Attachment I – Architecture Mapping of Alastria (Quorum version)

Section 1 Summary

Platform summary	
Platform ID	*Alastria (Quorum version)/ALAQ*
Status/Revision	*Mainnet for limited use cases – 1.0*
Type	*Public-Permissioned*
Domain	*Many sectors*
Description	*Alastria (Quorum version) is a country-level Public-Permissioned open-source platform for decentralized applications, compatible with Ethereum but with a proper governance model for better performance, better decentralization and better compliance with regulations.*

Section 2 Governance & Compliance Functions

Platform governance	
Governance Type	*Public-Permissioned, governed as Common-Pool Resource*
Chain Network Admin	*Non-profit association, open to anybody.* *Commissions, Working Groups and General Assembly.* *AIP – Alastria Improvement Proposal*
Pledge (cost of malicious action)	*Node compromise (standard BFT enhanced with Proactive Recovery)*
Tamper Proof (tamper cost)	*Rewriting history is impossible (assuming Ethereum digital signatures are safe), even with all consensus nodes compromised.* *Malicious actors can not invent or modify transactions, just censor them (only if more than 1/3 of nodes compromised). But this event can be detected and actions taken against the responsible (Alastria operates in an efficient legal environment).* *Any malicious action not avoided by IBFT can be audited post-facto and the responsible entity penalized accordingly.*
Description	*Alastria (Quorum version) is a Public-Permissioned network using Quorum (a light fork of Ethereum), operating in a good legal environment (inside the European Union).*

	The network is promoted by non-profit association, with a governance style "Common-Pool Resource". Any entity can join the network, but requires node permissioning. Network is collaboratively operated by the members. Anti-trust rules and no artificial or arbitrary rules of entry. The governance bodies allow the participation of any member of the association. *Being permissioned and using IBFT, Alastria (Quorum version) has "transaction finality" (required for most transactions in the productive economy), and the protocol does not allow reorgs or chain forks, so it is impossible that malicious actors can rewrite the history of the chain in a honest node.* *In addition, digital signatures are used for everything (transactions and consensus execution), so any malicious action is auditable at least post-facto. Alastria monitoring tools make any such actions transparent to the whole set of members. Incentive to cheat is extremely low.*

Platform trust endorsement policy	
Type	*Common-Pool Resources collaborative governance, coupled with Legaly binding SSI identities and transparent consensus execution monitoring tools.*
Tool	*Permissioning, IBFT, extensions to BFT (proactive recovery), digital signatures, consensus execution transparent monitoring tools.*
Policy	*Common-Pool Resources, auditability, transparency, fairness.*

Economic Model (optional)	
Price Model to Deploy Contracts and do Transactions	*Deploy a new contract is a kind of transaction,* *Charged by transactions only*
Who pays the costs of the network	*Users*
Monetary Policy of Tokens	*There is no cryptocurrency embedded in the infrastructure, and there is no mining.* *However, Gas (the same concept from Ethereum) is used both to protect the network and to meter the usage by the users. However, in Alastria (Quorum version), the Gas is not associated to any token or cryptocurrency subject to speculation.* *The incentives of the consensus nodes are long-term and associated to the Common-Pool Resource governance style (basically, all members of the association need a viable the network to run their transactions in the real economy, which is where they obtain the economic benefits).*

Rights of Tokens	*Not applicable*

Section 3 Application

Platform Smart Contract mechanism	
Language	*Same as Ethereum*
Turing Complete?	*Yes – Solidity*
Compiler	*Solidity or any other which compiles to the Ethereum VM*
Runtime VM	*EVM;*
DevTools	*Same as Ethereum*
Extra Tool(s)	*Alastria Block Explorer (Block data view)* *Alastria NetStats ()*
Lifecycle	*Same as Ethereum (the network is permissioned only when joining a node, afterwards the members are autonomous)*
Description	*It is mostly compatible with Ethereum*

Section 4 Protocol

Platform AAA Management	
Account type	*Address;*
Distributed ID	*There are two types of accounts which share the same address space: externally owned accounts and contract accounts. Externally owned accounts are controlled by public-private key pairs and have no code. Contract accounts are controlled by the code stored together with the account – the smart contract code.* *User should generate an externally owned account using a local software/hardware in order to keep the private key private;* *Contract accounts are created during deploy.*
AAA support	*N/A*
Description	*The rational is that there are so many possible addresses that the probability of collision is negligible.*

Platform Consensus Mechanism	
Algorithm	*IBFT enhanced with Proactive Recovery and Stand-by Consensus Set*
Consensus mode	*Event;*
Management solution	*Internal;*
Description	*The basic consensus algorithm is IBFT. At a given moment there is an Active Set of consensus nodes creating the blocks, and a Stand-by Set of nodes ready to be included in the Active Set. In a proactive way (even in the absence of failures), consensus nodes are rotated from the Active Set to the Stand-by Set and vice versa. When a node changes from Active Set to Stand-By set, the node is "rejuvenated" using safe techniques (eg. from safe read-only storage) to eliminate any infection made by a malicious actor while the node was in the Active Set.* *The recovery mechanism allows the system to tolerate any number of faults over the lifetime of the system provided fewer than 1/3 of the replicas become faulty within a small window of vulnerability.* *The Stand-By set increments further the robustness, inclusivity and trust on the system, by allowing any member of the association that so wishes to participate in the consensus algorithm even if it is BFT.*

Platform Ledger Management	
Model	*balance;*
Extra	*MPT support - modified Merkle Patricia tree (trie)*
Description	*Each account has a storage, a persistent memory area. A contract can neither read nor write to any storage apart from its own.* *From a block header there are 3 roots from 3 MPT: stateRoot, transactionsRoot and receiptsRoot.*

Section 5 Resources

Node Management	
Node Role	*Regular nodes, Consensus nodes and Bootstrap (permissioning) nodes.*
Joining	*For Regular nodes, the member of the association has to request permissioning for its node, which is always granted as per the rules of Alastria.The Regular nodes can use Bootstrap or other Regular nodes to sync with the network and start participating.* *For Consensus and Bootstrap nodes, the process is basically the same but the member has to agree on compliance to a much more stringent*

	Technical and Operational Policy. Any member complying with those policies can become a Consensus or Bootstrap node.
Leaving	*Regular nodes can stop working at any time.* *Consensus and Bootstrap nodes have agreed to a 24x7 operation and some other compromises to allow for stability of the network, even though there is not a standard formal SLA contract.*
Role changing	*A node can change role if they follow the stablished procedures.*
Description	-

Platform Data Storage Mechanism	
Mass storage mitigation[1]	*Concept of Gas* *Some operations may have negative gas cost, for example kill a contract.*
Decentralized Data Storage Support	*Same as Ethereum (IPFS, etc). In addition there is in the roadmap a decentralized storage system more compatible with privacy and regulation than IPFS, especially for AlastriaID (SSI).*
Data Privacy Solution	*Members can use Private Transactions provided by Quorum.*
Description	*The fundamental unit of computation is called "gas"; The fee system is to require a person to pay proportionately for every resource that they consume, including computation, bandwidth and storage; in Alastria the "gas" is not associated to any cryptocurrency or token susceptible to speculation.*

Platform Network Management	
Node Scalability	*Regular nodes: Thousands as in Ethereum.* *Consensus and Bootsrap nodes: tens to hundreds.*
Network Structure	*Distributed*
Network Discovery Protocol	*Kademlia-like;*
Byzantine Node Accepted?	*Yes*
P2P?	*Yes*
Data Exchange Protocol	*RLPx*

1 On chain storage cost much, solution/mechanism to resolve the problem of large cost of mass storage from node perspective. E.g., data maintenance, data storage and data cleaning.

