

Hyperledger Whitepaper

Abstract

This paper describes industry use cases that drive the principles behind a new blockchain platform, and outlines the basic requirements and high-level architecture based on those use cases. The design presented here describes this evolving open-source blockchain platform, called **Hyperledger**, as a protocol for business-to-business and business-to-customer transactions. Hyperledger allows for compliance with regulations, while supporting the varied requirements that arise when competing businesses work together on the same network. The central elements of this specification (described below) are smart contracts, digital assets, record repositories, a decentralized consensus-based network, and cryptographic security. Added to these blockchain staples are industry requirements for performance, verified identities, private and confidential transactions, and a pluggable consensus model.

Background

Blockchain is an emerging technology pattern that we believe can radically improve banking, supply-chain, and other transaction networks, creating new opportunities for innovation and growth while reducing the cost and risk of related business operations. With the rapid emergence of Bitcoin in the transactions domain since 2009, many businesses and industries have invested significant resources in investigating the underlying technology that powers the popular, yet controversial, cryptocurrency.

Blockchain is a peer-to-peer distributed ledger technology that first gained traction in the financial industry because of its capacity to issue, trade, manage, and service assets efficiently and securely. The distributed ledger makes it easy to create cost-efficient business networks without requiring a central point of control, in marked contrast to the world of SoR (System of Records), where every member in the ecosystem needs to maintain its own ledger system and reconcile transaction updates with one another in inefficient, expensive, and often non-standardized inter-organizational operation flows. As the shared ledger concept gains traction in the business world, blockchain **smart contracts** are also getting a lot of attention from industry [Eth]. A smart contract is a collection of business rules which are deployed on a blockchain, and shared and validated collectively by a group of stakeholders. A smart contract can automate business processes in a trusted way by allowing all stakeholders to process and validate contractual rules as a group.

Bitcoin and other cryptocurrencies were designed to be completely open, decentralized, and permissionless: anyone can participate without establishing an identity; one only has to contribute by spending computation cycles. Under the Bitcoin model of blockchain, there is no central authority that controls admission; these networks have

been called permissionless. Bitcoin is costly to operate because it requires innumerable proof-of-work computations [N09].

Hyperledger takes a much more flexible approach to consensus than the traditional blockchain model. We expect most--but not necessarily all--of our use cases to require permissioned blockchains, something that cryptocurrencies do not directly support. However, we expect that even some of the use cases of permissioned blockchains will require different consensus algorithms. For instance, round-robin consensus may be sufficient for certain small, highly trusted blockchains, while other blockchains may require Paxos or PBFT variants. For this reason, Hyperledger includes support for modular, plug-and-play consensus. This modularity gives Hyperledger the potential to save computation cycles, scale efficiently, and respond to the multitude of enterprise use case requirements by providing a secure, robust model for identity, auditability, and privacy.

Why a new blockchain?

As a fledgling technology, existing blockchain implementations have fallen short of meeting the multitude of requirements inherent in the complex world of business transactions. Scalability challenges, and the lack of support for confidential and private transactions, among other limitations, make its use unworkable for many business-critical applications. In order for the platform to be resilient to time and support requirements across the industries, it needs to be lightweight, modular and support extensibility through configuration and pluggability of various components (transaction validators, block consensus, etc.). To meet the varied demands of the modern marketplace, Hyperledger has been designed for a [broad array of industry-focused use](#)

[cases](#), thereby extending the work of the pioneers in the field by addressing the existing shortcomings.

Our vision

In Hyperledger, we have developed a vision for the future of blockchain technology. We believe that blockchain technology has the potential to fundamentally impact many aspects of our online lives, from commerce to data storage and many things in between. With this belief in mind, we think that it is important to have robust and efficient open standards for blockchain/distributed ledger technology, so that such technology can be brought forward to mainstream commercial adoption.

We believe that the future will involve a world with many interconnected distributed databases and blockchains, each of which will be specialized to suit the purposes of its users, but may also require communication with other ledgers.

Thus, we think that any open standard for blockchain technology must be as modular as possible. It must be built so that different versions of various components of a blockchain can be swapped in and out at the will of the developer. For instance, some blockchain use cases will require fast consensus algorithms that require a lot of trust, while other use cases may require less speed but also less trust. Cryptographic algorithms, smart contracts, and database storage are other features that we believe need to be “plug and play.”

