIP) process; Bitcoin Core GitHub repository.
		Protocol Change	Open alteration: running software client of choice (generally 'Bitcoin Core').

Genesis

A self-sufficient system, Bitcoin was released as open-source software in the form of a reference client (the 'Satoshi Client', now 'Bitcoin Core') by an individual or group of people under the pseudonym of Satoshi Nakamoto. There is no formal protocol specification: instead, the reference client specifies the rules which have tended to be followed by alternative client implementations (e.g. bitcoind, libbitcoin, Bcoin).

Alteration

Bitcoin's governance can be described as *anarchic*: there is no formal standard or set of procedures to make changes to the protocol. Instead, participants run the client version that implements (and thus enforces) the rule set that they deem valid. Applying changes to the protocol thus requires a global coordination effort to convince nodes to upgrade to a newer client version that supports the proposed changes.

Nodes that do not upgrade will cause a split in the network that effectively leads to the emergence of a new DLT system where both systems share the same transaction history up until the point of the fork (e.g. Bitcoin Cash in August 2017). Because each subnetwork's value to the participants is related to the number of users on it, users are strongly incentivised to upgrade simultaneously for all but the most contentious changes.

Users and developers can submit pull requests to the Github repository that hosts the relevant client implementation. The reference client Bitcoin Core has a standardised process (*Bitcoin Improvement Proposals*, or *BIPs*) to discuss proposed changes to the protocol. However, the changes are only effective if the vast majority of nodes decide to download and run the upgraded software client.[96] As a result, miners 'signal' support of BIPs within the blocks they produce as a way of gauging community sentiment before adopting the associated proposal. Changing Bitcoin's ruleset is very hard, as demonstrated by the SegWit debate which resulted in the creation of an alternative system - Bitcoin Cash - in August 2017.[97]

Network

Table 16: Bitcoin: Network Layer

Layer	Component	Process	Configuration
Network	Communications	Network Access	Open: open to anyone that downloads and runs a client.
		Data Broadcast	Universal data diffusion: data is broadcast globally to the entire network.
		Transaction Initiation	Unrestricted: anyone can submit a transaction through a node using a variety of ways.
	Transaction Processing	Record Proposal	Permissionless: Miners select unconfirmed transactions and create a candidate block. A valid candidate block requires attaching a valid SHA-256 hash to the block header (by selecting a nonce that gives the hash a sufficiently low value).
		Conflict Resolution Rule	Nodes will follow the blockchain instance that carries most cumulative Proof-of-Work ('longest/most-worked chain rule').
		Incentivised Transaction Processing	Intrinsic and monetary: miners receive a block reward (and transaction fees) in the form of a native token ('bitcoin') for submitting a valid block.
	Validation	Transaction Validation	Full nodes validate every unconfirmed transaction before relaying it to other nodes.
		Record Validation	Full nodes verify the PoW, block format, and every transaction in the block before adding the block to their journal and relaying it to other nodes.
		Transaction Finality	Probabilistic: a transaction included in a valid block may get reversed if a competing (longer) blockchain instance takes over. General rule of thumb: 6 confirmations before a transaction can be considered settled.

Communications

Bitcoin is an open system providing unrestricted access to the network: anyone can join, leave, and re-join the network simply by downloading and running a software client, limited only by their technical ability, equipment capability, and bandwidth. Within the network, all data is broadcast globally: every full node stores unconfirmed transactions in its mempool, and all confirmed transactions in the form of the Bitcoin blockchain. Anyone can submit a transaction, provided that transaction rules are respected.[98] End users external to the system can use wallet software to create a transaction, which will then get sent over alternative communication channels to a full node that will broadcast it to the network.

Transaction Processing

Record producers (called *miners*) select unconfirmed transactions in the mempool and bundle them together into a candidate block. Before submitting the candidate block to the network, miners need to attach a valid PoW to the candidate block.[99] This *mining process* generally requires a substantial amount of energy to find a valid solution because the difficulty adjusts to match the hashpower on the network. In case two unrelated miners find a valid solution at a similar time, the network applies the 'longest-chain rule' to decide which of the two candidate blocks to accept.[100] Miners are incentivised to process transactions by receiving a reward intrinsic to the system for every successfully mined block: in addition to transaction fees, successful miners are allocated new units of the native token (*bitcoin*).

Validation

Bitcoin full nodes verify transactions twice. First, a node checks the validity of incoming unconfirmed transactions before relaying it to other nodes. This prevents invalid transactions from getting widely broadcast in the network and taking up significant network resources. In a second step, a node verifies the block (*record*) that includes the now-confirmed transactions. If the block passes the validity test, the node updates its journal and broadcasts the block to connected peers. Settlement in Bitcoin is only *probabilistic*: in theory, miners could, at any point, cause a chain reorganisation that would reverse all transactions included in now-orphaned blocks. In practice, however, it is considered safe to regard a transaction finalised after waiting for 6 confirmations (i.e. blocks mined on top of the block that includes the transaction in question).

Data

Table 17: Bitcoin: Data Layer

Layer	Component	Process	Configuration
Data	Operations	Input	Primarily internal (e.g. previous outputs: UTXOs, scripts). External: arbitrary data for timestamping purposes (e.g. via OP_RETURN).
		Programmatically-executed Transactions	Fixed-function: limited scripting language enables simple on-chain smart contracts (e.g. multi-signature and time-locked contracts) .
		Locus of Execution	On-chain for native asset transfer.
	Journal	Reference	Endogenous: native asset unique to system (BTC).

Operations

Bitcoin primarily takes internal sources as inputs for creating new records since the main purpose of the system is to provide secure value transfer of the native token. This means that internal asset units ('bitcoin') get used as inputs for transactions. Bitcoin can also be used as a global public notary: data (or pointers to externally stored data) can be embedded into Bitcoin transactions for tamper-resistant timestamping purposes. In terms of automated executables ('smart contracts'), Bitcoin only allows the design and on-chain execution of relatively simple programs via its native scripting language (*'Script'*): multi-signature and hashed timelock contracts are good examples of simple smart contracts implemented on Bitcoin.

Journal

The system records transactions describing the creation and transfer of the native token *bitcoin. Bitcoins* exist exclusively within the boundaries of the Bitcoin system as entries in the Bitcoin 'ledger'. The records produced by the Bitcoin system thus point to internal values ('bitcoin') that have no direct connection to external systems.[101] This means that transfers recorded by the Bitcoin system do not rely on external agents for enforcement; instead, they are automatically and independently enforced 'on-chain' by network participants.[102]
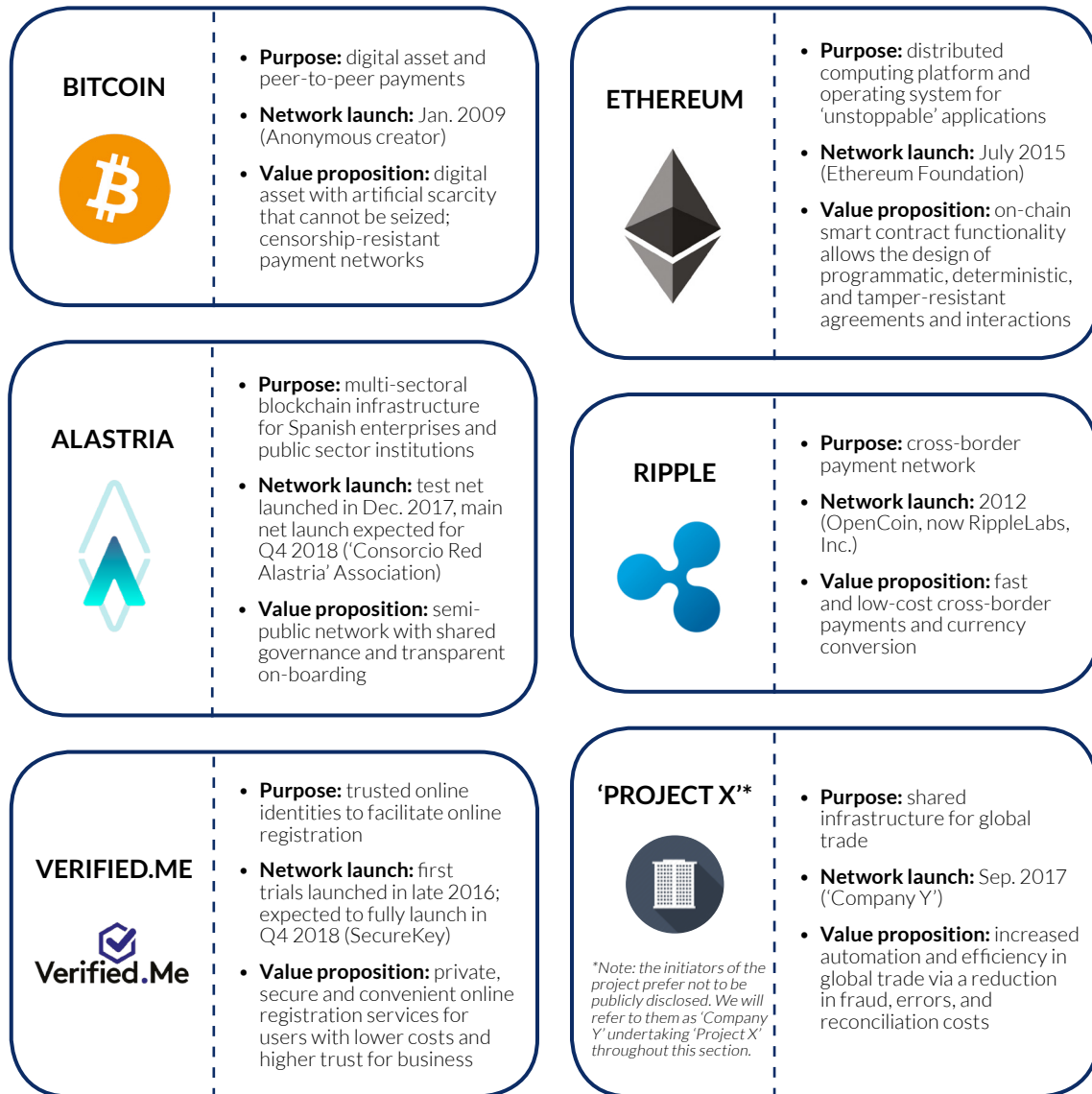
6.2 COMPARATIVE ANALYSIS

The framework we have developed is a tool to map the relationships between system layers, components, processes, and actors for different DLT systems. These systems are dynamic and constantly evolving, requiring frequent updates to the analysis. The following comparative analysis should be understood as our best attempt to describe the state of the indicated systems at the time of this report's publication.

6.2.1 Case Studies

We have selected a total of six case studies to show how the framework can be used for comparative analysis (**Figure 17**). Each of the selected systems has unique characteristics and properties that are a result of the design choices (chosen configurations) influenced by assumptions and assessments of trade-offs.

Figure 17: Case Studies Overview

BITCOIN

- **Purpose:** digital asset and peer-to-peer payments
- **Network launch:** Jan. 2009 (Anonymous creator)
- **Value proposition:** digital asset with artificial scarcity that cannot be seized; censorship-resistant payment networks

ETHEREUM

- **Purpose:** distributed computing platform and operating system for 'unstoppable' applications
- **Network launch:** July 2015 (Ethereum Foundation)
- **Value proposition:** on-chain smart contract functionality allows the design of programmatic, deterministic, and tamper-resistant agreements and interactions

ALASTRIA

- **Purpose:** multi-sectoral blockchain infrastructure for Spanish enterprises and public sector institutions
- **Network launch:** test net launched in Dec. 2017, main net launch expected for Q4 2018 ('Consorcio Red Alastria' Association)
- **Value proposition:** semi-public network with shared governance and transparent on-boarding

RIPPLE

- **Purpose:** cross-border payment network
- **Network launch:** 2012 (OpenCoin, now RippleLabs, Inc.)
- **Value proposition:** fast and low-cost cross-border payments and currency conversion

VERIFIED.ME

- **Purpose:** trusted online identities to facilitate online registration
- **Network launch:** first trials launched in late 2016; expected to fully launch in Q4 2018 (SecureKey)
- **Value proposition:** private, secure and convenient online registration services for users with lower costs and higher trust for business

'PROJECT X'*

Note: the initiators of the project prefer not to be publicly disclosed. We will refer to them as 'Company Y' undertaking 'Project X' throughout this section.

- **Purpose:** shared infrastructure for global trade
- **Network launch:** Sep. 2017 ('Company Y')
- **Value proposition:** increased automation and efficiency in global trade via a reduction in fraud, errors, and reconciliation costs

The following subsections will only highlight the most important differences between the case studies. A full comparative analysis using the framework can be found in Appendix B.