Description	https://github.com/ethereum/wiki/wiki/Kademlia-Peer-Selection *RLPx transport protocol, a TCP-based transport protocol used for communication among Ethereum nodes. The protocol carries encrypted messages belonging to one or more 'capabilities' which are negotiated during connection establishment.* https://github.com/ethereum/devp2p/blob/master/rlpx.md

Section 6 Utils

Platform Messaging Mechanism	
Protocol Type	*RPC*
Description	*JSON-RPC is a stateless, lightweight remote procedure call (RPC) protocol. Primarily this specification defines several data structures and the rules around their processing. It is transport agnostic in that the concepts can be used within the same process, over sockets, over HTTP, or in many various message passing environments. It uses JSON (RFC 4627) as data format.* https://github.com/ethereum/wiki/wiki/JSON-RPC

Platform Crypto Libraries	
Secure Network Connection Type	*Communication via public Internet (TCP + UDP).*
Cipher Suites	*ECDSA (Elliptic Curve Digital Signature Algorithm) for it's public-key cryptography and KECCAK-256 for hashing* *There is a discussion about how these algorithms were implemented at:* https://ethereum.stackexchange.com/questions/71657/cipher-suites-open-source
Description	*Alastria uses Quorum, a light fork of Geth (The official Ethereum client node software), which uses UDP connection to exchange information about the P2P network. After establishing peer connections, Geth nodes exchange blockchain information via encrypted and authenticated TCP connections.*

Section 7 Operation & Maintenance

Platform system management – Node

Log	*Yes*
Monitoring	*Alastria NestStats and Alastria Block Explorer*
Description	*Any member of Alastria can see the basic network operation.* *Alastria Netstats also displays basic information abot execution of consensus algorithm by individual consensus nodes, so they can be made accountable in front of the whole community.* *Additional monitoring tools can provide more detailed information to any member wishing to control or audit consensus nodes operation (eg, by Regulators, Civil society organizations or other entities interested in transparency).*

Platform system management – Chain Network	
Permission Control	*Currently manual, in roadmap to make it automatic, thanks to participation of proper official bodies and Trust Framework (eg. any business registered in the official Business Registry of Spain can participate).*
Auditing	*Information public available in Alastria Netstats and blockchain explorer (for members and non-members)*
Supervisory Support	*N/A*
Description	*Alastria Block explorer shows information about the network and blocks, transactions, tokens, smart contracts, addresses and the history of its transactions. Any member can operate its own block explorer in parallel.*

Section 8 External Resource Management

Platform External Resource Management	
Interoperation solution	
Description	

Section 9 Extensions

Platform Extensions – optional	
[the following list can be duplicated for multiple extensions]	
Name	

Extension type[2]	
Extension mode[3]	
Solution	
Serve domain	
Description	

2 Standing from DLT system instance perspective, any extension inside the instance is marked as "internal", while any extension outside the instance is marked as "external"

3 All extension instances are equal (with similar capability and functional features), targeting for the scalability of DLT instance, marked as "horizontal"; extensions with different functional features, targeting to enforce the capability of DLT instance, marked as vertical. Extension type and mode pair(s) is/are used to describe the extension as to the whole DLT system. E.g., sharding (internal – horizontal), lightening – BTC (external – vertical), Corda Contract (internal – vertical).

Attachment II – Architecture Mapping of Ardor

Section 1 Summary

Platform summary	
Platform ID	*Ardor / ARDR*
Status/Revision	*Stable Release V2.2.5*
Type	*Public with hybrid capabilities*
Domain	*Blockchain Infrastructure, Payments, Data security*
Description	*Ardor is a multi-chain platform where Ardor, as the parent chain, provides energy-efficient proof of stake consensus for transactions across each of the platform's child chains. All child chains have their own native token and are inherently compatible with one another. Hybrid capabilities enable child chains to be either permissioned or permissionless while leveraging the public permissionless Ardor network for consensus.*

Section 2 Governance & Compliance Functions

Platform governance	
Governance Type	*Permissionless*
Chain Network Admin	*Community (Jelurida, a Swiss based company, as developer)*
Pledge (cost of malicious action)	*Stake for parent chain. Business agreement for child chain.*
Description	*When malicious activity is detected, Jelurida releases an additional version of the software under the Jelurida Public License (JPL) with the necessary security updates. Jelurida then allows the platform stake holders to choose which version represents the legitimate version of the ledger. By doing this, final decisions on how to respond to malicious activity are controlled in a decentralized manner.* *When a company wishes to launch a new child chain within the Ardor ecosystem, they must engage in a business agreement with Jelurida. Jelurida then releases an official upgrade of the Ardor software including the new child chain.*

Platform trust endorsement policy	
Type	*Tokenomics*
Tool	*ARDR*

Policy	*ARDR as stake. Child chain tokens as transaction fees. Bundlers exchange between. New child chains can only be added through a business agreement with Jelurida.* *Trust is accomplished by means of decentralization. The process of staking is called 'forging' on the Ardor network. Forging ARDR is a public process open to anyone with a balance of at least 1000 ARDR. Jelurida as the development team does not exert censorship or transaction reversal capabilities of any kind on chain.*

Economic Model (optional)	
Price Model to Deploy Contracts and do Transactions	*Contracts are deployed as Java data files or jar files to the blockchain using a data cloud transaction type.* *Charged by transactions only.*
Who pays the costs of the network	*Users and/or Businesses* *Users pay fees when interacting directly with the blockchain.* *Businesses have the ability to provide app end-users with 0 fee transactions by sponsoring child chain transaction fees using custom bundlers.* *For info:* https://ardordocs.jelurida.com/Tutorial_on_custom_bundlers_for_child_chain_transactions
Monetary Policy of Tokens	*The supply of tokens on the Ardor parent chain is finite with all 998,999,495 ARDR tokens in circulation at the time of the network's launch. There are options for adjusting the supply of assets and monetary supplies issued on specific child chains.*
Rights of Tokens	*ARDR tokens represent an entity's stake for participating in the process of "forging" with the proof of stake consensus mechanism.*

Section 3 Application

Platform Smart Contract mechanism	
Language	*Java*
Turing Complete?	*Yes*
Compiler	*Java*
Runtime VM	*JVM*
DevTools	*Any Java IDE – software includes built in support for deployment, unit testing, and debugging using IDE plug ins (IntelliJ recommended).*