The other important facet of modularity is that it facilitates outside development. If a company can improve on some module of Hyperledger, it should be possible for that company to build it and distribute as they wish. Indeed, companies or individuals should be able to build entire collections of modules (that could be required to be used together, or “plug and play” with other Hyperledger components) that fit in or interact with Hyperledger. Essentially, it should be possible to build a blockchain that uses none of the Hyperledger core components yet still resides in the Hyperledger framework.

Our long-term vision for the Hyperledger is that it contains a rich, easy-to-use API along with numerous core modules that allow for easy development and interoperability.

While we want the core Hyperledger modules to be able to satisfy as many use cases as possible, we understand that it will be impossible for the Hyperledger core to be able to handle every possible industry use case. However, it should be the case that our API is flexible enough to allow for blockchains built for these use cases outside of the Hyperledger core to easily interact with core Hyperledger components and blockchains.

We cannot possibly imagine all of the future ways that Hyperledger, and blockchain technology in general, will be used. Therefore, Hyperledger is designed to be both as modular and extensible as possible in order to accommodate these future unknown developments. In addition, the modularity of Hyperledger should enable as many people as possible to work with Hyperledger. We hope that this modularity allows people who invent or develop new technologies relevant to the blockchain to find it easy and painless to incorporate them into or use them with Hyperledger.

We believe that one of the fundamental requirements for any blockchain platform is that the identity and transactional patterns of any party on a network must be difficult for unauthorized parties to ascertain by inspecting the ledger. We also anticipate a requirement to allow blockchain users to make certain business logics and/or other parameters of a transaction confidential, rendering them inaccessible to anyone other than the stakeholders of the contract or the asset being transferred.

Hyperledger should offer support for a rich variety of applications that are easily implemented on top of the core protocol. This will necessitate support for a wide range of transaction semantics, cryptographic algorithms, consensus mechanisms, and database storage protocols. As an example, we believe that, cryptographically, Hyperledger should include all manners of encryption, signatures, and higher-functionality crypto, from simple, fast symmetric encryption, to complicated functional encryption and attribute-based signatures. These underlying technical primitives should be configurable to support elements that are important to business transactions, such as varying degrees of guaranteed transaction finality and auditability.

In summary, we want Hyperledger to be an easy-to-use, highly functional, and robust platform that anyone who is interested in building blockchain software can use as core code. While Hyperledger may fall short of this ideal functionality for every possible user

and every possible use case due to practical considerations, it is our goal to make Hyperledger come as close to this ideal as possible.