6.2.2 Are These DLT Systems?

Recall that our formal definition of a DLT system required 5 elements. **Table 18** summarises whether these five elements are satisfied by the systems we include in our case study. We will discuss the technical aspect of

each system that causes it to deviate from the results of Bitcoin within this section as part of our framework. Most importantly, notice that while all the systems we included have launched ('Launch Date'), not all the systems we include are fully functional as DLT systems yet. However, all included systems have released plans that clearly indicate how they will gain properties required for DLT systems. We refer to these systems as *potential DLT systems.*

Are 'Potential' DLT Systems A Thing?

It is important to note that not all systems that claim to be DLT systems can be considered DLT systems according to the formal definition presented in Section 3. In particular, both Verified.Me and 'Project X' - in their current state - do not meet all conditions specified by the definition. However, these systems have the *potential* to become a DLT system as their architecture lends itself to satisfy all necessary conditions.

Table 18: Not All Case Studies Currently Meet The DLT System Criteria

	Launch Date	Shared Recordkeeping	Multi-party Consensus	Independent Validation	Tamper Evidence	Tamper Resistance
Bitcoin	January 2009	✔	✔	✔	✔	✔
Ethereum	July 2015	✔	✔	✔	✔	✔
Ripple	2012	✔	?	✔	✔	?
Alastria	2018 (Testnet)	✔	✔	✔	✔	?
Verified.Me	2016 (Trial)	✔	✔	✔	✔	?
'Project X'	September 2017	✔	✖	✖	✔	✖

Both Bitcoin and Ethereum satisfy the five properties required of a DLT system.[103] Ripple Labs' influence over validator nodes makes both multi-party consensus and tamper resistance properties contentious. Both Alastria and Verified.Me have unclear tamper resistance properties as of yet. These contentious properties result in the three systems having disputed DLT status -- some see them as DLT systems, some do not -- but we include them as they can be analysed by our framework. 'Project X', which is still in its early stages and uses only a single validator, does not yet have multi-party consensus, independent validation, or tamper resistance, but it does have a clear plan to increase the number of record producers and independent auditors and so could eventually become a full-fledged DLT system.

The ability to use this framework to analyse and compare systems that are not yet fully functional DLT systems is an additional benefit of a systems-based approach.

6.2.3 Protocol

System Launch

Bitcoin is the only DLT system within the case studies that was launched by an anonymous entity as an open-source project. In contrast, all other case studies were released by a known entity, albeit under different structures: for instance, Ethereum was released by the Ethereum Foundation and the Alastria main net will be launched by the 'Consorcio Red Alastria' Association, whereas Ripple, Verified.Me and 'Project X' have all been launched by a single company (OpenCoin/Ripple Labs, SecureKey, and 'Company Y', respectively).

Codebase

Bitcoin, Ethereum, and Ripple are based on codebases designed from scratch, whereas Alastria is built on a slightly derived version of Ethereum called Quorum (initially developed by J.P. Morgan). Verified.Me and 'Project X' use the Hyperledger Fabric framework (**Figure 18**).

Figure 18 - Codebase Comparison
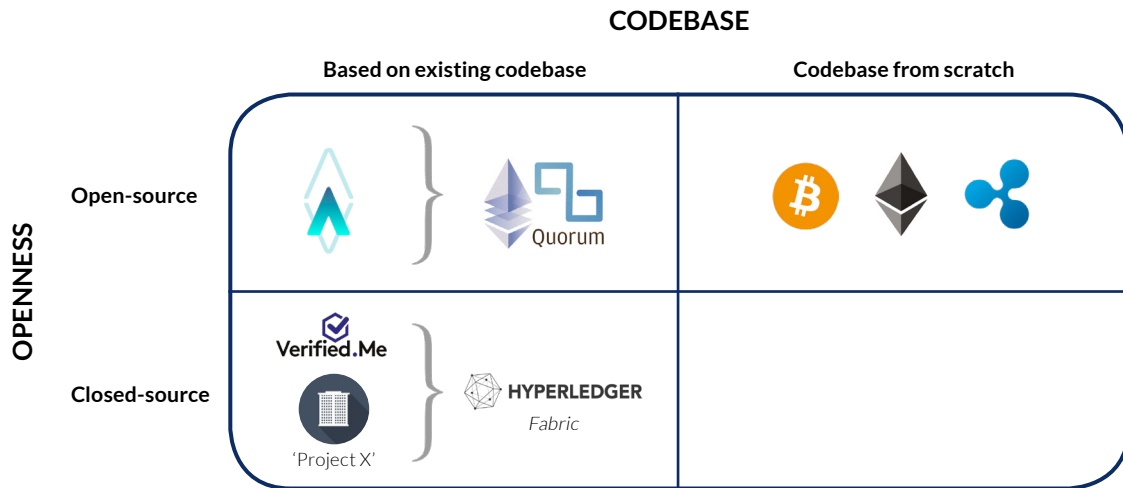
Bitcoin, Ethereum, Ripple, and Alastria are open-source: this means that network participants may decide to fork the project (i.e. 'copy-paste' the codebase) and create an alternative system which is based on similar premises. In contrast, Verified.Me and 'Project X' are closed-source, which prevents participants from cloning the system.

Governance

System governance is one of the key differentiators for the analysed DLT systems. Fully open and permissionless systems such as Bitcoin and Ethereum lack a formalised set of procedures and standards around how protocol rules are updated. However, there are differences in how both projects approach this issue: Bitcoin's reference client Bitcoin Core has a dedicated BIP (Bitcoin Improvement Proposal) process through which changes to the codebase are submitted, reviewed, and finally accepted or rejected. While in theory access to the BIP process is open to anyone, practice has shown that a limited number of volunteering core developers have disproportionate influence over protocol changes.[104] Nevertheless, the SegWit episode has shown that reaching global consensus over proposed rule changes is very difficult.[105]

On the other hand, Ethereum development is heavily influenced by the Ethereum Foundation and its development roadmap. While there is an EIP (*Ethereum Improvement Proposal*) process in place and users can choose from multiple clients, history has shown that with one notable exception[106], upgrades proposed by the Ethereum Foundation and its co-founder Vitalik Buterin have been accepted by system participants without contention.

Figure 19: Governance Comparison
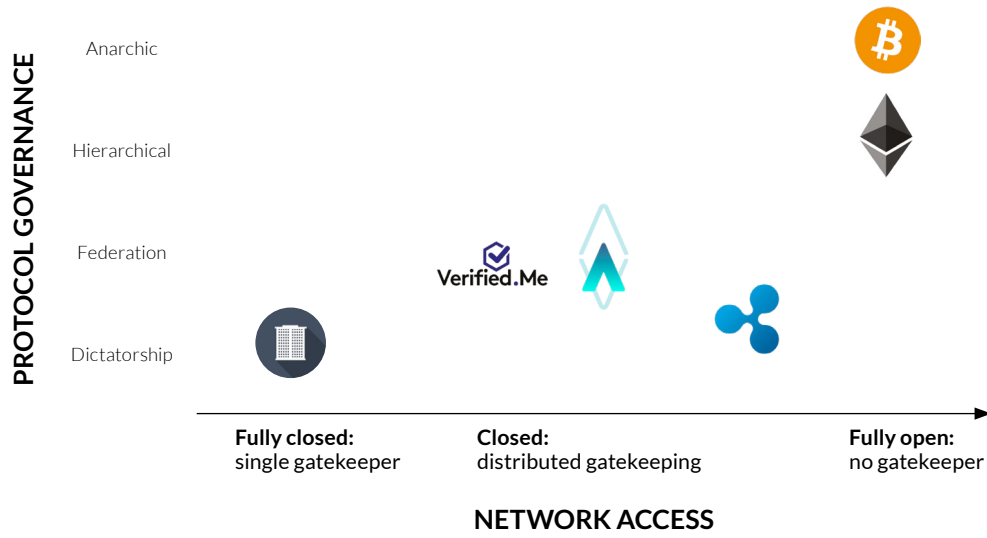
In closed systems, the administrators generally play a much more important role in the governance process. For example, protocol changes in 'Project X' are dictated by the key customer (a large logistics company) and implemented by 'Company Y'. However, there exist alternative, more collaborative models as well: protocol changes are voted upon by record producers (*validators*) in Alastria and by a Steering Committee composed of network participants in Verified.Me. In contrast, key decisions in Ripple are taken by Ripple Labs which acts as a 'benevolent dictator', although validators have the possibility to vote on so-called 'Amendments'.[107]

Table 19 provides a more detailed analysis of the key differences based on the processes at the protocol layer as defined by the framework.

Table 19: Comparative Analysis: Protocol Layer

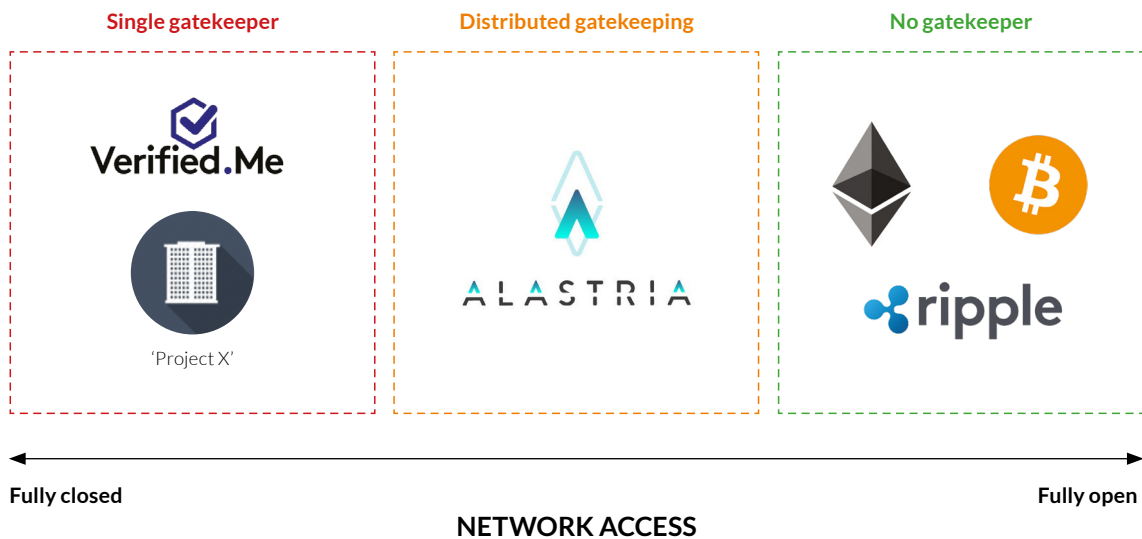| FRAMEWORK ELEMENTS			BITCOIN	ETHEREUM	RIPPLE	ALASTRIA	VERIFIED.ME	'PROJECT X'
LAYER	**COMPONENT**	**PROCESS**						
PROTOCOL	**GENESIS**	Inter-System Dependencies	Self-sufficient system	Self-sufficient system	Self-sufficient system	Self-sufficient system	Self-sufficient system	Self-sufficient system
		Codebase Creation	From scratch; open-source	From scratch; open-source	From scratch; open-source	Quorum-based codebase (which itself is a fork of Ethereum); open-source	Hyperledger Fabric; closed-source user modules	Hyperledger Fabric; closed source implementation
		Rule Initiation	Rules set by reference client	Formal protocol specification ('Yellow Paper')	Rules set by reference protocol and client	Formal protocol specification	Formal protocol specification	Formal protocol specification (default recommendation by Hyperledger)
	ALTERATION	Protocol Governance	Anarchic: nodes vote by running software client of choice; but the Bitcoin Core reference implementation has significant influence over development	Hierarchical: Ethereum Foundation and specific devs have substantial influence over general development	Dictatorship: Ripple Labs has nearly ultimate control; validators can vote via 'Amendments' feature	Democratic/Plutocratic: validator nodes need to reach agreement; no central administrator	Federation: a Steering Committee composed of providers is voting	Dictatorship: key customer has final authority
		Protocol Change	Bitcoin Improvement Proposal (BIP) via Core GitHub Repo; running software client of choice (generally Bitcoin Core)	Ethereum Improvement Proposal (EIP); running software client of choice (geth or Parity)	Validators vote on 'Amendments' - if 80% agree, changes get implemented	Unclear - likely pushed to clients who can choose whether to update or not	Technical updates get pushed to network endpoints and clients; substantial rule changes need to be performed through a formal change management process	'Company Y' will update the clients it runs for their customers

6.2.3 Network

The network level manifests a variety of important differences between the analysed systems.