	1. https://ardordocs.jelurida.com/Lightweight_Contracts 2. https://medium.com/@lyaffe/lightweight-contracts-articles-49c3032a50da 3. https://ardordocs.jelurida.com/Transaction_Vouchers
Extra Tool(s)	*Extensive set of development tools, block explorers and wallets provided out of the box and through 3rd party services.* *Block explorer 1: https://ardor.tools* *Block explorer 2: https://ardorportal.org* *Block explorer 3: https://ardor.world* *Load test:* https://www.jelurida.com/ardor-loadtest-report
Lifecycle	*Contracts are executed by specific events or when they receive a trigger transaction referencing the ID of the node responsible for running that contract. This makes it easy to upgrade, replace, and shutdown contracts. To upgrade or replace a contract, simply deploy the new contract to a node and update the reference ID in future trigger transactions. To shutdown a contract, ensure the node responsible for executing said contract is taken offline.* *https://ardordocs.jelurida.com/Lightweight_Contracts#Contract_Development* *Jelurida, the development company behind Ardor, manages enhancement proposals and formulates an extensive product roadmap.* *https://www.jelurida.com/ardor-roadmap*
Description	*Lightweight smart contracts on Ardor are stateless. Contracts are stored on chain as Java class or jar file but their execution is not part of the PoS consensus. Contracts are typically executed only by a subset of the nodes. Trust is achieved using multi-signature setup of verification and approval nodes. External oracles are allowed. See https://ardordocs.jelurida.com/Lightweight_Contracts and https://medium.com/@lyaffe/lightweight-contracts-articles-49c3032a50da*

Section 4 Protocol

Platform AAA Management	
Account type	*Address*
Distributed ID	*Ardor Account ID*
AAA support	*Full Support*
Description	*256 bit Private/Public key pair is derived from a passphrase (seed) with 128 bits of entropy. Account address is derived from the first 64 bits of the*

	public key SHA256 hash. Reed Solomon error correction is used to generate the public account address. *Authentication is supported using an account passphrase from which the account public key and id are derived* *For the permissioned version only an account authorization level is determined using on chain messages set from a root trust introduced by the Genesis block and standard role-based authentication.* *Account can be placed under account control of possibly one or more other accounts to confirm or reject every transaction submitted by this account.*

Platform Consensus Mechanism	
Algorithm	*PoS*
Consensus mode	*Event*
Management solution	*Internal*
Description	*Chance of generating the next block depends on your account balance i.e. stake.* *Pseudo random algorithm is used to assign time window for each block generator depending on its public key and the public key of the previous block generator hashed together.* *Stake must be stable for ~24 hours to be counted towards block generation to prevent manipulation.* *For security considerations see https://medium.com/@lyaffe/proof-of-stake-articles-cc6fbb346184*

Platform Ledger Management	
Model	*Balance*
Extra	*Child chains, Messages, Properties, Controls, Voting, Tokens, Marketplace, Cloud Data*
Description	*Rich state is maintained by the blockchain for each account including the main token balance, child chain token balances, auxiliary token balances, poll and voting state, account properties, on chain messages, marketplace balances, and more.*

Section 5 Resources

Node Management	
Node Role	*Full validating nodes and archival nodes*

Joining	*Setting up a new node does not require any permission. When started, a node will attempt to connect to a list of bootstrap nodes provided with the product installation. Once connected to another node it will query the remote node for additional nodes until reaching a configured number of open connections to remote nodes. A node operator can modify the list of bootstrap nodes before starting their node and connect/disconnect/blacklist nodes during runtime.*
Leaving	*Nodes may discontinue participation on the network at any time*
Role changing	*Nodes can independently change roles at any time*
Description	---

Platform Data Storage Mechanism	
Mass storage mitigation[1]	*Fee-based data storage pricing mitigates against mass on-chain data storage.*
Decentralized Data Storage Support	*Platform specific*
Data Privacy Solution	*Built-in encrypted messages based on AES* *On chain 'shuffling' to improve privacy is available on mainnet* *On-going Zero Knowledge Proof (ZKP) and Homomorphic Signatures research* *See https://ardordocs.jelurida.com/Arbitrary_messages and https://ardordocs.jelurida.com/Coin_Shuffling and* https://www.jelurida.com/standby-shuffling-explained
Tamper Proof (tamper cost)	*Immutable ledger. Storing encrypted data is supported.* *51% of the stake is needed in order to censor data. Transparent Forging provides higher protection. Since the identity of the next block generator can be determined with high probability, parties can send transactions directly to the block generator node or blacklist this node if it does not follow the consensus rules.*
Description	*Every transaction can have a plain text or encrypted attachment that can be used to store data, and also a dedicated "cloud data" transaction type exists. The network provides a mechanism to automatically request and restore pruned data from archival nodes that have stored it.*

Platform Network Management

[1] On chain storage cost much, solution/mechanism to resolve the problem of large cost of mass storage from node perspective. E.g., data maintenance, data storage and data cleaning.

Node Scalability	*No enforced limit*
Network Structure	*Distributed; Flexible*
Network Discovery Protocol	*Starting from a list of bootstrap nodes. Adding nodes based on nodes known to remote nodes.*
Byzantine Node Accepted?	*Yes*
P2P?	*Yes*
Data Exchange Protocol	*Gossip*
Description	*Data exchange between nodes relies on efficient peer to peer native socket communication network.*

Section 6 Utils

Platform Messaging Mechanism	
Protocol Type	*Proprietary*
Description	*Plain text messages and encrypted messages are supported. Messages can be prunable i.e. removed from the blockchain only leaving its hash, message file attachments are supported. Every transaction type can have a message attachment or encrypted "message to self" / memo attachment.*

Platform Crypto Libraries	
Secure Network Connection Type	*SSL; TLS; …*
Cipher Suites	*EC-KCDSA Curve25519*
Description	*Key exchange is based on the Curve25519 algorithm, which generates a shared secret key using a fast, efficient, high-security elliptic-curve Diffie-Hellman function. The algorithm was first demonstrated by Daniel J. Bernstein in 2006. Java-based implementations were reviewed by cryptography expert back in March, 2014.* *Message signing in Nxt is implemented using the Elliptic-Curve Korean Certificate-based Digital Signature Algorithm (EC-KCDSA), specified as part of IEEE P1363a by the KCDSA Task Force team in 1998.* *Both algorithms were chosen for their balance of speed and security for a key size of only 32 bytes.*

Section 7 Operation & Maintenance

Platform system management – Node

Log	*Yes*
Monitoring	*Yes*
Description	*Node operation can be controlled by the local operator or monitored by a public service like https://ardor.peerexplorer.com/ , nodes use extensive logging facility and advanced monitoring APIs.* *https://ardordocs.jelurida.com/API*

Platform system management – Chain Network	
Permission Control	*No permission is required*
Auditing	*Extensive audit is supported*
Supervisory Support	*Node automatic update is not supported as this would create a form of centralization. Node upgrade by itself is very simple and can be performed in batch by the node operator. Docker template is available.*
Description	

Section 8 External Resource Management

Platform External Resource Management	
Interoperation solution	*Transactional child chains are operational on mainnet since 1 January 2018.* *Unique transaction vouchers architecture enables secure offline communication of transaction data.* *Stateless Lightweight Contracts, Ardor's version of smart contracts, can be deployed on a child chain and used as clients to any external system which expose Java, Web Service, Restful or Http based APIs.* *Unlike other blockchain contract designs, the Ardor lightweight contracts are designed to interface and inter-operate with other external systems. Lightweight contracts are also designed to simplify contract life cycle management.*
Description	*Ardor is the first platform to release a fully operational multi-chain architecture to production since January 2018. Ardor's multi-chain architecture improves scalability, reduces blockchain bloat and provides several operational advantages like sponsored transaction fees.* *The new lightweight smart contract framework also offers a stateless solution for app creation on the blockchain.*