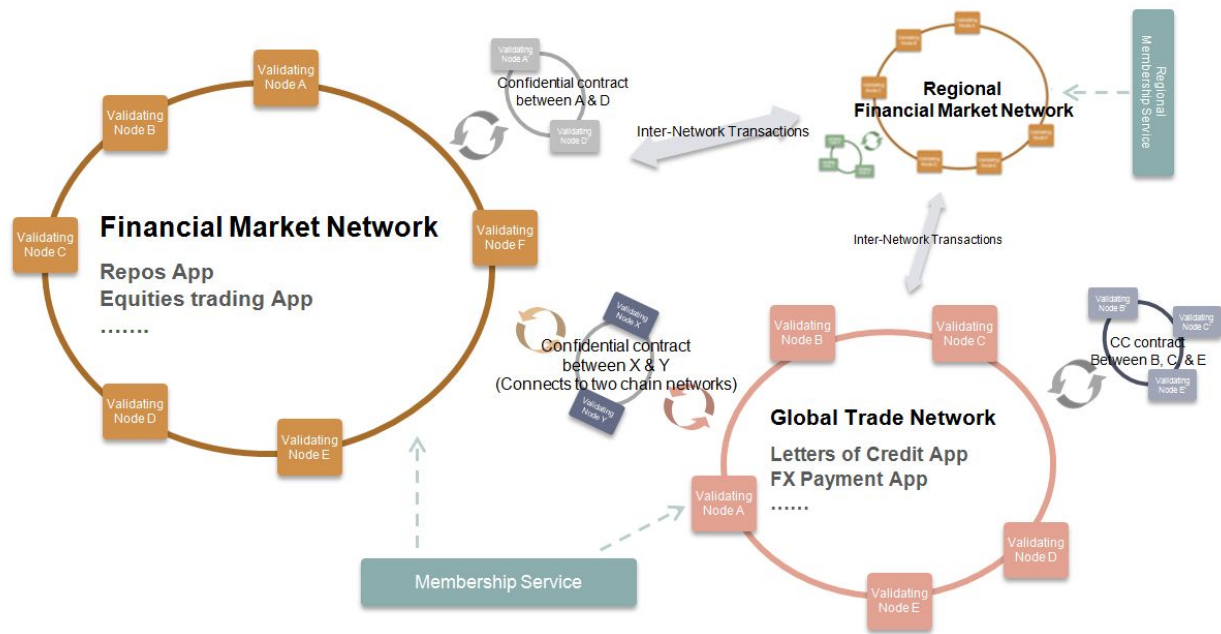


Figure 1: A world of many blockchain networks

Industry use cases

We have compiled a set of initial blockchain requirements that are considered essential for supporting the following abstract use cases. These use cases are not meant to represent the entire set of use cases for Hyperledger, but instead compose a representative sample that demonstrates some of the capabilities and features of Hyperledger.

(Note: The use cases here help guide architecture and test-driven development. While still a work in progress, the use cases should be something all contributors agree on: both in the content and the stack-ranked prioritization of them. Propose changes if you feel these miss the mark. It is ideal if there are no more than four abstract use cases, and three is preferred.)

Financial Asset depository

Assets such as financial securities must be able to be dematerialized on a blockchain network, so that all stakeholders of an asset type will have direct access to each asset, allowing them to initiate trades and acquire information on an asset without going through layers of intermediaries. Transactions can take place in a timeframe agreed upon between the stakeholders, including near real time if required, and all stakeholders must be able to access asset information in near real time. A stakeholder should be able to add business rules for any given asset type, which further reduces operating costs by implementing automation logic. The creator of the asset must be able to make the asset and any rules associated with the trading of that asset private and confidential, or public as the use case warrants. For example, the creator should be able to create an asset such that the trade history and current balance of a holder of the asset is not available to the other asset holders and perhaps not even available to the creator itself.

Corporate Action

Company A announces a voluntary corporate action event. Company A needs to ensure that complete details of the offer are sent to shareholders in real time, regardless of how many intermediaries are involved in the process (such as receiving/paying agents, CSD, ICSD, local/global custodian banks, asset management

firms, etc). Once a shareholder has made a decision, that decision will also be processed and settled (including the new issuance of shares if that's part of the corporate action event) in real time. If required, investors' responses can be kept confidential such that they can make their decisions based on the merit, without fearing coercion or retribution based on their actions.

Supply chain

The blockchain platform must provide a means to allow every participant on a supply chain network to input and track sourcing of raw materials, record parts manufacturing telemetry, track provenance of goods through shipping, and maintain immutable records of all aspects of the production and storage of a finished good through to sale and afterwards. In addition to employing both the **Business contracts** and **Asset depository** patterns described previously, this case emphasizes the need to provide deep searchability, backwards in time through many transaction layers. This requirement is at the core of establishing provenance for any manufactured good that is built from other component goods.

Master Data Management

Master data, which is usually non-transactional business information, is a key and foundational component for many industries. Having one version of truth on this core data, where authorized parties can submit changes and the designated validators accept those changes, will resolve many of data quality and integrity issues.

Sharing Economy and Internet of Things

The Sharing Economy will generate new types of revenues in many industries, including smart cities, connected homes, automotive, transportation, healthcare, retailing, construction, education, and fitness.

While transacting, however, individuals, organizations and regulators will not always trust each other. Properly implemented, distributed blockchain ledger technology will help resolve many of the trust issues that exist between various parties.

Many transactions should be settled, and status of assets should be accessible in near real time. Flexible deployment models, pluggable consensus, private transactions and confidential contracts will be important for many deployments of Hyperledger.

For more details about use cases and their requirements, and to visualize how these use cases can be plugged into a blockchain-based system, please click [here](#).

Featured requirements

We next describe some of the featured requirements of Hyperledger. While these requirements allow for many of the proposed use cases and business applications of Hyperledger, we expect that Hyperledger will evolve to have many more features than what we describe here.