Network Access

While the Bitcoin, Ethereum, and Ripple networks are universally accessible (with the two former having several thousand auditors), access to Alastria, Verified.Me, and 'Project X' is restricted to select participants (**Figure 20**).[108] Alastria is open to any Spanish business, subject to a semi-open application process in which validators vote on whether to accept new members. Alastria is expecting to onboard hundreds of companies. In contrast, access to Verified.Me and 'Project X' is controlled by a single gatekeeper. Verified.Me has around 15 service providers running fully-validating nodes, while 'Project X' is limited to three entities, with all nodes currently hosted by 'Company Y' and accessible via API calls.
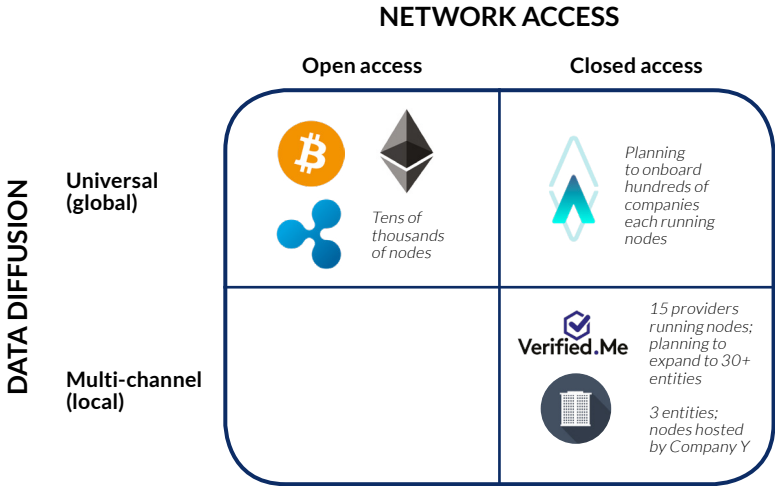
Figure 20: Network Access Comparison

Single gatekeeper   Distributed gatekeeping   No gatekeeper

Verified.Me   'Project X'   ALASTRIA   ripple

Fully closed   Fully open

NETWORK ACCESS

Communications

In Bitcoin, Ethereum, Ripple, and Alastria, data is broadcast *globally* to all nodes in the network, meaning that every node has to currently store and process every single transaction from genesis.[109] On the other hand, Verified.Me and 'Project X' are based on the Hyperledger Fabric codebase which has native support for *multi-channel* data diffusion: data is only shared among participants of a specific channel (i.e. sub-network).

Figure 21: Network Architecture Comparison

NETWORK ACCESS

	Open access	Closed access
Universal (global)	Tens of thousands of nodes	Planning to onboard hundreds of companies each running nodes
Multi-channel (local)		15 providers running nodes; planning to expand to 30+ entities / 3 entities; nodes hosted by Company Y

DATA DIFFUSION

Transaction Processing

When it comes to determining how transactions get included in the form of records to the ledger, the systems in question take different approaches.

- **Participation:**
  In a first step, we observe that once on-boarded to the network, any network participant in Bitcoin, Ethereum, Ripple, and Verified.Me has the right to participate in transaction processing as a record producer (*permissionless*).[110] This contrasts with Alastria and 'Project X', where only a select subset of network participants are authorised to become validators (*permissioned*). Alastria will launch with around 30 different validators, whereas 'Project X' currently only employs a single validator, with plans to gradually distribute control to multiple validators in the future.[111]

- **Record Creation And Conflict Resolution:**
  There are significant differences across these systems in terms of record processing: in Bitcoin and Ethereum, miners compete against each other to find a valid PoW to attach to their candidate record. In the case of two competing records, the 'longest valid

chain' rule kicks in and the ledger version that contains the most accumulated PoW is considered authoritative.[112]

In contrast, the other systems use a less resource-intensive consensus mechanism to reach agreement over the state of the ledger: Ripple uses multiple consensus rounds until a 'supermajority' of 80% is reached, while Alastria plans to launch with a rather traditional Raft-based consensus mechanism in a first stage. 'Project X' does not use a distributed consensus mechanism at all since there is only a single validator at this stage. As such, 'Project X' cannot be properly called a DLT system until it implements a consensus mechanism that involves more than one validator.

- **Incentives**
  Bitcoin and Ethereum are secured through economic incentives: miners have intrinsic monetary incentives in the form of the block subsidy (i.e. newly minted native token units) and transaction fees (denominated in the native token). Record producers in these systems operate on the basis of economic incentives summarised by the Bitcoin white paper as follows: 'He ought to find it more profitable to play by the rules [...] than to undermine the system and the validity of his own wealth'.113 As a result,

exclusive focus of record producers on economic self-interest ensures a smooth functioning of the system.

In contrast, record producers in other systems have primarily extrinsic incentives of non-monetary nature, such as reputation (e.g. being perceived as responsible), reliability (e.g. providing a good service), and the threat of

litigation in case they do not play by the rules. Verified.Me validators also have an extrinsic monetary incentive in that they receive fees (denominated in national currency) for the provision of their service. Security in these systems is primarily based on access control and contractual obligations between record producers.

Figure 22: Transaction Processing Comparison
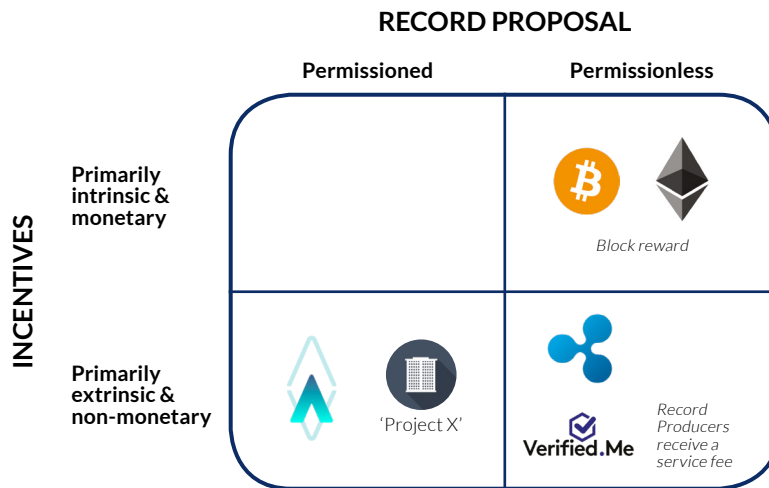
Validation

Validation is a crucial aspect that provides individual auditors with the ability to independently verify transactions, records, and the state of the system without having to rely on a third party. Since Bitcoin, Ethereum, Ripple and Alastria use the global data diffusion model, every auditor has to validate and store every single transaction and record that have ever been generated. Auditors in Verified.Me and 'Project X', on the other hand, use the multi-channel data diffusion model and thus only need to validate transactions and records within their channels (*local* verification). In the case of 'Project X', there is no formal independent validation since all nodes are currently hosted by Company Y.

Bitcoin and Ethereum only provide probabilistic finality: as a result of the PoW mechanism used for transaction processing, a

confirmed transaction runs the risk of getting reversed at any point in time. In practice, however, the likelihood of reversal decreases with each additional record added to the ledger, as reorganising the ledger requires re-doing the entire PoW for all subsequent blocks.[114] A common rule of thumb is thus to consider records 'quasi-final' after more than six (Bitcoin) and 24 confirmations (Ethereum), i.e. additional records on top of the record in question. Ripple, Alastria, Verified.Me, and 'Project X' have deterministic finality, which means that the records can be considered final after a specific provisional settlement phase. The duration of the provisional settlement phase generally differs from one system to another, although all of the case studies in question claim to have instant settlement after the record has been confirmed.[115]

Table 20 summarises the comparative analysis of the six case studies at the network layer.

Table 20: Comparative Analysis: Network Layer

FRAMEWORK ELEMENTS			BITCOIN	ETHEREUM	RIPPLE	ALASTRIA	VERIFIED.ME	'PROJECT X'
LAYER	COMPONENT	PROCESS					Verified.Me	
NETWORK	COMMUNICATIONS	Network Access	Open and unrestricted	Open and unrestricted	Open and unrestricted	Semi-open: gatekeeping distributed across validator nodes	Closed: access control performed by gatekeeper (SecureKey)	Closed: access control performed by gatekeeper (Company Y according to formal process
		Data Broadcast	Universal data diffusion (public)	Universal data diffusion (public)	Universal data diffusion (public)	Universal data diffusion (public)	Multi-channel data diffusion (selective privacy)	Multi-channel data diffusion (selective privacy); but all nodes are hosted and run by Company
		Transaction Initiation	Unrestricted: anyone with a corresponding private key can create and sign a transaction: needs to broadcast it to the network via an auditor/listener, SPV client or third-party service (API)	Unrestricted: anyone with a corresponding private key can create and sign a transaction: needs to broadcast it to the network via an auditor/listener, SPV client or third-party service (API)	Unrestricted: anyone with a corresponding private key can create and sign a transaction: needs to broadcast it to the network via an auditor/listener, SPV client or third-party service (API)	Network participants can create transactions; likely external users can transmit signed transactions as well via an auditor/listener	Restricted: select set of end users trigger transactions via an API to nodes	Restricted: key customer's ERP system creates transactions that get submitted via API to one of the auditors/listeners operated by Company Y
	TRANSACTION PROCESSING	Record Proposal	Permissionless: miners select unconfirmed transactions from their mempool and bundle them together into a candidate block. A valid candidate block requires attaching a valid SHA-256 hash to the block header (by selecting a nonce that gives the hash a sufficiently low value)	Permissionless: miners select unconfirmed transactions from their mempool and bundle them together into a candidate block. A valid candidate block requires attaching a valid Ethash PoW to the block header (by selecting a nonce that gives the hash a sufficiently low value)	Permissioned: validators select unconfirmed transactions and create a new ledger instance. They relay candidate records for a 'round of consensus': multi-computation of new ledger	Permissioned: validator nodes (± 30 different entities) select unconfirmed transactions from their mempool and bundle them together into a candidate block. Raft-based consensus mechanism to reach agreement	Permissioned: validator nodes (all 15 providers) create records that contain transactions relating only to trades they are involved in. Simple state change proposal: generally no disagreement	Permissioned: Company Y-controlled validator selects unconfirmed transactions and creates records (centralised node)
		Conflict Resolution Rule	Longest valid chain rule (i.e. most cumulative PoW)	Longest valid chain rule (i.e. most cumulative PoW)	Multiple consensus rounds among Unique Node List (UNL) until a 'supermajority' (80%) reach consensus	Race: the first block wins, competing blocks are discarded (rare since generally only one minter/leader at a time)	Generally no dispute; all participants agree that something has happened. Exact consensus algorithm used is unknown	No-op (consensus ignored): no conflict possible
		Incentivised Transaction Processing	Intrinsic and monetary:: block reward (newly minted BTC and transaction fees)	Intrinsic and monetary:: block reward (newly minted ETH and transaction fees)	No monetary reward, implicit extrinsic incentives (network robustness & resilience)	No intrinsic monetary incentive (no native token)	(1) Extrinsic monetary incentive: providers get paid a service fee by destination service, denominated in national currency ; (2) Extrinsic non-monetary incentive: (a) value creation for customers of providers that helps them compete against GAFA; (b) helps them reduce fraud	No intrinsic nor monetary incentive - extrinsic non-monetary incentive of running platform smoothly
	VALIDATION	Transaction Validation	Auditors and listeners validate every unconfirmed transaction before relaying it to connected nodes	Auditors and listeners validate every unconfirmed transaction before relaying it to connected nodes	Tracking nodes validate unconfirmed transactions before relaying them	Auditors and listeners validate every unconfirmed transaction before relaying it to connected nodes	Each auditor/listener validates every transaction within its channel	Company Y-controlled nodes validate every transaction occurring within a particular channel

6.2.4 Data

Operations

Bitcoin's inputs are generally of internal nature (e.g. previous outputs and scripts), whereas inputs in Ethereum, Ripple and Alastria originate from a combination of both internal and external sources. Verified.Me and 'Project X' have primarily external inputs sourced from external, connected systems.

Ethereum and Alastria are two systems that support general-purpose on-chain computations that can be used to design and run complex agreements and programs directly 'on-chain' (*expressive*). Applications and programs will be automatically executed at the system level - either by all fully-validating nodes (*global data diffusion*) or by those involved in that particular agreement (*multi-channel data diffusion*).

In contrast, Bitcoin, Ripple, Verified.Me and 'Project X' have rather limited capabilities for 'on-chain' computations (*prescribed*). These systems do not come with an integrated runtime environment and virtual machine (VM), which means that expressive programs cannot be executed directly at the system level. Instead, more complex computations are often processed and executed in connected but external systems. This layered approach can provide certain advantages (e.g. better scaling, increased privacy, higher security) over more expressive systems.

Reference And Value Linking

Bitcoin, Ethereum, and Ripple all keep track of endogenous system variables that only exist within the boundaries of their systems: a native digital asset (bitcoin/BTC, ether/

ETH, and ripple/XRP, respectively). Since these assets are intrinsic to the system, transfers of ownership recorded by the system can effectively be automatically enforced by the system itself without requiring the intervention of external agents.

In contrast, Verified.Me and 'Project X' are exclusively used for keeping track of exogenous system variables that reference resources and events external to the systems. For instance, records in Verified.Me contain hash pointers that point to identity data stored in external proprietary databases, whereas 'Project X' is keeping track of insurance records that exist in external ERP systems.