Section 9 Extensions

Platform Extensions - optional	
[the following list can be duplicated for multiple extensions]	
Name	*Child chains*
Extension type[2]	*Internal*
Extension mode[3]	*Horizontal*
Solution	*[internal] child-chain*
Serve domain	*Horizontal: sharding*
Description	*Transactional child chains receive their security from the POS consensus on the decentralized Ardor "parent chain." Each child chain is connected to the Ardor parent chain by "bundlers." Bundlers operate as mini-exchanges – they collect fees on child chains in the native child chain token, and pay ARDR to validate the transactions on the Ardor parent chain. Anyone with 1000 ARDR or more can set up a bundler. Bundlers can be customized so they only package transactions originating from specific accounts or involving specific digital assets or monetary supplies. This design means child chain owners can sponsor their end user fees. Additionally, the Ardor network is never monopolized by an individual application since each block on the Ardor parent chain has a limit to the number of transactions it will bundle per child chain. Additionally, all child chains are compatible with one another.*

Platform Extensions - optional	
[the following list can be duplicated for multiple extensions]	
Name	*Stateless lightweight smart contracts*
Extension type[4]	*Internal*

[2] Standing from DLT system instance perspective, any extension inside the instance is marked as "internal", while any extension outside the instance is marked as "external"

[3] All extension instances are equal (with similar capability and functional features), targeting for the scalability of DLT instance, marked as "horizontal"; extensions with different functional features, targeting to enforce the capability of DLT instance, marked as vertical. Extension type and mode pair(s) is/are used to describe the extension as to the whole DLT system. E.g., sharding (internal – horizontal), lightening – BTC (external – vertical), Corda Contract (internal – vertical).

[4] Standing from DLT system instance perspective, any extension inside the instance is marked as "internal", while any extension outside the instance is marked as "external"

Extension mode[5]	*Vertical*
Solution	*[external] secure oracles that interface with off-chain databases*
Serve domain	*Vertical: storage*
Description	*Stateless Lightweight Contracts, Ardor's version of smart contracts, can be deployed on a child chain and used as clients to any external system which expose Java, Web Service, Restful or Http based APIs.* *Unlike other blockchain contract designs, the Ardor lightweight contracts are designed to interface and inter-operate with other external systems. Lightweight contracts are also designed to simplify contract life cycle management.*

[5] All extension instances are equal (with similar capability and functional features), targeting for the scalability of DLT instance, marked as "horizontal"; extensions with different functional features, targeting to enforce the capability of DLT instance, marked as vertical. Extension type and mode pair(s) is/are used to describe the extension as to the whole DLT system. E.g., sharding (internal – horizontal), lightening – BTC (external – vertical), Corda Contract (internal – vertical).

Attachment III – Architecture Mapping of BTC

Section 1 Summary

Platform summary	
Platform ID	*Bitcoin / BTC*
Status/Revision	*Core Version 0.18.0*
Type	*Public*
Domain	*Peer to peer payments, Financial*
Description	*Bitcoin is the first global, open source peer-to-peer decentralized monetary system. It is based on the original Bitcoin white paper published by the anonymous Satoshi Nakamoto.* https://bitcoin.org/bitcoin.pdf https://www.lopp.net/bitcoin-information.html https://bitcoin.org/en/release/v0.18.0#wallet-gui

Section 2 Governance & Compliance Functions

Platform governance	
Governance Type	*Permissionless*
Chain Network Admin	*Community (public)* *Bitcoin Improvement Proposal (BIP)*
Pledge (cost of malicious action)	*Resources (hardware + electricity) – measured by hash rate (H/s)*
Tamper Proof (tamper cost)	>50% of network H/s
Description	*"Bitcoin Core" is the main implementation of the node software and acts as the de-facto protocol specification. Bitcoin Core is an open governance model where everyone is free to propose and discuss changes to the system through BIP.* *A BIP is a design document for introducing features or information to Bitcoin. This is the standard way of communicating ideas since Bitcoin has no formal structure.*

	https://bitcoin.org/en/bitcoin-core/contribute/ *https://github.com/bitcoin/bips* *https://en.bitcoin.it/wiki/Bitcoin_Improvement_Proposals*

Platform trust endorsement policy	
Type	*Tokenomics*
Tool	*BTC*
Policy	*Schelling point, mechanism design with Proof of Work consensus, bounded rationality, specialised ASICs as a grim trigger policy*

Economic Model (optional)	
Price Model to Deploy Contracts and do Transactions	*Bitcoin supports a limited set of smart contract functionalities. These are charged per transaction.*
Who pays the costs of the network	*Users*
Monetary Policy of Tokens	*Finite supply of BTC: 21,000,000 BTC* *No pre-mine.* *Currently 12.5 new BTC are minted per block as rewards for miners. The number of new BTC minted per block halves every 210,000 blocks, approximately every 4 years. Next halving will occur around June 2020.*
Rights of Tokens	*N/A*

Section 3 Application

Platform Smart Contract mechanism	
Language	*C++*
Turing Complete?	*No*

Compiler	*N/A*
Runtime VM	*N/A*
DevTools	*Bitcoin Script IDE and list of other DevTools and Resources* https://www.lopp.net/bitcoin-information/developer-tools.html
Extra Tool(s)	*List of websites providing Bitcoin network statistics* https://www.lopp.net/bitcoin-information/statistics-metrics.html
Lifecycle	*N/A*
Description	*N/A*

Section 4 Protocol

Platform AAA Management	
Account type	*UTXO*
Distributed ID	*There is no identification system attached to wallet addresses.*
AAA support	*N/A*
Description	*N/A*

Platform Consensus Mechanism	
Algorithm	*SHA-256*
Consensus mode	*Hashcash Proof of Work (PoW)*
Management solution	*Internal*
Description	*Bitcoin uses the hashcash Proof_of_work function as the mining core. All bitcoin miners (whether CPU, GPU, FPGA or ASICs) are expending their effort creating hashcash proofs-of-work which act as a vote in the blockchain evolution and validate the blockchain transaction log.* *More information may be found here:* *https://en.bitcoin.it/wiki/Proof_of_work*

	https://en.bitcoin.it/wiki/Hashcash

Platform Ledger Management	
Model	*Balance*
Extra	*Merkle tree*
Description	*Each block contains a list of transactions that it validates. The header contains, among other things, (i) the root of the merkle tree of these transactions, (ii) the hash of the previous block, a "nonce" number that the miners can arbitrarily set, and (iii) the hash of the block itself.* *The hash of the block itself must be below a certain difficulty target. The process of finding a nonce producing a block hash below a certain difficulty target is what makes proposing a new block difficult.* *Due to hashing function (SHA256) characteristics, there is no other way than to use brute force to find the nonce until a satisfying block hash is found, giving a statistical "proof of work". On average, N-different hashes will have to be tried by all miners to find a single satisfactory result.*