The first, and perhaps most important, requirement of Hyperledger is modular structure. As we have repeatedly stated, different applications will require potentially very different

cryptographic algorithms, consensus algorithms, and database storage. However, with this in mind, we detail some more specific requirements that will be useful for many common applications.

Private Transactions and Confidential Contracts

Hyperledger should eventually support a wide variety of cryptographic tools and approaches to ensure that desired choices of confidentiality and privacy are available. Various tools for selectively revealing information should be available for identities, properties of transaction, smart contract state, and so on. These tools should not compromise privacy properties.

Some uses cases (such as IoT) would require basic confidentiality functionality that is optimized for performance which might not be appropriate for some financial use case. The crypto and consensus algorithm should handle both basic confidentiality functionality optimized for performance, as well as sophisticated algorithms for complex, bespoke cryptographical requirements.

Identity and Auditability

In addition to the existence of private transactions and confidential transactions, the well-vetted concepts of identity and auditability based on a mature public key infrastructure (PKI), completes cryptographic algorithms and allows fully-realized confidentiality on Hyperledger.

Besides the pure existence of a PKI providing current access and identity to users and relevant entities on the respective blockchain, Hyperledger is also required to provide the possibility to support a comprehensible, immutable documentation/historization of these identities - including all requirements on confidentiality around them. This is

necessary in order to be able to implement any use case around change of ownership, audit trails on document changes, etc.

In addition to positive identity, it is also important that Hyperledger offer users the ability to mask their identity in certain situations, and to only prove it when necessary (if ever). This, of course, goes well beyond the notion of traditional identity.

Furthermore, this flexible PKI allows users to adjust the strength of the cryptographic measures for their specific bespoke requirements.

Interoperability

In the loosely coupled world of many networks, separate networks don't need to know the details of how they each work. These separate networks, however, do need to have enough common ground to reliably exchange messages without error or misunderstanding. Especially with the expected future widespread use of blockchain technology, the parallel existence of a variety of blockchains needs to be taken into account. It is very likely that many use cases will span across several blockchains.

The differences in implementation of various blockchain networks, and their evolving and dynamic nature may result in a variety of highly specialized implementations. Standardized specification for inter-ledger communication will go a long way towards creating this as a common language across many networks.

Interoperability thus truly occurs when services can interact with each other despite the likely differences in design and implementation of blockchain technology. It is defined by the ability of two or more systems, or components, to exchange information, and to use the information that has been exchanged. In order to allow for envisioned broad

usability of Hyperledger across industries and use cases a functionality/ protocol allowing for interoperability between two or more blockchains is available.

Portability

The Hyperledger Project achieves portability by abstracting the value-added systems from the interfaces of its core components. For instance, smart contracts could be moved from one deployment to another without having to make any other changes.

Portability of the value-added systems, such as API libraries and GUIs for developing applications, extensions, will ultimately ensure application of such value-added systems across the many versions, implementations and deployments of the Hyperledger Project.

Portability on the infrastructure level will ultimately ensure that the Hyperledger Project functions in the same way across many heterogeneous computing platforms and network environments, which is essential to running large blockchain networks in practice.

Architecture

Figure 2 below shows the Hyperledger reference architecture aligned in four categories: Identity services, Policy services, Blockchain and Smart-contracts. These categories are a logical structure, not a physical depiction of partitioning of components into separate processes, address spaces or (virtual) machines.