In addition to managing their native digital asset, Ethereum and Ripple can also be used to create records that reference exogenous resources at the system level.[116] An example would be Ripple IOUs that are issued by Ripple gateways and function as a digital representation of national currency, which is held in custody by the gateways. Transactions that involve IOUs are referencing national currency held in external systems, which requires external agents and off-chain process to enforce transfers in the 'real world'. As a result, Ethereum and Ripple can be considered *hybrid* in terms of record value linking.

Since Alastria has not formally launched yet, it is not possible to determine what the records will eventually reference. However, it is safe to assume that similar to Ethereum, network participants will take advantage of the platform versatility and create records that reference both endogenous and exogenous objects.

Table 21 provides an overview of each case studies' configurations at the data layer.

Table 21: Comparative Analysis: Data Layer

FRAMEWORK ELEMENTS			BITCOIN	ETHEREUM	RIPPLE	ALASTRIA	VERIFIED.ME	'PROJECT X'
LAYER	**COMPONENT**	**PROCESS**						
DATA	**OPERATIONS**	**Input**	Primarily internal (e.g. previous outputs: UTXOs, scripts). External: arbitrary data for timestamping purposes using OP_RETURN	Internal (previous outputs: accounts, smart contracts) and external (oracles)	Internal (previous outputs - accounts) and external (data related to IOU creation)	Internal (previous outputs: accounts, smart contracts) and external (oracles, IPFS implementation, off-chain private storage)	Primarily external (OpenID Connect (OIDC): connection service protocol). Internal = previous outputs: payload hashes; consent instruction; proof of reception	Primarily external (key customer ERP system via API; insurance records by insurance company). Internal = previous outputs/records
		Programmatically-executed Transactions	Stateless: limited scripting language enabling multi-signature and timelocked contracts	Stateful: Turing-complete smart contract language allows for general-purpose computations	Stateless: special-purpose basic computations on-chain	Stateful: Turing-complete smart contract language allows for general-purpose computations	Stateless: very simple business logic available on-chain	Stateless: very simple business logic available on-chain
		Locus of Execution	Fixed-purpose machine for running simple scripts on-chain	General-purpose virtual machine: Ethereum Virtual Machine (EVM) allows for the execution of complex computations on-chain	Fixed-purpose machine for basic on-chain computations	General-purpose virtual machine: Ethereum Virtual Machine (EVM) allows for the execution of complex computations on-chain	Business logic to manage user consent is executed at a higher layer (off-chain)	Business logic is executed on an external platform (off-chain rule engine)
	JOURNAL	**Reference**	Endogenous: native asset unique to the system (BTC)	(1) Endogenous (native asset: ETH; user-defined tokens: dApps); (2) Hybrid (collateralised tokens and/or records referencing external events)	(1) Endogenous (native asset: XRP); (2) Hybrid (gateway-issued IOUs and trust lines)	Depends on use case: endogenous if a native asset or user-defined token; hybrid if a combination of endogenous and exogenous references. Also possible to have fully exogenous references.	Fully exogenous:: identity data resides in an external system (identity sources: government, banks, etc.)	Fully exogenous: ERP systems and insurance records

6.3 COMPARING KEY DIFFERENCES ACROSS DLT SYSTEM CASE STUDIES

6.3.1 Summarising Framework Results

Figure 23: Overview Of Major Differences Between Case Studies

		Bitcoin	Ethereum	ripple	ALASTRIA	Verified.Me	'Project X'
GOVERNANCE	Anarchic	✓					
	Hierarchical		✓				
	Dictatorship			✓			✓
	Federation				✓	✓	
NETWORK ACCESS	Open	✓	✓	✓			
	Semi-open				✓		
	Closed					✓	✓
TRANSACTION PROCESSING	Decentralised	✓	✓				
	Semi-centralised			✓	✓	✓	
	Centralised						✓
INCENTIVES	Intrinsic	✓	✓				
	Extrinsic			✓	✓	✓	✓
REFERENCE	Endogenous	✓					
	Hybrid		✓	✓	✓		
	Exogenous					✓	✓

Each of the selected case studies attempts to serve different use cases and objectives, which results in a great variety of architecture and design decisions. Systems like Bitcoin are optimising for trust-minimisation and censorship resistance, which require reasonable degrees of decentralisation at all layers and processes. This comes at the expense of performance, throughput capacity, speed, scaling, and user experience, to name just a few. Moreover, the lack of centralised governance and decision-making complicates coordination among diverse network actors and slows down collective action.

Systems like Verified.Me and 'Project X' are designed to operate in a different context - a regulated multi-enterprise setting. This allows them to choose different trade-offs and adopt a more flexible approach, at the expense of a more centralised protocol and network layer. 'Project X' takes a very conservative approach in the bootstrapping phase by starting off with a system in which every function is centralised to 'put a toe into the water and get people on board'. The plan is to then gradually distribute control over these functions to more participants over time. The reasoning behind that decision is to gain a better understanding of the system's functions and properties while operating in a safe environment: this enables participants to gain invaluable experience and insights that can then be used to gradually move forward. In contrast to open experiments such as Ethereum, the rationale is to move slowly and not break things.

6.3.2 Differences In Participation

Figure 24 shows how assessing 'participation' in a DLT system using the different components can result in a more nuanced final conclusion. The systems with the most open participation are Bitcoin and Ethereum. At its current stage, 'Project X' is purposefully choosing restricted participation to bootstrap the system and initiate a learning process in a contained environment. Of the remaining three systems, Verified.Me has the lowest participation in the protocol and data Layer, though things are less certain in the network layer.

When performing a comparative analysis on highly complex and dynamic systems, choosing the accurate lense(s) is critical. Failing to take into account the diverse nature of DLT systems can lead to incomplete conclusions and assessments. As a consequence, we recommend the use of multiple lenses to get a broader - and thus likely a more complete and accurate - picture.

Figure 24: Different Levels Of 'Participation'
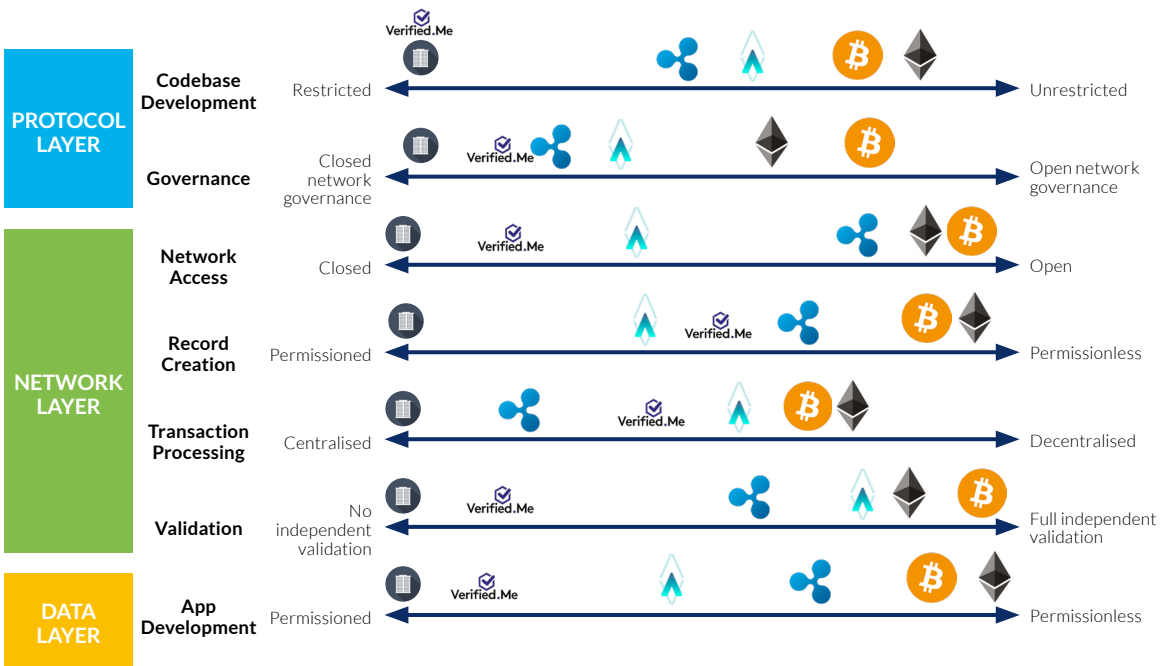
6.3.3 Exploring The Current DLT Systems Landscape

Figure 25 presents an overview of the current DLT systems landscape by mapping selected DLT systems according to three dimensions. *Transaction processing* refers to the degree of centralisation in terms of selecting transactions and adding records to the global ledger.

Reference establishes whether the records produced by system participants reference purely internal - *endogenous* (e.g. native digital assets), or entirely external - *exogenous*. The latter refers to DLT systems that are exclusively used for recordkeeping purposes (i.e. tracking information external to the system, such as items in a supply chain). An additional category represent *hybrid assets* (e.g. physical assets in tokenised form), which

share a combination of both endogenous and exogenous attributes.[117]

Network access determines the level of accessibility to the DLT system: access can be unrestricted and open to anyone or restricted to a selected group of entities that have to go through a particular selection process.

Figure 25: Current DLT Systems Landscape
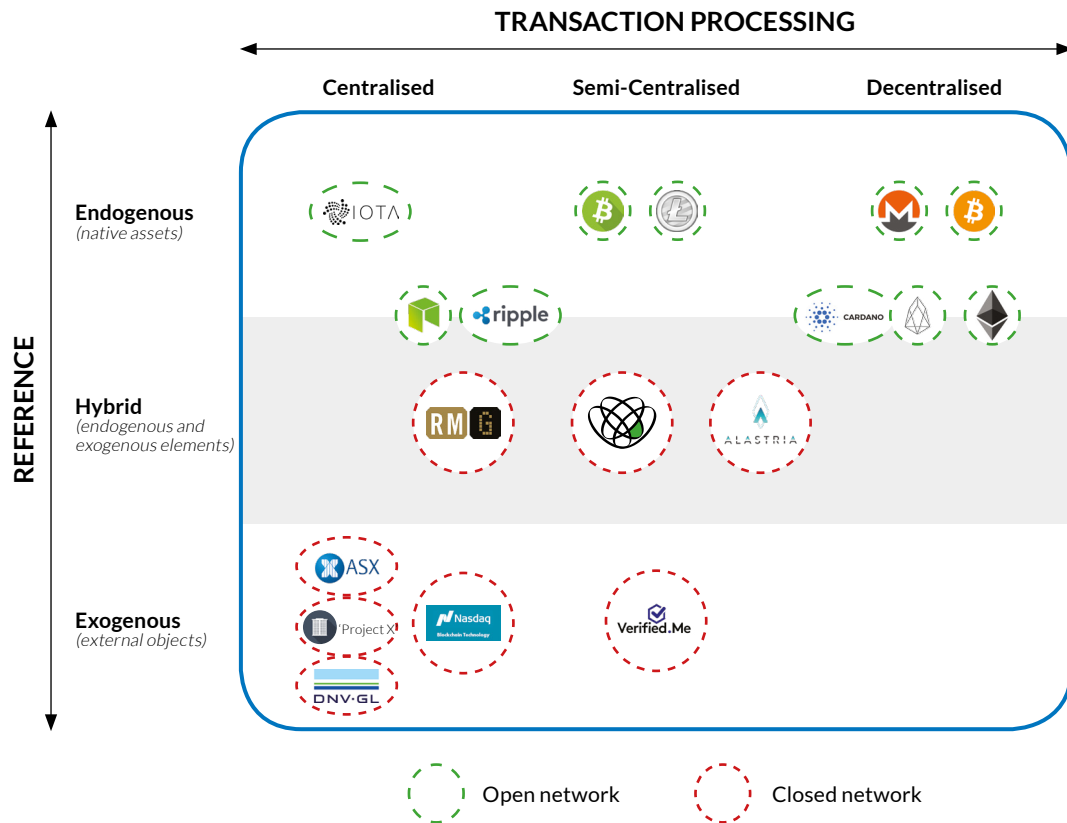
Two main observations can be derived from the landscape map:

Open networks all require a native asset (generally referred to as *cryptocurrency*) that is being used as an economic coordination mechanism to align incentives of system participants to work towards a common goal: the native asset plays an essential role in incentivising record producers to process transactions. An increasing number of open DLT systems are used for referencing non-native assets as well (e.g. Ripple, Ethereum, Cardano[118]).

On the opposite end, closed DLT systems are currently - with a few notable exceptions (e.g. World Reserve Trust[119], Royal Mint Gold[120])

- primarily being used for recordkeeping purposes to track and record external information. Enforcement can thus not be performed by these systems on their own; it is reliant on external agents.