Section 5 Resources

Node Management	
Node Role	*Full mining validating nodes and full non-mining validating nodes*
Joining	*No permission is required for joining the network. One can simply set up a node and begin the Initial Block Download (IBD)*
Leaving	*Nodes can discontinue operation at any time.*
Role changing	*Nodes can independently change roles at any time.*
Description	*Full node info and basic hardware requirements: https://bitcoin.org/en/full-node#what-is-a-full-node*

	IBD info: https://bitcoin.org/en/full-node#initial-block-downloadibd

Platform Data Storage Mechanism	
Mass storage mitigation[1]	*N/A*
Decentralized Data Storage Support	*N/A*
Data Privacy Solution	*N/A*
Tamper Proof (tamper cost)	*N/A*
Description	*N/A*

Platform Network Management	
Node Scalability	*No upper bound*
Network Structure	*Distributed*
Network Discovery Protocol	*TCP*
Byzantine Node Accepted?	*Yes*
P2P?	*Yes*
Data Exchange Protocol	*Gossip;*
Description	*More information may be found here:* *https://en.bitcoin.it/wiki/Network*

Section 6 Utils

Platform Messaging Mechanism	
Protocol Type	*N/A*
Description	*N/A*

Platform Crypto Libraries

[1] On chain storage cost much, solution/mechanism to resolve the problem of large cost of mass storage from node perspective. E.g., data maintenance, data storage and data cleaning.

Secure Network Connection Type	*SSL; TLS.*
Cipher Suites	*ECDSA; Secp256k1*
Description	*Elliptic Curve Digital Signature Algorithm or ECDSA is a cryptographic algorithm used by Bitcoin to ensure that funds can only be spent by their rightful owners.* *More information may be found here:* *https://en.bitcoin.it/wiki/Elliptic_Curve_Digital_Signature_Algorithm*

Section 7 Operation & Maintenance

Platform system management – Node	
Log	*Yes*
Monitoring	*bitcoind*
Description	*bitcoind is the daemon client that manages all interactions with the Bitcoin network. It also acts as the interface between wallet software and the Bitcoin network. A number of log levels may be activated with the software. It is a crucial element of node management.*

Platform system management – Chain Network	
Permission Control	*N/A*
Auditing	*Auditing mechanisms are self-contained within each wallet and pertains to each wallet address managed by the wallet software.* *Anyone can audit the history and current balance associated to any address by having a copy of the blockchain or using a public "block explorer" that facilitates visualizing this information.*
Supervisory Support	*N/A*

Description	*N/A*

Section 8 External Resource Management

Platform External Resource Management	
Interoperation solution	*N/A*
Description	*N/A*

Section 9 Extensions

Platform Extensions - optional	
[the following list can be duplicated for multiple extensions]	
Name	*Lightning Network*
Extension type[2]	*Second Layer Interaction Solution*
Extension mode[3]	*Hash Time Locked Contracts (HTLCs)*
Solution	
Serve domain	*Financial Transactions*
Description	*Lightning Network is a proposed implementation of Hashed Timelock Contracts (HTLCs) with bi-directional payment channels which allows payments to be securely routed across multiple peer-to-peer payment channels. This allows the formation of a network where any peer on the network can pay any other peer even if they don't directly have a channel open between each other. As of March 2019, there were more than 37,000 channels carrying more than 764 bitcoins.*

---

[2] Standing from DLT system instance perspective, any extension inside the instance is marked as "internal", while any extension outside the instance is marked as "external"

[3] All extension instances are equal (with similar capability and functional features), targeting for the scalability of DLT instance, marked as "horizontal"; extensions with different functional features, targeting to enforce the capability of DLT instance, marked as vertical. Extension type and mode pair(s) is/are used to describe the extension as to the whole DLT system. E.g., sharding (internal – horizontal), lightening – BTC (external – vertical), Corda Contract (internal – vertical).

Attachment IV – Architecture Mapping of Corda

Section 1 Summary

Platform summary	
Platform ID	*Corda …*
Status/Revision	V4.0
Type	*Private, Consortium…*
Domain	*Mainly Financial, do to R3 consortium focus but can be adapted to many other segments and needs as Cordapps and network structurers can be easily adapted.*
Description	*Corda is an Open Source DLT that allow business to transact in a strict privacy P2P way by using Cordapps (smart contracts), reducing costs for the network owners.*

Section 2 Governance & Compliance Functions

Platform governance	
Governance Type	*Permissioned;*
Chain Network Admin	*Entity (Consortium/Private)*
Pledge (cost of malicious action)	***Business agreement, third parties liabilities (open source version)***
Description	*Governing body who led the Consortium. Need 3-rd party to arbitrate the dispute based upon the agreement.*

Platform trust endorsement policy	
Type	*Law/Agreement;*
Tool	*N/A*
Policy	

Economic Model (optional)	
Price Model to Deploy Contracts and do Transactions	*N/A*
Who pays the costs of the network	*N/A*
Monetary Policy of Tokens	*N/A*
Rights of Tokens	*N/A*

Section 3 Application

Platform Smart Contract mechanism	
Language	*Java, Kotlin*
Turing Complete?	*Yes*
Compiler	*Java, Kotlin*
Runtime VM	*JVM*
DevTools	*IntelliJ IDEA, Eclipse IDE*
Extra Tool(s)	*Node Explorer, Load Testing, Corda Network Builder*
Lifecycle	*Until now Cordapps must be installed manually inside Nodes specific folder and then restarted.*
Description	*Corda, uses a "Contract" code to validate State transactions.* *The contract code is a "pure" function executed in a deterministic environment, on a need-to-know basis which verifies transactions.*

Section 4 Protocol

Platform AAA Management	
Account type	*Address*

Distributed ID	*Commonly used Public/Private RSA 3072 bit Keypair with X.509 v3 Standard Certificates on a TLS v1.2 Standard Protocol*
AAA support	*Fabric CA;* *Membership Service Providers,*
Description	*Corda's network permissioning is composed by an certificate hierarchy as follows: Root Network CA, The doorman CA, Node CA, legal identity CA.* *Root Network CA: Used to issue the Doorman and control the Network.* *The Doorman CA (intermediary): Used to sign Node Keys on a day-to-day to not compromise Root's CA Private Key.* *Node CA: Each node issues its own certificate that is used to sing its identity keys and TLS certificates*

Platform Consensus Mechanism	
Algorithm	*Contract Code*
Consensus mode	*Event*
Management solution	*External*
Description	*Corda offers 2 types of Consensus:* *a) Where each required signer node, must validate the proposal before they sign the transaction.* *b) The transaction is only checked and validated by a 3rd party node "Notary Service".*

Platform Ledger Management	
Model	*UTXO*
Extra	*State*
Description	*Corda uses UTXO (Unspent transaction output) model where every state on the ledger is immutable.*