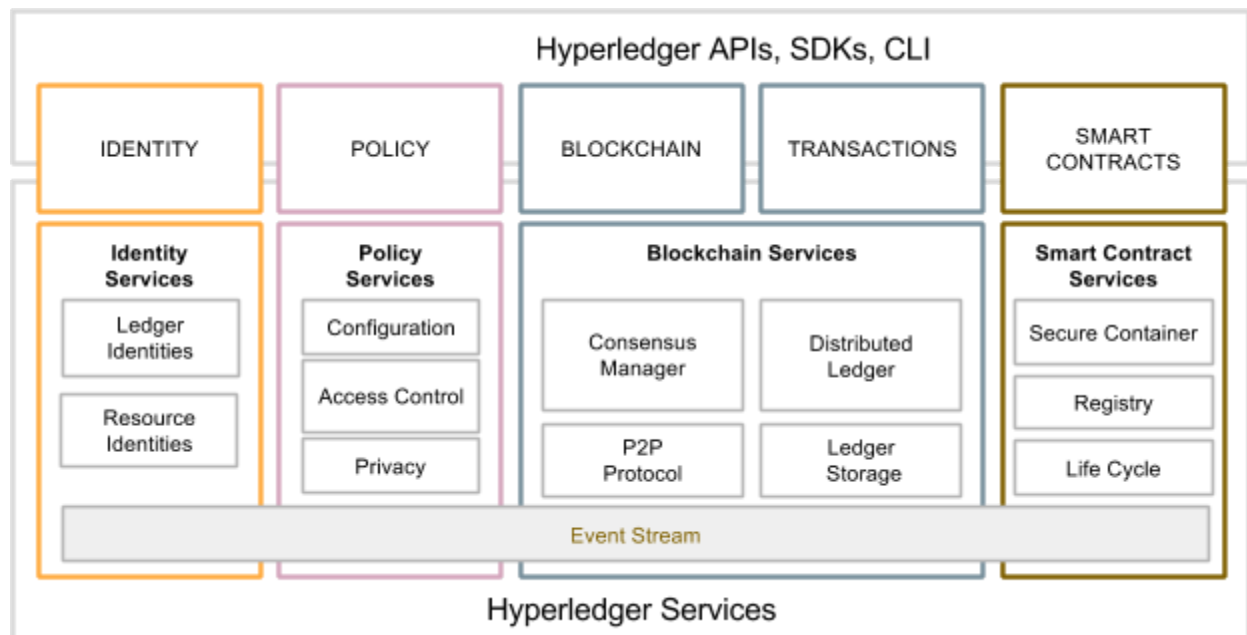


Figure 2: Hyperledger reference architecture

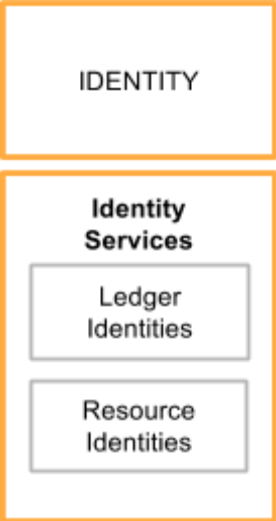
Identity services manages identities of entities, participants and ledger objects such as assets and smart-contracts.

Policy services manages access control, privacy, consortium rules, consensus rules, etc.


Blockchain services manage the distributed ledger through a peer-to-peer communication protocol. The data structures are optimized to provide efficient schemes for maintaining the world state replicated at many participants. Different consensus algorithms guaranteeing strong consistency (tolerating misbehavior with BFT, tolerating delays and outages with crash-tolerance, or tolerating censorship with proof-of-work) may be plugged in and configured per deployment.

Smart-contract services are a secured and lightweight way to sandbox the smart-contract execution on validating nodes. The environment is a "locked down" and secured container with a set of signed base images.

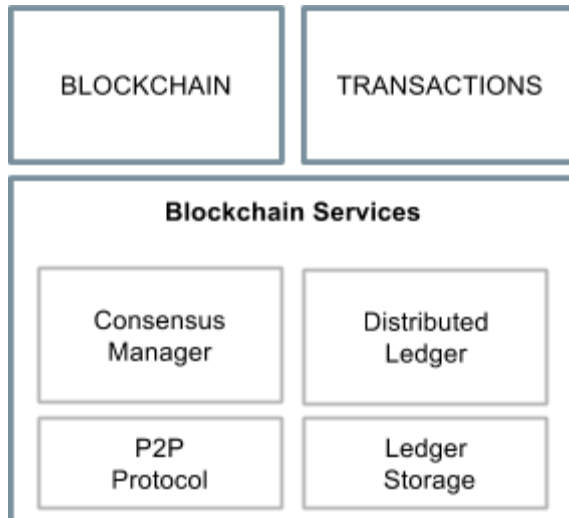
Identity Services

 <p>The diagram shows a hierarchical structure for Identity Services. At the top is a box labeled 'IDENTITY'. Below it is a box labeled 'Identity Services'. Inside the 'Identity Services' box are two sub-boxes: 'Ledger Identities' and 'Resource Identities'.</p>	<p>Identity is a pervasive requirement for the Hyperledger protocol. Identity services manage identities for participating organizations, validators, and transactors; objects contained in the ledger like assets and smart contracts; and the system components like networks, servers, and execution environments. Identity services includes a representation of the various roles that objects play in the ledger.</p>
---	---