DLT systems with open networks operate across the entire spectrum from fully centralised transaction processing to nearly decentralised transaction processing, whereas the majority of closed DLT systems currently operate in a more centralised fashion with regards to transaction processing. This does not come at a surprise as enterprises tend to take a more prudent and conservative approach when it comes to deploying new systems in production.

6.3.4 Key Design Decisions And Implications

Figure 26 summarises the key design decisions that result in distinct DLT systems. Each configuration can have specific implications on the system's characteristics, properties and nature. Furthermore, the combination of particular configurations can give rise to additional implications on the level of second-order effects: these can be difficult to predict as they often manifest themselves only post-launch.

Figure 26: Key Design Decisions And Implications

GOVERNANCE

Anarchic · Hierarchical · Dictatorship · Federation · Democratic

Potential implications on:
- Decision-making
- Ruleset
- Sustainability/antifragility
- Perceived legitimacy
- Transparency
- Outsider access
- Efficiency and coordination

NETWORK ACCESS

Open · Semi-open · Closed

Potential implications on:
- Diversity of network participants
- Choice of consensus mechanism
- Trust requirements

BROADCAST

Universal · Multi-channel

Potential implications on:
- Privacy and confidentiality
- Scalability
- Complexity

TRANSACTION PROCESSING

Resource-intensive · 'Light' on resources

Potential implications on:
- Transaction finality
- Participation
- System maintenance costs
- Degree of tamper resistance

INCENTIVES

Monetary · Non-monetary

Potential implications on:
- Nature of consensus (secured by economic incentives vs. secured by contractual agreements)
- Security

REFERENCE

Endogenous (native) · Exogenous (non-native) · Hybrid · Self-referential

Potential implications on:
- Enforcement

SECTION 7: CONCLUSION

7.1 SUMMARY

Nearly 10 years after Bitcoin entered the world, the DLT ecosystem is still in early stages: it is constantly evolving and characterised by relentless experimentation and R&D. New systems, applications, and implementations are emerging on an almost daily basis, and novel configurations, technology assumptions, and vulnerabilities are frequently presented. Despite growing interest and much progress in recent years, DLT systems are generally still considered relatively immature. Doubts about cost-benefit trade-offs and the utility of deploying such systems to solve specific problems continue to cast clouds over the ubiquitous hype that surrounds the technology.

As a result, there are many misconceptions about the nature of DLT systems, their properties, applicability for specific use cases, and remaining technological (and other) challenges. Apart from native digital assets issued on open, public and permissionless DLT systems - used primarily as speculative instruments - meaningful applications and implementations of DLT systems in production have rarely materialised to date: most projects are still in early trial or pilot phases, and it is unclear when they will be mature enough to be live. In addition, 'blockchain' and 'DLT' have become almost meaningless buzzwords that are - in many cases - mainly used for marketing and PR purposes.

For this reason, we have attempted to outline the five key properties that a DLT system needs to be capable of ensuring with no or little modification to its architecture. A DLT system is a distributed recordkeeping system that operates in an adversarial environment and is collectively maintained and updated by multiple entities. Every participant needs to be able to independently verify the validity and integrity of transactions and ultimately the system state. Finally, any attempt to tamper with transaction history needs to be trivial to detect and difficult to perform.

We find that on the basis of their current settings and configurations, many self-proclaimed 'DLT systems' do not meet these criteria and can thus only be considered 'potential DLT systems' that have the basic architectural features to allow eventual evolution into 'pure' DLT systems. In order to analyse a specific DLT system's key properties and power dynamics, we propose a conceptual framework that breaks down the system into three layers; each layer features a set of components, which in turn are composed of several processes that make the component function. We show how these components and processes interact, and how specific design choices in one process can have a significant impact on related processes and ultimately system properties. We also introduce different actor types, the roles they perform, and on which layers they are active.

We then apply the framework to six case studies of real systems that have been deployed (Bitcoin, Ethereum, Ripple, Alastria, Verified.Me and 'Project X'), and use it as an analytical tool to examine similarities and differences between these systems. We demonstrate how every design choice is a conscious trade-off between a variety of properties, and exhibit that the most common trade-off is between 'decentralisation' and performance. As a result, every DLT system is a unique culmination of multiple configuration choices and needs to be examined at an individual level.

7.2 CONTRIBUTION

The study addresses the need for a clear and shared ontology regarding DLT systems. It aims at filling the gap in common terminology by proposing a formal definition of a DLT system that involves a set of criteria it should meet. Furthermore, it provides a conceptual framework that breaks down the system into a set of system-critical layers, components and processes.

The framework serves four purposes:

- Identifying DLT systems
- Analysing existing DLT systems
- Comparing different DLT systems
- Serving as a useful tool for new system design by highlighting the trade-offs of different design choices

The key contribution consists in providing a conceptual tool for studying the components of a DLT system and understand the dependencies: before looking at the assets, tokens, or recorded information, one needs to understand the infrastructure that the former are based upon. A system is not simply 'decentralised' or 'centralised', i.e. simple binary: instead, there are different degrees of control and authority prevalent at *each* layer, component, and process. For this reason, it is of utmost importance to understand the dependencies between these layers to accurately assess what is built on top of the infrastructure.

This framework provides regulators with a clear picture of where authority - if any - is held in a DLT system, and hence who can be held accountable for the resulting technology and outcomes. For instance, it would appear that in many cases, the protocol layer is often controlled by a central authority who has the capability of modifying the rules of the game. In other cases, validating and processing transactions may be restricted to a single entity - or alternatively to a small group of intimately related entities. It is important to understand how power dynamics are distributed across different layers in a DLT system. The framework identifies the layers and participants who may be subjected to regulation, in addition to promoting general understanding of the technology.

For businesses, system engineers, and developers seeking to develop these systems in-house, the framework will prove useful as a guide to the different aspects essential to the development of new DLT systems.

For investors looking to get exposure to DLT enterprises, the framework could serve as a yardstick to understand the credibility of design proposals what different trade-offs this technology implies, and how economic value can be produced and extracted. Importantly, it will help investors make informed investment decisions, and not be misled into a 'distributed' technology that would, in fact, be centralised or incapable of meeting its design objectives.

Academics and researchers may find the framework useful as a clear foundation upon which to develop theories related to communications, economics, industrial organization, and many other disciplines.

7.3 SHORTCOMINGS AND AVENUES FOR FUTURE RESEARCH

The framework and its applications described in this report is a modular and generic tool for the analysis of DLT systems. This tool uses a three-layer analysis approach that is mostly based on qualitative interpretation. While this analysis has attempted to remain as objective as possible, the difficulty of objectively quantifying abstract aspects of DLT systems such as 'decentralisation' necessitates an inherently subjective inquiry based on the authors' assessments, conceptions of the technology stack, and role of the actors inhabiting them.

Future research on the distributed technology systems could focus on the more technical aspects of the processes described in this report. For instance, in performing case studies of DLT systems, the framework could be a first step towards more case-specific technical developments. Similarly, additional configurations for given processes could be added or developed, including new processes or components that might be necessary over the development trajectory of the technology. Additionally, regulatory and legislative research could be conducted in relation to this framework. This would help determine which process configurations correspond to what legal framework (e.g. authority, dependency).

APPENDICES

APPENDIX A: ANATOMY OF A DLT SYSTEM

Layer	Component	Process	Description
Protocol Layer	Genesis Component	Inter-System Dependencies	Investigate the dependencies of the system at the operation and/or the data level (self-sufficient, dependent, interfacing, external).
		Codebase Creation	Choose adequate codebase (existing, from scratch) as foundation of the system and set access conditions (open-source, closed-source).
		Rule Initiation	Define and agree on the rules that govern the DLT system.
	Alteration Component	Protocol Governance	Specify the decision-making process for altering the protocol in an orderly and legitimate manner.
		Protocol Change	Specify how agreed-upon rule changes will be implemented.
Network Layer	Communications Component	Network Access	Decide who to grant access to the system (open, closed).
		Data Broadcast	Specify how data is replicated (universal, multi-channel).
		Transaction Initiation	Determine who can create transactions and how these are broadcast to the system (unrestricted, restricted).
	Transaction Processing Component	Record Proposal	Select a set of unconfirmed transactions and bundle them together into a candidate record. Propose adding the candidate record to the ledger by performing the necessary steps specified by the protocol rules (e.g. attach valid PoW).
		Conflict Resolution Rule	Set the rule that solves the conflict between equally-valid proposed records for addition to the ledger (e.g. longest-chain rule).
		Incentivised Transaction Processing	Specify the incentive nature behind transaction processing (intrinsic/extrinsic, monetary/non-monetary).
	Validation Component	Transaction Validation	Confirm the legitimacy and validity of unconfirmed transactions before adding them to the log.
		Record Validation	Verify whether a record complies with protocol rules before adding it to the journal.
		Transaction Finality	Determine the transition period between 'provisional' settlement and 'permanent' settlement for confirmed records (deterministic, probabilistic).
Data Layer	Operations Component	Input	Designate the data sources used to generate ledger entries (internal, external).
		Programmatically-executed Transactions	Specify the degree of expressiveness of on-chain computations at the core system layer - often referred to as smart contract capabilities (stateless, stateful).
		Locus of Execution	Determine where computations are being executed (on-chain, off-chain).
	Journal Component	Reference	Decide what the data stored in records is pointing to (endogenous, exogenous, hybrid, self-referential).

APPENDIX B: CASE STUDY COMPARISON

LAYER	COMPONENT	PROCESS	BITCOIN	ETHEREUM	RIPPLE
PROTOCOL	**GENESIS**	Inter-System Dependencies	Self-sufficient system	Self-sufficient system	Self-sufficient system
		Codebase Creation	From scratch; open-source	From scratch; open-source	From scratch; open-source
		Rule Initiation	Rules set by reference client	Formal protocol specification ('Yellow Paper')	Rules set by reference protocol and client
	ALTERATION	Protocol Governance	Anarchic: nodes vote by running software client of choice; but the Bitcoin Core reference implementation has significant influence over development	Hierarchical: Ethereum Foundation and specific devs have substantial influence over general development	Dictatorship: Ripple Labs has nearly ultimate control; validators can vote via 'Amendments' feature
		Protocol Change	Bitcoin Improvement Proposal (BIP) via Core GitHub Repo; running software client of choice (generally Bitcoin Core)	Ethereum Improvement Proposal (EIP); running software client of choice (geth or Parity)	Validators vote on 'Amendments' - if 80% agree, changes get implemented
NETWORK	**COMMUNICATIONS**	Network Access	Open and unrestricted	Open and unrestricted	Open and unrestricted
		Data Broadcast	Universal data diffusion (public)	Universal data diffusion (public)	Universal data diffusion (public)
		Transaction Initiation	Unrestricted: anyone with a corresponding private key can create and sign a transaction: needs to broadcast it to the network via an auditor/listener, SPV client or third-party service (API)	Unrestricted: anyone with a corresponding private key can create and sign a transaction: needs to broadcast it to the network via an auditor/listener, SPV client or third-party service (API)	Unrestricted: anyone with a corresponding private key can create and sign a transaction: needs to broadcast it to the network via an auditor/listener, SPV client or third-party service (API)
	TRANSACTION PROCESSING	Record Proposal	Permissionless: miners select unconfirmed transactions from their mempool and bundle them together into a candidate block. A valid candidate block requires attaching a valid SHA-256 hash to the block header (by selecting a nonce that gives the hash a sufficiently low value)	Permissionless: miners select unconfirmed transactions from their mempool and bundle them together into a candidate block. A valid candidate block requires attaching a valid Ethash PoW to the block header (by selecting a nonce that gives the hash a sufficiently low value)	Permissioned: validators select unconfirmed transactions and create a new ledger instance. They relay candidate records for a 'round of consensus': multi-computation of new ledger
		Conflict Resolution Rule	Longest valid chain rule (i.e. most cumulative PoW)	Longest valid chain rule (i.e. most cumulative PoW)	Multiple consensus rounds among Unique Node List (UNL) until a 'supermajority' (80%) reach consensus
		Incentivised Transaction Processing	Intrinsic and monetary:: block reward (newly minted BTC and transaction fees)	Intrinsic and monetary:: block reward (newly minted ETH and transaction fees)	No monetary reward, implicit extrinsic incentives (network robustness & resilience)
	VALIDATION	Transaction Validation	Auditors and listeners validate every unconfirmed transaction before relaying it to connected nodes	Auditors and listeners validate every unconfirmed transaction before relaying it to connected nodes	Tracking nodes validate unconfirmed transactions before relaying them
DATA	**OPERATIONS**	Input	Primarily internal (e.g. previous outputs: UTXOs, scripts). External: arbitrary data for timestamping purposes using OP_RETURN	Internal (previous outputs: accounts, smart contracts) and external (oracles)	Internal (previous outputs - accounts) and external (data related to IOU creation)
		Programmatically-executed Transactions	Stateless: limited scripting language enabling multi-signature and timelocked contracts	Stateful: Turing-complete smart contract language allows for general-purpose computations	Stateless: special-purpose basic computations on-chain
		Locus of Execution	Fixed-purpose machine for running simple scripts on-chain	General-purpose virtual machine: Ethereum Virtual Machine (EVM) allows for the execution of complex computations on-chain	Fixed-purpose machine for basic on-chain computations
	JOURNAL	Reference	Endogenous: native asset unique to the system (BTC)	(1) Endogenous (native asset: ETH; user-defined tokens: dApps); (2) Hybrid (collateralised tokens and/or records referencing external events)	(1) Endogenous (native asset: XRP); (2) Hybrid (gateway-issued IOUs and trust lines)