Section 5 Resources

Node Management	
Node Role	*The roles of the Corda nodes are exposed to the entire network through the Network Map and also Corda's certificates have a custom X.509 v3 extension that specifies the role the certificate relates to. This is how roles are defined inside the Network, as Doorman, Network Map, Node CA, etc…The extension contains a single ASN.1 integer identifying the identity type the certificate is for:* 1. *Doorman* 2. *Network map* 3. *Service identity (currently only used as the shared identity in distributed notaries)* 4. *Node certificate authority (from which the TLS and well-known identity certificates are issued)* 5. *Transport layer security* 6. *Well-known legal identity* 7. *Confidential legal identity*
Joining	*To Join to a Corda network a Regular Node must make the request to a "Doorman" server (Intermediary) so it can validate and authenticate the request. In addition to the Network Map, all the nodes must also use the same set of network parameters. These are a set of constants which guarantee interoperability between the nodes. The HTTP network map distributes the network parameters which are downloaded automatically by the nodes. Every new node must be listed inside the network map with their roles and profiles.*
Leaving	*If a Corda Node gets offline for any reason, he will still be listed inside the network map as a member of that network, so every transaction that is sent to him, it will be "on hold" until his return. It is up to Network Admin, to clear Network Map cache (updating the list), and kicking the specific "dark node".*
Role changing	*To change a Node role in corda few steps must be made.* *1) Change Node configuration file attending his new Role inside network structure.* *2) Issue new Certificate for the node accordingly to his new role inside network.* *3) Update Network Map accordingly to his current new functions and values.*
Description:	

Platform Data Storage Mechanism	
Mass storage mitigation[1]	*N/A*
Decentralized Data Storage Support	*N/A*
Data Privacy Solution	*Enables confidentiality trough Node P2P transaction (need-to-know basis).*
Tamper Proof (tamper cost)	
Description	

Platform Network Management	
Node Scalability	*Hundreds*
Network Structure	*Flexible*
Network Discovery Protocol	*HTTP Network Map*
Byzantine Node Accepted?	*Not Natively*
P2P?	*Yes*
Data Exchange Protocol	*AMQP/1.0 TLS*
Description	*A Notary demo, based on BFT-Smart Protocol was released.*

Section 6 Utils

Platform Messaging Mechanism	
Protocol Type	*RPC external and AMQP/1.0 TLS for internal Network Messaging*
Description:	*Nodes owners uses RPC Client to communicate with the Node.*

[1] On chain storage cost much, solution/mechanism to resolve the problem of large cost of mass storage from node perspective. E.g., data maintenance, data storage and data cleaning.

Platform Crypto Libraries	
Secure Network Connection Type	*TLS*
Cipher Suites	*ECDSA Nist P-256 curve (Secp256r1) or RSA with 3072bit keys*
Description:	

Section 7 Operation & Maintenance

Platform system management – Node	
Log	*yes*
Monitoring	*Node Explorer*
Description	*Corda Network Builder, Load Testing tool*

Platform system management – Chain Network	
Permission Control	*The Root CA*
Auditing	*In Schedule*
Supervisory Support	*N/A*
Description	

Section 8 External Resource Management

Platform External Resource Management	
L2 solution:	*N/A*
Non-DLT system interoperation solution:	*Support for Oracle and SQL Server Database*
Description:	

Section 9 Extensions

Platform Extensions - optional	
[the following list can be duplicated for multiple extensions]	
Name	*Business Network*
Extension type[2]	*Internal*
Extension mode[3]	*Capability (vertical) and* Scalability (*horizontal)*
Solution	*Corda Multiple Cordapps/Contract*
Serve domain	*Scalability: Cordapps/Contracts*
Description	*Corda can have Multiple Cordapps/Contracts inside same node, providing as many individual P2P Business Networks Extensions needed. This way each network can enforce its own access control policies and process but at same time, they can have their own determination about which business networks they choose to participate.*

Platform Extensions - optional	
[the following list can be duplicated for multiple extensions]	
Name	*Corda Settler*
Extension type	*Internal*
Extension mode	*Capability (vertical)*
Solution	*Corda Settler is a DLT Cordapp that allows settlements payments transactions between crypto and traditional assets.*
Serve domain	*Scalability: Cordapps/Contracts*
Description	*Corda Settler is already working with Ripple XRP and also implemented SWIFT gpi link integration, that allows DLT users to settle payments obligation to DLT's, blockchains and traditional non-DLT rails.*

[2] Standing from DLT system instance perspective, any extension inside the instance is marked as "internal", while any extension outside the instance is marked as "external"

[3] All extension instances are equal (with similar capability and functional features), targeting for the scalability of DLT instance, marked as "horizontal"; extensions with different functional features, targeting to enforce the capability of DLT instance, marked as vertical. Extension type and mode pair(s) is/are used to describe the extension as to the whole DLT system. E.g., sharding (internal – horizontal), lightening – BTC (external – vertical), Corda Contract (internal – vertical).

Attachment V – Architecture Mapping of EOS

Section 1 Summary

Platform summary	
Platform ID:	*EOS*
Status/Revision:	1.8.0
Type:	*Public*
Domain:	Infrastructure

Section 2 Governance & Compliance Functions

Platform governance	
Governance Type:	*Permissionless*
Chain Network Admin:	*BP (Block producer)*
Pledge (cost of malicious action):	***Stake***
Description:	*Delegate stake to BPs, BP as block producer*

Platform trust endorsement policy	
Type:	*Tokenomics*
Tool:	EOS
Policy:	*EOS stake owner can delegate EOS and get EOS RAM, EOS RAM as gas to use blockchain resources*

Section 3 Application

Platform smart contract mechanism	
Language	*WASM compiler supported languages[1]: C/C++*
Turing Complete?	*yes*
Compiler:	*WASM compiler with page protection[2]*

[1] https://github.com/appcypher/awesome-wasm-langs

[2] see footnote 1.

Runtime VM:	*WASM VM*
DevTools	*Contract development toolkit*[3]
Extra Tool(s):	*Explorer*[4]
Lifecycle	*Live per app call / api call*

Section 4 Protocol

Platform AAA Management	
Account type:	*address*
Distributed ID:	*N/A*
AAA support:	*Membership Service Providers*
Description:	*Governance section in EOS whitepaper* [5]

Platform consensus mechanism	
Algorithm:	*DPoS + asynchronized BFTN/A*
Consensus mode:	*Event*
Management solution:	*Internal*
Description:	*Delegate on stake, single block producer keep generating blocks, with verifiers' signatures*

Platform ledger management	
Model:	*balance*
Extra:	*N/A*
Description:	-

Section 5 Resources

[3] https://github.com/EOSIO/eosio.cdt

[4] https://bloks.io/

[5] https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md#governance

Node Management	
Node Role	*Sync node; BP*
Joining	*Any entity can run a sync node, and call for stake delegation from stakeholders; nodes with delegated stake reaches rank of top 21 will be block producer (BP)*
Leaving	*So far, no BP is leaving the network. Technically, BP(s) give away their stake and stop the node instances will quit the network*
Role changing	*When stake rank lower than 21, BP will downgrade to common sync node*
Description	*Each token holder determines the accounting right of the blockchain by voting, similar to the election of the board of directors. All nodes whose votes exceed the agreed votes become system trustees, forming a "board of directors" and alternately signing blocks. If a director missed the chance to sign a block, the nodes would vote for the others. Those boards that miss the chance to sign are disqualified and others can join the board.*

Platform data protection - core	
Mass storage mitigation[6]	*On-chain storage with consume of EOS RAM*
Decentralized Data Storage Support	*N/A*
Data Privacy Solution	*N/A*
Tamper Proof (tamper cost):	*stop service, average PoS * 1/3 network scale (nodes)* *tamper*
Description:	*Can be rollback on vote result by stake* *Off-chain punishment on BP node(s)*

Platform Network Management	
Node Scalability:	*1 continuous block generation +21 asynchronized BFT*
Byzantine Node Accepted? :	*Partially*
Network Structure	*Flexible*

[6] On chain storage cost much, solution/mechanism to resolve the problem of large cost of mass storage from node perspective. E.g., data maintenance, data storage and data cleaning.