Policy Services

 <p>The diagram shows a hierarchical structure for Policy Services. At the top is a box labeled 'POLICY'. Below it is a box labeled 'Policy Services'. Inside the 'Policy Services' box are three sub-boxes: 'Configuration', 'Access Control', and 'Privacy'.</p>	<p>The policy services function enables the configuration and management of system policies. This includes access control and authorization permissions, consortium policies which codify the agreed upon bylaws and rules for on-boarding and off-boarding of members, identity registration and verification policies, privacy, confidentiality and accountability policies, and consensus policies.</p>
---	--

BLOCKCHAIN



Blockchain services consists of three key components: Peer-to-Peer (P2P) Protocol, Distributed Ledger and Consensus Manager.

P2P Protocol provides many capabilities including bidirectional streaming, flow control, and multiplexing requests over a single connection. Most importantly, it works with existing Internet infrastructure, including firewalls, proxies and security. This component defines messages used by peer nodes, from point-to-point to multicast.

Distributed Ledger manages the blockchain and the world state by processing and validating transactions, updating and maintaining the state of ledger objects. Distributed Ledger also provides some essential, non-functional aspects such as:

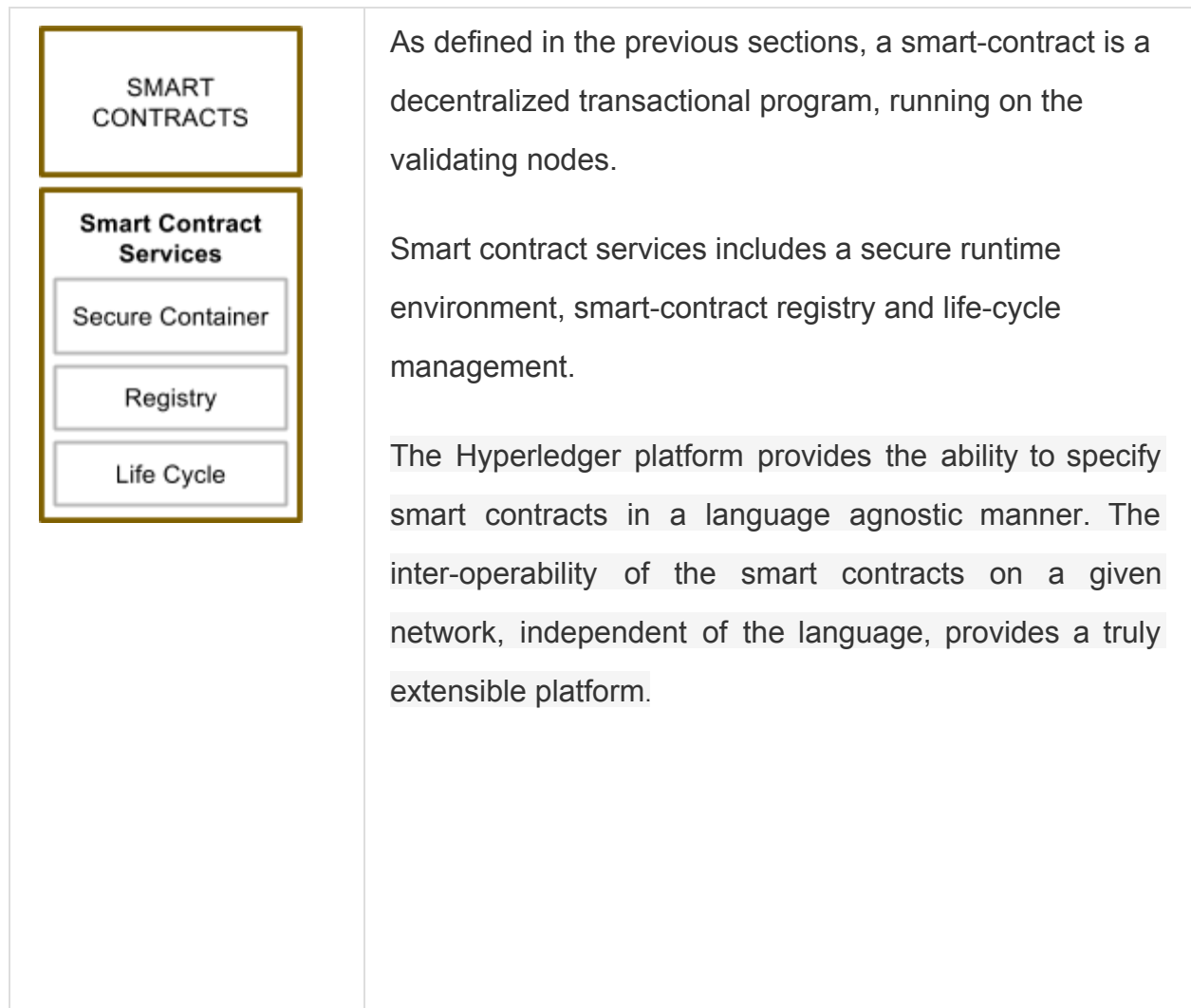
	<ul style="list-style-type: none"> • Efficiently calculate a cryptographic hash of the entire dataset after each block. • Efficiently transmit a minimal "delta" of changes to the dataset, when a peer is out of sync and needs to "catch up." • Minimize the amount of stored data that is required for each peer to operate.
--	--