ALASTRIA	VERIFIED.ME	'PROJECT X'
Self-sufficient system	Self-sufficient system	Self-sufficient system
Quorum-based codebase (which itself is a fork of Ethereum); open-source	Hyperledger Fabric; closed-source user modules	Hyperledger Fabric; closed source implementation
Formal protocol specification	Formal protocol specification	Formal protocol specification (default recommendation by Hyperledger)
Democratic/Plutocratic: validator nodes need to reach agreement; no central administrator	Federation: a Steering Committee composed of providers is voting	Dictatorship: key customer has final authority
Unclear - likely pushed to clients who can choose whether to update or not	Technical updates get pushed to network endpoints and clients; substantial rule changes need to be performed through a formal change management process	'Company Y' will update the clients it runs for their customers
Semi-open: gatekeeping distributed across validator nodes	Closed: access control performed by gatekeeper (SecureKey)	Closed: access control performed by gatekeeper (Company Y according to formal process
Universal data diffusion (public)	Multi-channel data diffusion (selective privacy)	Multi-channel data diffusion (selective privacy); but all nodes are hosted and run by Company
Network participants can create transactions; likely external users can transmit signed transactions as well via an auditor/listener	Restricted: select set of end users trigger transactions via an API to nodes	Restricted: key customer's ERP system creates transactions that get submitted via API to one of the auditors/listeners operated by Company Y
Permissioned: validator nodes (± 30 different entities) select unconfirmed transactions from their mempool and bundle them together into a candidate block. Raft-based consensus mechanism to reach agreement	Permissioned: validator nodes (all 15 providers) create records that contain transactions relating only to trades they are involved in. Simple state change proposal: generally no disagreement	Permissioned: Company Y-controlled validator selects unconfirmed transactions and creates records (centralised node)
Race: the first block wins, competing blocks are discarded (rare since generally only one minter/leader at a time)	Generally no dispute; all participants agree that something has happened. Exact consensus algorithm used is unknown	No-op (consensus ignored): no conflict possible
No intrinsic monetary incentive (no native token)	(1) Extrinsic monetary incentive: providers get paid a service fee by destination service, denominated in national currency ; (2) Extrinsic non-monetary incentive: (a) value creation for customers of providers that helps them compete against GAFA; (b) helps them reduce fraud	No intrinsic nor monetary incentive - extrinsic non-monetary incentive of running platform smoothly
Auditors and listeners validate every unconfirmed transaction before relaying it to connected nodes	Each auditor/listener validates every transaction within its channel	Company Y-controlled nodes validate every transaction occurring within a particular channel
Internal (previous outputs: accounts, smart contracts) and external (oracles, IPFS implementation, off-chain private storage)	Primarily external (OpenID Connect (OIDC): connection service protocol). Internal = previous outputs: payload hashes; consent instruction; proof of reception	Primarily external (key customer ERP system via API; insurance records by insurance company). Internal = previous outputs/records
Stateful: Turing-complete smart contract language allows for general-purpose computations	Stateless: very simple business logic available on-chain	Stateless: very simple business logic available on-chain
General-purpose virtual machine: Ethereum Virtual Machine (EVM) allows for the execution of complex computations on-chain	Business logic to manage user consent is executed at a higher layer (off-chain)	Business logic is executed on an external platform (off-chain rule engine)
Depends on use case: endogenous if a native asset or user-defined token; hybrid if a combination of endogenous and exogenous references. Also possible to have fully exogenous references.	Fully exogenous:: identity data resides in an external system (identity sources: government, banks, etc.)	Fully exogenous: ERP systems and insurance records

APPENDIX C: GLOSSARY

ADMINISTRATOR
Actors that controls access to the core codebase repository and can decide to add, remove and amend code to change system rules. An administrator is often considerably involved in the governance process.

CANDIDATE RECORD
A record that has not yet been propagated to the network and thus not been subject to network consensus.

CENSORSHIP RESISTANCE
Inability of a single party or cartel to unilaterally perform any of the following: 1) change rules of the system; 2) block or censor transactions; and 3) seize accounts and/or freeze balances.

CONFIRMATION
The number of records that must be reversed or overwritten to remove a transaction from the ledger state.

CONSENSUS ALGORITHM
A set of rules and processes used by the network to reach agreement and validate records.

DEVELOPER
Actor that writes and reviews code that underlies the technological building blocks of a DLT system and its connected system(s). A developer can be professionally employed or participating as volunteer contributor.

DLT SYSTEM
A system of electronic records that (i) enables a network of independent participants to establish a consensus around (ii) the authoritative ordering of cryptographically-validated ('signed') transactions. These records are made (iii) persistent by replicating the data across multiple nodes, and (iv) tamper-evident by linking them by cryptographic hashes. (v) The shared result of the reconciliation/ consensus process - the 'ledger' - serves as the authoritative version for these records.

ENDOGENOUS REFERENCE
Data which can be created and transferred solely through the means of the system and has meaning within the system. Enforcement is automatically performed by the system.

EXOGENOUS REFERENCE
Data that makes reference to some real-world condition and needs to be incorporated from the outside. This generally requires a gateway to make the connection to the external system and enforce decisions outside the DLT system.

FORK
The event of a DLT system splitting into two or more networks. A fork can occur when two or more record producers publish a valid set of records at roughly the same time, as a part of an attack (e.g. 51% attack) or when a DLT system protocol change is attempted (such a fork is 'hard' if all users are required to upgrade, otherwise it is 'soft').

GATEWAY
Actor that provides interfaces to the system by acting as a bridge between the system and the external world.

HYBRID REFERENCE
Data that shares both endogenous and exogenous characteristics. Enforcement is dependent to some extent on gateways.

INDEPENDENT VALIDATION
Ability of the system to enable each participant to independently verify the state of their transactions and integrity of the system.

JOURNAL
Ahe set of records held by a node, although not necessarily consistent with the consensus of other nodes. Journals are partial, provisional, and heterogeneous: they may or may not contain all the same records.

LEDGER

The authoritative set of records collectively held by a substantial proportion of network participants at any point in time, such that records are unlikely to be erased or amended (i.e. 'final').

LOG

An unordered set of valid transactions held by a node, which have not yet been incorporated into a formal record subject to network consensus rules (i.e. 'unconfirmed' transactions). Also called *mempool*.

MULTI-PARTY CONSENSUS

Ability of the system to enable independent parties to come to agreement on a shared set of records without requiring a central authority.

NATIVE ASSETS

The primary digital asset(s), if any, specified in the protocol that are typically used to regulate record production, pay transaction fees on the network, conduct 'monetary policy', or align incentives.

NETWORK

Interconnected actors and processes that implement the protocol.

NODE

A network participant communicating with peers over a shared communication channel.

OFF-CHAIN

Interactions, actions, and processes that occur outside of the formal system boundaries.

ON-CHAIN

Interactions, actions, and processes that occur within the system (i.e. at the system level) and are reflected in the data layer.

ORACLE

A gateway that bridges the gap between the DLT system and external systems by serving as a source of information.

PARTICIPANT

Actor interconnected with other participants in the network and communicating by passing messages among each other.

PERSISTENCE

The ability of data to remain available after the program execution, and to survive the catastrophic loss of an arbitrary number of nodes.

PROGRAMMATICALLY-EXECUTED TRANSACTION

A computer script that, when triggered by a particular message, is executed by the system. When the code is capable of operating as all parties intend, the deterministic nature of the execution reduces the level of trust required for individual participants to interact with each other. They are commonly referred to as *smart contracts* due to the scripts' ability to replace certain fiduciary relationships, such as custody and escrow, with code. However, they are not autonomous or adaptive ('smart'), nor contracts in a legal sense - rather, they can be the technological means of implementing a contract or agreement.

PROTOCOL

Set of software-defined rules that determine how the system operates.

RECORD

A bundle of transaction data which has been subject to network consensus rules and is part of the global ledger.

RECORD REORGANISATION

A node discovers that a new ledger version has been formed which excludes one or more records that the node previously thought were part of the ledger. These excluded records then become 'orphaned'.

SHARED RECORDKEEPING

The ability of the system to enable multiple parties to collectively create, maintain, and update a shared set of records.

SMART CONTRACT

See 'Programmatically-executed transaction'.

TAMPER EVIDENCE

The ability of participants to easily detect arbitrary changes to confirmed records.

TAMPER RESISTANCE

The ability to make it hard for a single party to unilaterally change past records (i.e. transaction history).

TRANSACTION

Any proposed change to the ledger; despite the connotation, a transaction need not be economic (value-transferring) in nature. Transactions can be *unconfirmed* (not included in the ledger) or *confirmed* (part of the ledger).

TRANSACTION FINALITY

Determines when a confirmed record can be considered 'final' (i.e. not reversible). Finality can be *probabilistic* (e.g. PoW-based systems that are computationally impractical to revert) or *explicit* (e.g. systems that incorporate 'checkpoints' that must appear in every transaction history). Finalised records are considered *permanently settled*, whereas records that have been produced but which are feasible to revert are referred to as *provisionally settled*.

TRANSACTION PROCESSING

The set of processes that specifies the mechanism of updating the ledger: (i) which participants have the right to update the the shared set of authoritative records (*permissionless* vs. *permissioned*) and (ii) how participants reach agreement over implementing these updates. Also called *mining*.

VALIDATION

The set of processes required to ensure that actors independently arrive at the same conclusion with regard to the state of the ledger. This includes verifying the validity of unconfirmed transactions, verifying record proposals, and auditing the state of the system.

WALLET

A software program capable of storing and managing public and private key pairs used to store and transfer digital assets.

ENDNOTES

1   Lamport, L., Shostak, R. & Pease, R. (1982) The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems*. **4** (3): 387–389.

2   Castro, M. & Liskov, B. (2002) Practical byzantine fault tolerance and proactive recovery. *ACM Trans. Comput. Syst.* **20** (4): 398-461.

3   Haber, S. & Stornetta, W. S. (1991) How to time-stamp a digital document. *Journal of Cryptology*. **3** (2): 99-111.

4   Bayer, D., Haber, S. & Stornetta, W. S. (1992) Improving the efficiency and reliability of digital time-stamping. In: Capocelli R., De Santis A., Vaccaro U. (eds) *Sequences II*. Springer: New York, NY.

5   Bertalanffy, L. v. (1949) General Systems Theory, *Biologia Generalis*, **19:** 114-129.

6   Laszlo, E. (1972) Introduction to Systems Theory: Toward a New Paradigm of Contemporary Thought. New York: Harper Torchbooks and (1973) The Rise of General Theories in Contemporary Science, *Journal for General Philosophy of Science*, **4:** 335-344.

7   Buckley, W. (1967) Sociology and Modern Systems Theory, Englewood Cliffs: Prentice-Hall.

8   Miller, J. G. (1978) Living Systems. New York: McGraw Hill.

9   We use the name 'Project X' to refer to a live DLT system in production whose operator prefers to remain anonymous.

10 World Bank Group (2017) Distributed Ledger Technology (DLT) and Blockchain. *FinTech Note No. 1*. Available at: http://documents.worldbank.org/curated/en/177911513714062215/pdf/122140-WP-PUBLIC-Distributed-Ledger-Technology-and-Blockchain-Fintech-Notes.pdf [Accessed: 28 May 2018].

11 Pinna, A. & Ruttenberg, W. (2016) Distributed Ledger Technologies in Securities Post-Trading Revolution or Evolution?. *ECB Occasional Paper No. 172*. Available at: https://www.ecb.europa.eu/pub/pdf/scpops/ecbop172.en.pdf [Accessed: 29 May 2018].

12 Davidson, S., De Filippi, F. & Potts, J. (2016) Disrupting Governance: The New Institutional Economics of Distributed Ledger Technology. Available at: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2811995 [Accessed: 29 May 2018].

13 Bank of England (2017) The economics of distributed ledger technology for securities settlement. Staff Working Paper n.670. Available at: https://www.bankofengland.co.uk/-/media/boe/files/working-paper/2017/the-economics-of-distributed-ledger-technology-for-securities-settlement.pdf?la=en&hash=17895E1C1FEC86D37E12E4BE63BA9D9741577FE5 [Accessed: 29 May 2018].