P2P? :	*Yes*
Network Discovery Protocol	*Kademlia-like*
Data Exchange Protocol	*Gossip*
Description	*DPoS, with aBFT, the cost of BFT is very high, comparing with the performance of block generation, BPs are able to rollback before the block reaches its finality.* *The BP node is not changing rapidly, the network is stable, and connect directly.* *EOS community provides some P2P solutions for sync node.*

Section 6 Utils

Platform Messaging Mechanism	
Protocol Type	*RPC*
Description:	-

Platform Crypto Libraries	
Secure Network Connection Type	*SSL; TLS; …*
Cipher Suites	*ECDSA; EECDH; AESGCM; aRSA; EECDH; SHA; AES*
Description:	-

Section 7 Operation & Maintenance

Platform system management – Node	
Log:	*Yes*
Monitoring:	-
Recommend Operation:	-
Description:	*[Operation and Maintenance] there are many system management tool from EOS community, this doc is based on EOSIO github solution. For the rest, not list the detail*

Platform system management – Chain Network	
Permission Control:	*Yes*
Auditing:	*N/A*
Supervisory Support:	*N/A*
Description:	*[Operation and Maintenance] -*

Section 8 External Resource Management

Platform External Resource Management	
L2 solution:	*N/A*
Non-DLT system interoperation solution:	*N/A*
Description:	-

Section 9 Extensions

***** *There are many extensions built from EOS community, this doc is based on EOSIO github solution. For the rest, not list the detail.*

Attachment VI – Architecture Mapping of Ethereum

Section 1 Summary

Platform summary	
Platform ID	*Ethereum/ETH…*
Status/Revision	*Mainnet – Geth 1.8.27*
Type	*Public*
Domain	*Many sectors*
Description	*Ethereum is a global, open-source platform for decentralized applications.* *There is a textual Ethereum assessment at:* https://archive-ouverte.unige.ch/unige:112558

Section 2 Governance & Compliance Functions

Platform governance	
Governance Type	*Permissionless;*
Chain Network Admin	*NA/Community (public)* *EIP - Ethereum Improvement Proposal*
Pledge (cost of malicious action)	*Computer Power – measured by hash rates*
Tamper Proof (tamper cost)	>50%
Description	*https://eips.ethereum.org/* *The EIP (Ethereum Improvement Proposals) is an open governance model where everyone is free to propose and discuss changes to the system. There are several different stages that an EIP can be in. Draft EIPs are works in progress, are open for consideration and discussed on Github. Accepted EIPs can be expected to be included in the next hard fork. Final EIPs are proposals that have already been adopted and deferred EIPs are not being considered for immediate adoption, but may be considered again in the future.* *ERC (Ethereum request for change) is a type of EIP to application-level standards and conventions.*

Platform trust endorsement policy

Type	*Tokenomics[1];*
Tool	*ETH (according with coinmarketcap)*
Policy	~~*Game theory – tokens*~~*PoW with bounded rationality*

Economic Model (optional)	
Price Model to Deploy Contracts and do Transactions	*Deploy a new contract is a kind of transaction,* *Charged by transactions only*
Who pays the costs of the network	*Users*
Monetary Policy of Tokens	*Unlimited supply* *New ethers are constantly been created together with new blocks. During each block creation, Ethereum implementation of PoW give rewards to the winner miner and to miners of stale descendants of ancestors blocks which attend some protocol rules (GHOST protocol).* https://github.com/ethereum/EIPs/blob/master/EIPS/eip-1234.md proposed minimize block rewards. *In order to minimize the incentive of node centralization into pools, Ethash is ASIC resistant and do not generate super-linear profits in mining rewards.*
Rights of Tokens	*Not applicable*

Section 3 Application

Platform Smart Contract mechanism	
Language	*Solidity; Vyper (beta)*
Turing Complete?	*Yes – Solidity*
Compiler	*Solcjs - Solidity*
Runtime VM	*EVM;*

[1] Alternative term: economic incentives. Depends on the terms in the output of D1.1, if the term of tokenomics has clear definition, use tokenomics, otherwise, economic incentives

DevTools	*Development: Visual Studio Code; Sublime; Remix;* *Build framework: Truffle, Embark, Remix* *Test framework: Truffle, Embark, Remix*
Extra Tool(s)	*Explorer (Block data view): EtherScan* *Speed-test:* https://www.blocktivity.info/ *Gas price metrics: https://ethgasstation.info/*
Lifecycle	*The developer have to code if the contract can stop or be killed. It is not possible to update the smart contract, but there are recommendations to that.*
Description	*There are many tools in this link:* https://ethereum.consensys.net/?utm_medium=social&utm_source=lin *It includes browser extensions, testnets, front end libs, smart contract libraries and security tools.*

Section 4 Protocol

Platform AAA Management	
Account type	*Address;*
Distributed ID	*There are two types of accounts which share the same address space: externally owned accounts and contract accounts. Externally owned accounts are controlled by public-private key pairs and have no code. Contract accounts are controlled by the code stored together with the account – the smart contract code.* *User should generate an externally owned account using a local software/hardware in order to keep the private key private;* *Contract accounts are created during deploy.*
AAA support	*N/A*
Description	*The rational is that there are so many possible addresses that the probability of collision is negligible.*

Platform Consensus Mechanism	
Algorithm	*PoW;*
Consensus mode	*Event;*

Management solution	*Internal;*
Description	

Platform Ledger Management	
Model	*balance;*
Extra	*MPT support - modified Merkle Patricia tree (trie)*
Description	*Each account has a storage, a persistent memory area. A contract can neither read nor write to any storage apart from its own.* *From a block header there are 3 roots from 3 MPT: stateRoot, transactionsRoot and receiptsRoot.*

Section 5 Resources

Node Management	
Node Role	*Full Nodes and Full archiving nodes.*
Joining	*The node has to sync with the network and start to participate without permission*
Leaving	*Nodes can stop working at any time.*
Role changing	*A node can independently and at any time to change role.*
Description	-

Platform Data Storage Mechanism	
Mass storage mitigation[2]	*Concept of Gas* *Some operations may have negative gas cost, for example kill a contract.*
Decentralized Data Storage Support	*IPFS, SIA*
Data Privacy Solution	*N/A*
Description	*The fundamental unit of computation is called "gas"; The fee system is to require a person to pay proportionately for every resource that they consume, including computation, bandwidth and storage;*

[2] On chain storage cost much, solution/mechanism to resolve the problem of large cost of mass storage from node perspective. E.g., data maintenance, data storage and data cleaning.