Distributed Ledger uses a data store to persist the dataset, and builds an internal data structure to represent the state that satisfies the three attributes. Large files (documents, etc.) are stored in off-chain storage, not on the ledger. Their hashes can be stored on-chain as part of the transactions, which is required to maintain the integrity of files.

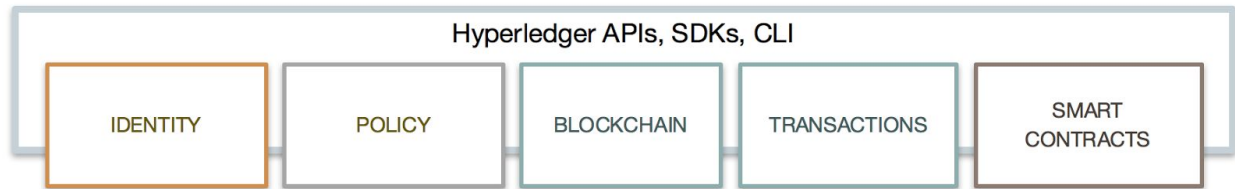
Consensus Module is responsible for confirming the correctness of all transactions in a proposed block, according to validation and consensus policies. The consensus function achieves agreement (among nodes on the network,) on the order and correctness of transactions in a block. This function interfaces to and depends on the smart-contract module to verify correctness of transactions.

By supporting a modular pluggable consensus function, Hyperledger supports the configuration of different types of consensus modules designed for specific threat/security models.

Smart-Contract



Application programming interface



An easy-to-use, flexible API is one of the cornerstones of Hyperledger. Each module type of Hyperledger has a clear, well-defined API so that Hyperledger algorithms can be used in a manner that is as “plug-and-play” as possible. For instance, there is a consensus algorithm API that ensures that people can painlessly swap consensus algorithms without impacting other parts of the code. Well-defined APIs are essential in order for Hyperledger to support the many use cases that people have developed.

In addition, the external APIs--those that are not designed for module-to-module communication--are designed to be extremely usable, so that a relatively unskilled developer can write code on top of Hyperledger without too much trouble. A totally separate API, smart contract model, and consensus, will lay the foundation upon which ecosystem participants could lay their contributions. This will ultimately allow rapid growth of the ecosystem.

Network Topology

The network topology of Hyperledger can, in principle, be quite varied: in particular, participants can use cloud services to host all kinds of peer nodes, including validating nodes, or they can run such nodes themselves. It is important to note, though, that Hyperledger runs in an agnostic manner to the underlying network structure, so who is actually running the hardware behind nodes matters little. However, if a cloud is hosting nodes, care must be taken to avoid a potentially malicious cloud server compromising

secret information, so more powerful cryptographic solutions may need to be used in this case.

Some Hyperledger deployments will experience a great degree of variability when it comes to latency of communication between nodes in the network. Network failures, node failure, overall network resiliency and recoverability should be factored into the requirements when planning deployments.