14 Tasca, P. & Tessone, C. (2018) Taxonomy of Blockchain Technologies. Principles of Identification and Classification. Available at: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2977811 [Accessed: 29 May 2018].

15 Cong, L.W. & He, Z. (2018) Blockchain disruption and smart contracts. *NBER working paper series. Working paper 24399*. Available at: http://www.nber.org/papers/w24399.pdf [Accessed: 29 May 2018].

16 Atzori, M. (2015) Blockchain Technology and Decentralized Governance: Is the State Still Necessary?. Available at: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2709713 [Accessed: 29 May 2018].

17 Okada, H., Yamasaki, S. & Bracamonte, V. (2017) Proposed classification on blockchains based on authority and incentive dimensions. *2017 19th International Conference on Advanced Communication Technology (ICACT)*. Available at: https://ieeexplore.ieee.org/document/7890159/#full-text-section [Accessed: 29 May 2018].

18 Lemieux, V. (2017) A typology of blockchain recordkeeping solutions and some reflections on their implications for the future of archival preservation. In: Big Data, *IEEE International Conference on Big Data*: 2271-2278.

19 Platt, C. (2017) Thoughts on the taxonomy of blockchains & distributed ledger technologies. *Medium*. Available at: https://medium.com/@colin_/thoughts-on-the-taxonomy-of-blockchains-distributed-ledger-technologies-ecad1c819e28 [Accessed: 29 May 2018].

20 de Kruijff, J. & Weigand, H. (2017) Understanding the Blockchain Using Enterprise Ontology. In: Dubois E., Pohl K. (eds) Advanced Information Systems Engineering. CAiSE 2017. *Lecture Notes*

in Computer Science, vol 10253. Springer, Cham. Available at: https://link.springer.com/content/pdf/10.1007%2F978-3-319-59536-8.pdf [Accessed: 29 May 2018].

21 Xu, X., Weber, I., Staples, M., Zhu, L., Bosch, J., Bass, L., Pautasso, C. & Rimba, P. (2017) A Taxonomy of Blockchain-Based Systems for Architecture Design. *2017 IEEE International Conference on Software Architecture*. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7930224 [Accessed: 29 May 2018].

22   Glaser, F. (2017) Pervasive Decentralisation of Digital Infrastructures: A Framework for Blockchain Enabled System and Use Case Analysis. *50th Hawaii International Conference on System Sciences*. Available at: https://pdfs.semanticscholar.org/859d/0535e16095f274df4d69df54954b21258a13.pdf [Accessed: 29 May 2018].

23 Glaser, F. & Bezzenberger, L. (2015) Beyond Cryptocurrencies - A Taxonomy of Decentralized Consensus Systems. *23rd European Conference on Information Systems (ECIS), 2015*. Available at: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2605803 [Accessed: 29 May 2018].

24 Bertalanffy, L. v. (1949) General Systems Theory, *Biologia Generalis*, **19:** 114-129.

25 'Strategic' actors opportunistically extract benefits from others through the misuse of a system. In contrast, 'honest' or 'non-strategic' actors use the system as intended by its designer.

26 An adversary within a system might attempt to double-spend an asset. In contrast, an adversary outside the system may attempt to control the communication networks the DLT system relies upon.

27 Nodes may display faulty behaviour that does not originate from malicious intent (e.g. hardware failure, connectivity issues).

28 For example, a payment network can be trusted to reliably transfer funds from a consumer to a merchant, but cannot guarantee that the merchant will provide the desired goods or services.

29 DLT systems do not remove trust requirements; they merely shift them from operators to users.

30 As we will see, a number of DLT systems currently operate in a closed, safeguarded environment that is void of adversarial dynamics for a variety of reasons.

31 The term 'authoritative' is used to designate the set of records that all network participants agree upon, and which are not subject to subsequent alteration without consensus.

32 'Transaction' is taken in the sense used by computer scientists: a change to the records in the system (i.e. state change).

33 Persistence refers to the ability of data to remain available after the program execution, and to survive the catastrophic loss of an arbitrary number of nodes.

34 A node is a network participant communicating with peers over a shared communication channel.

35 Tamper-evident refers to the the ability of participants to easily detect changes to records.

36 The *ideal* DLT system is Byzantine-fault tolerant, which refers to the ability of a system to remain operational even in the presence of unreliable components and where there is imperfect information about whether a component is faulty. Byzantine faults may originate from imperfect information, latency, hardware failures, or adversaries, and may originate from either unintentional error or malicious behaviour.

37 Computer scientists refer to this as 'truth', in the sense that a particular piece of data in the authoritative record exists and does not conflict with other data in the system, but the data itself is not necessarily true in the objective sense. For example, a record saying that 'Alice reports that the sky is green' would be 'truth' only in that it confirms what Alice reported, not the truth of the statement itself.

38 'Close examination discloses that blockchain technology, as implemented in Bitcoin and many other systems, does not meet the formal accounting definition of a ledger; it is a journal which provides limited netting as each transaction discloses the net remaining unspent amount ('UTXO') belonging to the private encryption key associated with the transaction. It does not organise, summarise or present a comprehensive report by category of the transaction results.' In Vagneur, K. (2018) Blockchain Distributed Technology: Governance, accounting and risk implications in the face of potential disruption. Available at: https://www.researchgate.net/publication/325541604_Blockchain_Distributed_Technology_Governance_accounting_and_risk_implications_in_the_face_of_potential_disruption? [Accessed: 04 June 2018].

39 A transaction is valid if it is properly formatted, authorised (i.e. cryptographically signed), and conforms to other standard rules determined by the protocol (e.g. equality of inputs and outputs).

40 Formally, 'the ledger' has no independent existence and is not stored anywhere uniquely; rather, it is a latent, abstract construct.

41   Records in Bitcoin are generally referred to as 'blocks'.

42   In Bitcoin, the journal/ledger uses a specific data structure called 'blockchain': records ('blocks') are cryptographically linked together as to form a chain of blocks. This data structure is used by many instances of DLT systems.

43   We discuss transaction finality in more detail in Section 5.2.3.

44   Because each node technically possesses only a journal, it is common for nodes to insist on a 'safety factor' of some number of new records before attempting to interact with the data in a particular record. This safety factor helps ensure that the node only interacts with data on the ledger; records that have achieved this safety level are called 'final'. The concept of settlement finality will be discussed in section 5.2.3.

45   This authentication and asymmetric encryption system is known as 'public-key cryptography' and was pioneered in 1976 by American cryptographers Whitfield Diffie and Martin Hellman. Using a pair of mathematically related keys (public and private), the public key is used to encrypt a message before sending it, and only the paired private key holder can decrypt the message encrypted with the public key. This way, effective security only requires keeping the private key private; the public key can be openly distributed without compromising security.

46   For instance, many custodial cryptocurrency wallets have emerged to store customer assets(or more accurately, the private keys that provide access to the assets). While many wallet providers operate secure and reliable services, some providers have been dishonest and have stolen customers' assets, while others have lacked sufficient security and lost those assets to hackers. This has frequently been misreported as a 'hack' of the cryptocurrency system itself rather than the result of improper private key management among external service businesses. Such losses do not compromise the network itself but can have a significant reputational effect upon the DLT system.

47   A software client is a computer programme which sends requests to other programmes or computers to access services made available by a server. While the term continues to be applied to computers that run 'client software' or access the 'client software' code which is located on other computers, the client and server may be separate computer programs which run on the same machine and connect via interprocess communication (IPC) techniques (i.e. mechanisms in an operating system that permit separate processes to manage shared data).

48   Auditors can store the entire transaction history since the genesis of the system (*archival nodes*) or opt for reduced storage requirements by deleting older records that have received sufficient confirmations (*pruned nodes*).

49   The term used to describe this role usually depends on the consensus algorithm of a particular DLT system. Bitcoin popularised 'miner' as a general term for block producers in blockchain systems, but others regard mining as particular to Proof-of-Work algorithms, describing Proof-of-Stake or enterprise record producers variously as 'validators', 'forgers', 'mints', or 'bakers'.

50   Examples include discussions on platforms like Reddit and GitHub, or decisions taken within a boardroom.

51   Many proposed use cases for DLT systems involve the unification of many disparate and proprietary recordkeeping systems into one universal standard, for the purpose of reducing economic transaction costs.

52   For instance, an attacker finding a vulnerability in The DAO smart contract in June 2016 that allowed to steal 3.6m ether led to the Ethereum community - spearheaded by the Ethereum Foundation - to implement an 'emergency hard fork' to recover the stolen funds. A hard fork is a software update that changes the protocol in a backwards-incompatible way: all nodes need to upgrade in order to avoid a network split. The DAO hard fork resulted in a permanent network split which led to the emergence of Ethereum Classic. Another example is the Monero system: the roadmap specifies updates to the protocol once every six months through a hard fork in order to change its proof-of-work algorithm to deter the development and use of specialised ASIC mining equipment, among other reasons.

53   Indeed, the mechanics of the BIP process implicitly defer to the reality that the approval of block producers is necessary to maintain and enforce any protocol change. A protocol change made without the support of block producers would tend to be vulnerable to attack.

54   This is not limited to software: centralisation of hardware supply (e.g. mining equipment) may render a network susceptible to control by a single party.

55   For an example of particular trade-offs, see Zamfir, V. (2018) Zamfir's Triangle. *Twitter.* Available at: https:// twitter.com/VladZamfir/status/942271978798534657 [Accessed 8 June 2018].

56   The US Securities and Exchange Commission (SEC) has indicated that they will incorporate decentralisation in their measure for whether tokens in DLT systems represent a security for regulatory purposes. They do not define, nor provide a measure for, decentralisation in this context. The first such

public statement can be found at https://www.sec.gov/news/speech/speech-hinman-061418 [Accessed: 15 July 2018].

57   Buterin, V. (2017) The Meaning Of Decentralization. *Medium*. Available at: https://medium.com/@VitalikButerin/the-meaning-of-decentralization-a0c92b76a274 [Accessed: 29 May 2018].

58   Srinivasan, B. & Lee, L. (2017) Quantifying Decentralization. *Medium*, Available at: https://news.earn.com/quantifying-decentralization-e39db233c28e [Accessed: 17 June 2018].

59   There exist proposals to increase throughput by 'sharding' (i.e. dividing the total transaction volume into subsets and allocating each subset to a smaller number of nodes for processing). In effect this would be a hybrid, as it would be designed to operate both distributed and decentralised processing.

60   This effect was a central issue in the SegWit on-/off-chain scaling debates leading to the split of the Bitcoin network and culminating in the creation of Bitcoin Cash (BCH) in 2017.

61   This loss may range from temporary (e.g. connectivity problems) to more challenging (e.g. increased regulatory scrutiny, difficult media attention) to catastrophic (e.g. natural disaster, government prohibition).

62   This loss may range from temporary (e.g. connectivity problems) to more challenging (e.g. increased regulatory scrutiny, difficult media attention) to catastrophic (e.g. natural disaster, government prohibition).

63   Proof-of-Stake (PoS) is a consensus mechanism in which a new record producer is chosen proportionally to the amount or age of coins 'staked', i.e. held by users during the election period. Tokens are usually bonded (locked up) to motivate honest behavior, and risk destruction if malicious actions are detected by the network.

64   A Proof-of-Work (PoW) is a piece of data which satisfies a set of requirements and is difficult to produce (e.g. resource- or time-consuming) but it is easy for others to verify. Producing a PoW can be designed as a low probability random cryptographic challenge which requires trial and error to produce a valid answer (e.g. Bitcoin's PoW) or it may be a true PoW which is a complex mathematical computation. In a competition to be the first to complete a random puzzle PoW, anyone has a chance to win; in a true PoW, the most powerful and fastest computer will win. PoW computations are used in programmes designed to prevent spam email (e.g. Hashcash) as well as in cryptocurrency applications.

65   Omni leverages the OP_RETURN opcode to achieve this effect. The bit strings embedded in OP_RETURN outputs are collectively understood by users of the Omni protocol as representations of assets, but to other, indifferent Bitcoin users not interacting through the Omni protocol, they look like typical transactions (albeit with some embedded metadata), and are treated as such. The outputs of these transactions are commonly called 'coloured coins'.

66   These networks are built on top of the respective platform and are based on a specific concept called 'state channels'. In essence, parties 'route' payments by exchanging signed transactions among each other off-chain, and only broadcast back to the DLT system to open or close channels. This allows for near-real-time and cost-efficient transactions by converting the base layer from a 'cash' layer to a 'settlement' layer.

67   For instance, Bitcoin does not have a formal specification; instead, the reference client 'Bitcoin Core' determines its consensus model and rules.

68   Orderliness refers to the extent to which a protocol alteration is coordinated to minimize network disruptions during the alteration process.

69   Legitimacy refers to the degree to which a protocol alteration is accepted by the community as a whole (users, holders, miners, etc.). For example, the perceived illegitimacy of the Ethereum hard fork after the DAO incident led to parts of the network remaining on the original blockchain (Ethereum Classic/ETC).