Conclusion

Hyperledger's mission is to enable mainstream industry adoption of blockchain technology. After reviewing the available blockchain solutions and hearing use cases from both industry leaders and technology evangelists, we are convinced that blockchain will be an extremely important technology pattern that could revolutionize many industries and businesses.

We have observed that industry is urgently calling for a business-ready blockchain platform that is both efficient and scalable, and offers enterprise-grade support for privacy and confidentiality. We have also discovered many different categories of use cases, each of which may require a different underlying blockchain implementation.

To fully realize the potential of blockchain technology and to create a standard that can be adopted into many different uses, we designed the Hyperledger platform to be both flexible and extensible.

Glossary

Roles & Participants

Roles

Chain Transactor	Entities that have permission to create transactions and query network data.
Chain Validator	Entities that own a stake of a chain network. Each chain validator has a voice in deciding whether a transaction is valid, therefore chain validators can interrogate all transactions sent to their chain.

Types of Network

Permissioned vs. Non-permissioned

Permissioned Network	A blockchain network collectively owned and operated by a group of identifiable and verifiable business entities.
Non-permissioned Network	A blockchain network with no identifiable ownership structure and is operated by a community of participants that may or may not be identifiable.

Types of Chains

Standard Chain	A blockchain network with many participants; each chain operates one or multiple applications/solutions validated by a group of organizations/business entities.
Confidential Chain	A special purpose chain created to run confidential business logic that is only accessible by contract stakeholders.

Transactions

Types of Transactions

Deployment Transaction	Transactions that deploy a new smart contract to a chain.
Invocation Transaction	Transactions that invoke a function on a smart contract.

Confidentiality of Transactions

Public Transaction	A transaction with its payload in the open. Anyone with access to a chain network can interrogate the details of public transactions.
Confidential Transaction	A transaction with its payload cryptographically hidden such that no one besides the stakeholders of a transaction can interrogate its content.
Confidential contract Transaction	A transaction with its payload encrypted such that only validators can decrypt them. Smart contract confidentiality is determined during deploy time. If a smart contract is deployed as a confidential smart contract, then the payload of all subsequent invocation transactions to that smart contract will be encrypted.

Inter-chain Transactions

Inter-Network Transaction	Transactions between two business networks (standard chains).
Inter-Chain Transaction	Transactions between confidential chains and main chains. Smart contracts in a confidential chain can trigger transactions on one or multiple chain(s).

Hyperledger Entities

Smart Contract

Public Smart contract	Smart contracts deployed by public transactions, these smart contracts can be invoked by any member of the network.
Confidential Smart contract	Smart contracts deployed by confidential transactions, these smart contracts can only be invoked by validating members (Chain validators) of the network.
Access Controlled Smart contract	Smart contracts deployed by confidential transactions that also embed the tokens of approved invokers. These invokers are also allowed to invoke confidential smart contracts even though they are not validators.

Ledger

Smart contract-State	HPL provides state support; Smart contracts access internal state storage through state APIs. States are created and updated by transactions calling smart contract functions with state accessing logic.
Transaction List	All processed transactions are kept in the ledger in their original form (with payload encrypted for confidential

	transactions), so that network participants can interrogate past transactions to which they have access permissions.
Ledger Hash	A hash that captures the present snapshot of the ledger. It is a product of all validated transactions processed by the network since the genesis transaction.

Node

DevOps Service	The frontal module on a node that provides APIs for clients to interact with their node and chain network. This module is also responsible to construct transactions, and work with the membership service component to receive and store all types of certificates and encryption keys in its storage.
Node Service	The main module on a node that is responsible to process transactions, deploy and execute smart contracts, maintain ledger data, and trigger the consensus process.
Consensus	The default consensus algorithm of Hyperledger fabric is called Sieve. It is a new algorithm, enhancing the “classic” PBFT mechanism in that it allows validating nodes to do a best effort in identifying non-deterministic transactions.

References

- [CL02] Miguel Castro and Barbara Liskov, [Practical Byzantine Fault Tolerance](#)
- [N09] Satoshi Nakamoto, [Bitcoin: A Peer-to-Peer Electronic Cash System](#)
- [Eth] [Ethereum Whitepaper](#)