70   The process of forming consensus and implementing a protocol update is a visible - but not all-encompassing - aspect of governance in a DLT context. Not all DLT systems have formal procedures in place to decide on protocol changes. These systems rely instead on implicit and social norms that inform admission of new participants to the governance process.

71   It should be noted that some entities (or particular groups/types of entities) may still have disproportionate influence over the protocol governance process.

72   For example, a system based explicitly on *miner* consensus (as opposed to *user consensus*) would be governed primarily by a fluid set of anonymous entities, each with fluctuating degrees of influence over time.

73   This can also involve 'on-chain' voting in which network participants vote on whether to accept or reject a suggested protocol change, generally proportionate to their ownership of endogenous network resources (e.g. native assets).

74   Some protocols allow users to delegate their voting power to other users. Due to the need to prevent undue influence from Sybil identities (described in Section 5.2.1), some 'democratic' protocols may be

plutocratic in reality; alternatively, custodians may exercise 'political' rights on behalf of their customers, with or without explicit consent. Democratic/plutocratic mechanisms are generally the most diverse, sophisticated, and unproven.

75 A fork need not necessarily preserve transaction history - for example, some records or transactions may be blacklisted, as in the case of the Ethereum hard fork after the DAO incident. Similarly, not all network splits destroy aggregate network value; in cases where each subnetwork represents a distinct and conflicting vision, aggregate value may be enhanced by freeing each to pursue its vision unfettered by the other. It should also be noted that some changes to the protocol can be implemented via a soft fork rather than a hard fork. This prevents a network split: non-upgraded nodes will continue to be on the same network despite not understanding the semantics of the rule changes.

76 As noted in Section 4.1.4, 'decentralisation' does not necessarily preclude a concentration of power or influence, but rather reflects the ability of participants to route around a compromised actor. In this case, the catastrophic loss of a repository or a key contributor would not necessarily disrupt the network or its governance processes - provided that the reference code was not exclusively held by a small number of entities.

77 The infeasibility of an exit may allow for more rapid innovation. This is not always an advantage because while open-source projects are likely to be more conservative about adopting protocol changes, the ability of users to exit can help encourage developers to make decisions which are aligned with users' interests.

78 While projects like Tezos and Decred emphasize the importance of stakeholder votes in determining policy, on-chain votes alone do not constitute the entire set of governance processes. In many cases, users must necessarily coordinate off-chain to at least some extent (for example, to become aware of proposals or calls for votes. For example, Tezos boasts a foundation, a corporation, and acknowledged leaders all of which will influence decision-making, especially in the immature network. stages; Decred has an off-chain assembly, a corporation which controls pooled funds, and PoW-based validation which complements on-chain votes.

79 There remain practical limitations such as technical proficiency of the operator, equipment requirements, and connectivity/bandwidth.

80 See Platt, C. (2017) Thoughts on the taxonomy of blockchains & distributed ledger technologies. *Medium*. Available at: https://medium.com/@colin_/thoughts-on-the-taxonomy-of-blockchains-distributed-ledger-technologies-ecad1c819e28 [Accessed: 29 May 2018].

81 There are, of course, trade-offs involved: these channels are better for scaling as long as all relevant operations are performed within a particular channel. Moving records from one channel to another adds complexity and generally requires trusted gateways as a bridge between the channels.

82 The number of confirmations of a transaction is the number of records that must be reversed or overwritten to remove it from the ledger state.

83 A cryptographic hash function is computer code which takes a string of data of any length as an input and produces a fixed length string which can act as a 'fingerprint' for the provided data. Knowing the output ('hash value') does not enable someone to reconstruct the original message; only a person who knows the original message can prove the hash was created from that message. Hashing power measures the number of times a particular hashing function is computed within a given system during a specified time window (e.g. hashes per second).

84 Newer variants are in development. Delegated PoS (DPoS) schemes resemble representative voting systems by enabling users to elect record producers, with votes weighted according to the amount of assets staked. 'Proof-of-Burn' requires record producers to prove that they 'burned' some endogenous resources (e.g. tokens) by sending them to a verifiably unspendable address. This consumes no resources other than the burned underlying asset, and thus simulates the economic costs of PoW without requiring consumption of real-world resources.

85 The 'nothing-at-stake' problem describes the fact that a block producer may be able to add records to multiple subchains simultaneously, because all but one will be discarded. As a result, multiple 'histories' can persist because no record producer is incentivised to detect or resolve conflicts, opening the door for double-spending attacks. In a 'grinding' attack, a block producer can strategically choose (or create) transactions to manipulate the source of randomness used to select stakes, thereby causing the system to 'randomly' select the same record proposer repeatedly, enabling censorship.

86 Seibold, S. & Samman, G. (2016) Consensus: Immutable agreement for the Internet of Value. *KPMG Publication*. Available at: https://assets.kpmg.com/content/dam/kpmg/pdf/2016/06/kpmg-blockchain-consensus-mechanism.pdf [Accessed: 19 June 2018].

87 The longest-chain rule is also called the most-worked-chain rule, because it is sometimes possible for a series of records to be shorter despite carrying more work.

88   Transactions in a DLT system generally get validated and verified multiple times. Figure 12 provides an overview of the different phases.

89   Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J. A. & Felten, E. W. (2015) Research Perspectives and Challenges for Bitcoin and Cryptocurrencies, *36th IEEE Symposium on Security and Privacy.* Available at: http://www.jbonneau.com/doc/BMCNKF15-IEEESP-bitcoin.pdf [Accessed: 25 June 2018].

90   In Bitcoin, the provisional settlement period is typically considered 6 'confirmations' (i.e. blocks built over the transaction of interest). However, this is a choice made by each user; a node may require more (or fewer) confirmations based on its risk tolerance. Exchanges and custodians typically require 6 confirmations, while some merchants even accept '0-confirmation' transactions if they can be assured of a block being mined before a conflicting transaction is posted.

91   An eclipse attack is a situation where a node's connections are blocked or manipulated by the attacker, who then feeds the target records which are not part of the network's consensus. The effect is to cause the node to accept double-spends when it eventually rejoins the network. Eclipse attacks are generally much easier to perpetrate than 51% attacks.

92   'Settlement finality' may become a legal matter - rather than a technical distinction - which arises from agreements between network participants.

93   It should be noted that despite its popularity, usage of the term in this context is not entirely accurate because even systems considered 'stateless' have some notion of state. For example, Bitcoin's state is the entire UTXO (unconfirmed transaction output) set.

94   These terms are somewhat erroneous, as connected nodes/systems will generally be relied upon to pass transactions to initiate some function in the smart contract. Simply deploying an executable will not lead to its autonomous execution.

95   An emerging area of cryptography, called zero-knowledge proofs, is currently being investigated as a possible scaling solution for stateful systems, as it allows computation to be moved off-chain and paired with a proof that the computation was performed correctly.

96   It should be noted that not all nodes are equal. Record producers have the right to create and propose new records, whereas auditors are verifying whether these records comply with protocol rules. Record producers have a certain degree of power in terms of deciding whether to adopt or reject rule changes; auditors can be considered to have a veto right as they are ultimately deciding whether to accept or reject records submitted by record producers.

97   SegWit ('Segregated Witness') is a soft fork change that has been implemented in Bitcoin in 2017. It fixes long-standing transaction malleability issues preventing the development of applications and 'layer-2' systems. The SegWit proposal was met with resistance by some community members and eventually led to the Bitcoin Cash hard fork, which established a new system with a different scaling roadmap (primarily on-chain).

98   This generally involves referencing the coins to move (select UTXOs as *inputs*), proving that one has ownership of the coins to move (provide a *digital signature*), and specifying the necessary conditions that the recipient will have to fulfil to unlock the funds (setting the *encumbrance conditions*).

99   A valid Bitcoin PoW is achieved by finding a nonce that causes the SHA-256 hash of the block to have a sufficiently low value.

100   The 'longest-chain rule' specifies that nodes should accept the block that is built on top of the blockchain instance that has accumulated the most PoW, i.e. was hardest to produce. Contrary to the name, the longest chain in terms of the number of blocks does not always correspond to the chain with most accumulated PoW. This means that the 'longest-chain rule' terminology is confusing and actually refers to the chain with the most accumulated PoW. Thus, it is sometimes called the 'most-worked chain rule'.

101   Bitcoin (the system) has a link to the physical world via its PoW-based mining process: miners are 'burning' physical resources in the form of energy and specialised equipment to solve the cryptographic hashing puzzles which lead to the creation of new bitcoin units.

102   However, it should be noted that this only applies to one side of the trade: the system can enforce the transfers of bitcoin (*endogenous* to the system) but not the corresponding transfer of goods or services that makes up the other side of the trade (*exogenous* to the system).

103   We assume that there is no collusion between the different Bitcoin or Ethereum mining pools. If there is collusion, as some warn, then this statement is no longer true.

104   It is important to note here that because Bitcoin has no formal protocol specification, clients implement the protocol rules instead. In Bitcoin, Bitcoin Core currently functions as the reference client which dictates rules. Nevertheless, there are half a dozen competing clients that users can choose to run. This means that

any of the competing clients could implement a rule change which would then be enforced by the users running that particular client.

105 The Bitcoin network needs at least 95% to agree for a proposal to be implemented.Thus, SegWit seemed unlikely to be achieved, until key players signed an agreement that the bar for SegWit activation would be lowered. Following this, consensus for SegWit was reached and it was activated in August 2017.

106 Disagreements over how to handle The DAO smart contract bug led to the Ethereum blockchain split into two separate systems: a minority disagreed with the Foundation's plan to reverse The DAO smart contract and refused to upgrade their clients, which led to the creation of Ethereum Classic following the original Ethereum blockchain (including The DAO smart contract).

107 Currently, the default Unique Node List (UNL) is composed of a majority of Ripple-controlled validators. A successful 'Amendment' requires an 80% consensus threshold.

108 It should be noted that while access to the Ripple DLT system is theoretically open and unrestricted, it can prove difficult in practice to reliably receive and collect network data.

109 There are methods available to alleviate the storage burden by pruning older transactions that are buried deep enough under new records. The exact implementation differs across systems and implementations.

110 In practice, however, we observe that except for Verified.Me, the majority of auditors do take the role of record producers for a variety of reasons. Transaction processing ('mining') in Bitcoin and Ethereum requires substantial upfront investment into mining equipment and electricity contracts, whereas validators in Ripple need to be part of the Unique Node List (UNL) of the majority of auditors in the system in order to participate in transaction processing.

111 'Company Y' chose this conservative approach to account for unforeseen circumstances at this early stage. The goal is to test the system and make participating entities comfortable with the idea of gradually distributing control.

112 It is important to note that the 'longest-chain rule' only applies to competing system versions that share the same protocol rules. For instance, Bitcoin Cash (BCH) is 'longer' than Bitcoin (BTC) but operates on the basis of a different rule set, which invalidates the 'longest-chain rule'.

113 Nakamoto, S. (2008) Bitcoin: A Peer-to-Peer Electronic Cash System. Available at: https://bitcoin.org/bitcoin.pdf [Accessed: 21 June 2018]. The quote can be found on page 4.

114 A record (or block) reorganisation designates the ability of record producers to mine an alternative version of the 'ledger' that, provided it overtakes the original ledger, will replace the original version and invalidate all transactions contained in these records.

115 This effectively reduces the provisional settlement phase to nearly zero. However, one needs to keep in mind that changes to the protocol rules can always override transaction processing, which then impacts finality as well. In the end, 'settlement finality' is rather a legal than a technical concept that is ultimately based on social agreements between network participants.

116 Metalayers built on Bitcoin - e.g. Counterparty, Coloured Coins and Omni - are considered external systems that are dependent on Bitcoin in this context. Ethereum's ERC20 tokens and smart contracts are considered to be part of the core system since nodes are able to understand the semantics without having to run an additional client.

117 Some systems support both endogenous and hybrid assets.

118 Cardano is building a distributed computing platform based on PoS consensus. An initial version with limited functionality was released in September 2017.

119 The World Reserve Trust (WRT) is building a government-endorsed DLT system that issues a native digital currency (*SiLuBi*) acting as an intermediary currency to facilitate global trade by eliminating inefficiencies (e.g. substantial reduction in settlement period, foreign-exchange risk and transaction costs). Participants will have the ability to control and oversee platform development via a DAO governance model.

120 The Royal Mint Gold (RMG) platform is a DLT system operated by the CME Group to facilitate the issuance and trading of digital gold tokens backed by gold reserves. Each token confers direct ownership rights of physical gold securely stored and held in custody by the UK Royal Mint.

Cambridge Centre for Alternative Finance
10 Trumpington Street
Cambridge CB2 1QA
United Kingdom
Email: ccaf@jbs.cam.ac.uk
Tel: +44 (0)1223 339111