Platform Network Management	
Node Scalability	*Thousands*
Network Structure	*Distributed*
Network Discovery Protocol	*Kademlia-like;*
Byzantine Node Accepted?	*Yes*
P2P?	*Yes*
Data Exchange Protocol	*RLPx*
Description	https://github.com/ethereum/wiki/wiki/Kademlia-Peer-Selection *RLPx transport protocol, a TCP-based transport protocol used for communication among Ethereum nodes. The protocol carries encrypted messages belonging to one or more 'capabilities' which are negotiated during connection establishment.* https://github.com/ethereum/devp2p/blob/master/rlpx.md

Section 6 Utils

Platform Messaging Mechanism	
Protocol Type	*RPC*
Description	*JSON-RPC is a stateless, lightweight remote procedure call (RPC) protocol. Primarily this specification defines several data structures and the rules around their processing. It is transport agnostic in that the concepts can be used within the same process, over sockets, over HTTP, or in many various message passing environments. It uses JSON (RFC 4627) as data format.* https://github.com/ethereum/wiki/wiki/JSON-RPC

Platform Crypto Libraries	
Secure Network Connection Type	*Communication via public Internet (TCP + UDP).*
Cipher Suites	*ECDSA (Elliptic Curve Digital Signature Algorithm) for it's public-key cryptography and KECCAK-256 for hashing*

	There is a discussion about how these algorithms were implemented at: https://ethereum.stackexchange.com/questions/71657/cipher-suites-open-source
Description	*Geth (The official Ethereum client node software) uses UDP connection to exchange information about the P2P network. After establishing peer connections, Geth nodes exchange blockchain information via encrypted and authenticated TCP connections.*

Section 7 Operation & Maintenance

Platform system management – Node	
Log	*Yes*
Monitoring	https://www.ethernodes.org/
Description	*Ethernodes allows anyone to see the number of nodes and where they are located* *There is no special nodes (masternodes, special block producers etc) in the network.*

Platform system management – Chain Network	
Permission Control	*By code only* *N/A*
Auditing	*Information public available in blockchain explorers like https://etherscan.io.*
Supervisory Support	*N/A*
Description	*Etherscan.io shows information about blocks, transactions, tokens, smart contracts, addresses and the history of its transactions. Etherscan.io is independently operated and developed independent of the Ethereum Foundation*

Section 8 External Resource Management

Platform External Resource Management	
Interoperation solution	
Description	

Section 9 Extensions

***** *There are many extensions built from Ethereum community, with PoC. This doc is based on Geth 1.8.27 only. For the rest, not list the detail.*

Platform Extensions – optional	
[the following list can be duplicated for multiple extensions]	
Name	
Extension type[3]	
Extension mode[4]	
Solution	
Serve domain	
Description	

[3] Standing from DLT system instance perspective, any extension inside the instance is marked as "internal", while any extension outside the instance is marked as "external"

[4] All extension instances are equal (with similar capability and functional features), targeting for the scalability of DLT instance, marked as "horizontal"; extensions with different functional features, targeting to enforce the capability of DLT instance, marked as vertical. Extension type and mode pair(s) is/are used to describe the extension as to the whole DLT system. E.g., sharding (internal – horizontal), lightening – BTC (external – vertical), Corda Contract (internal – vertical).

Attachment VIII – Architecture Mapping of Hyperchain

Section 1 Summary

Platform summary	
Platform ID	*Hyperchain*
Status/Revision	*V1.8.0*
Type	*Permissioned, Consortium*
Domain	*Provide solutions for financial, medical, energy, trade and other fields.*
Description	*Hyperchain provides technical support to companies, government agencies and industry alliances.*

Section 2 Governance & Compliance Functions

Platform governance	
Governance Type	*Permissioned;*
Chain Network Admin	*ACO (autonomous consortium organization)*
Pledge (cost of malicious action)	*The certificate of malicious node will be revoked.*
Description	*Before this, Hyperchain has strict access mechanisms by member management model. If the Byzantine node is spotted, the members of the group will vote on a proposal to remove it from the organization.*

Platform trust endorsement policy	
Type	*Law/Agreement;*
Tool	*Legal Contract*
Policy	*A peer with Ecert and Rcert is verified as validate peer, all validate peers can participate in the endorsement.*

Section 3 Application

Platform Smart Contract mechanism

Language	*Solidity, Java*
Turing Complete?	*Yes*
Compiler	*Solidity; Java*
Runtime VM	*EVM, JVM, HVM*
DevTools	*GoSDK*、*JavaSDK*
Extra Tool(s)	*Hyperchain Radar (Contract data view);* *MQ (Message push);* *Hypervision (visualized monitoring platform)*
Lifecycle	*Hyperchain's VM supports whole smart contract lifecycle management, including contract deployment, upgrade, freeze, and more.*
Description	*Support JVM*、*EVM and HVM with multiple programming languages compiler.* *SDK provides many interfaces to facilitate application development as a tool for application to interact with hyperchain.*

Section 4 Protocol

Platform AAA Management	
Account type	*Identity; address; …*
Distributed ID	*PKI structure, a digital identity encapsulated in an X.509 digital certificate.*
AAA support	*Hyperchain CA;* *CFCA.*
Description	*Authority control is at the Namespace level, which means that each Namespace will have a corresponding CaManager for CA certificate management and authority control at the Namespace level. The CA system is mainly used for node authority control and transaction authority control.*

Platform Consensus Mechanism	
Algorithm	*RBFT(Robust Byzantine Fault Tolerant) ;*

Consensus mode	*Event;*
Management solution	*Internal*
Description	*RBFT adds active recovery and dynamic node addition and deletion mechanism by optimizing PBFT execution process. Under the premise of ensuring strong consistency of node data, RBFT improves the overall transaction throughput capacity and system stability of the system.*

Platform Ledger Management	
Model	*Account*
Extra	*HMT(HyperMerkle Tree)*
Description	*HyperMerkle trees combine the advantages of both Merkle trees and hash tables, greatly increasing the speed of ledger hash calculations.*

Section 5 Resources

Node Management	
Node Role	*Validate peer (VP);* *Candidate validate peer (CVP);* *Non-validate peer (NVP).*
Joining	*Node will be joined in chain when the entity is allowed to join the consortium, the CAs for node will be offered, then node will be started with CAs.*
Leaving	*Node will be deleted when it become a byzantine node.*
Role changing	*When VP is failover, the CVP node will become validate node.*
Description	*Hyperchain consists of validate peers (VP), candidate validate peers (CVP) and non-validate peers (NVP):* *Validate node refers to the node participating in consensus validate in the blockchain network.* *Candidate validate node refers to node which is the candidate of validate node, when validate node is failover, this node will be become validate node to join consensus.* *Non-validate node refers to the node in the blockchain network that does not participate in consensus validate and only participates in accounting and needs to connect the validate node.*

Platform Data Storage Mechanism	
Mass storage mitigation[1]	*Data Archiving*
Decentralized Data Storage Support	*TiKV*
Data Privacy Solution	*End-to-End TLS encrypted data;* *namespace;* *private transaction.*
Tamper Proof (tamper cost)	*More than 1/3 nodes tampered.*
Description	*Namespace can protect privacy of business layer.* *Private transaction can protect privacy on transaction layer.*

Platform Network Management	
Node Scalability	*Hundreds*
Network Structure	*Distributed; Flexible*
Network Discovery Protocol	*gRPC*
Byzantine Node Accepted?	*Yes*
P2P?	*Yes*
Data Exchange Protocol	*gRPC*
Description	*gRPC uses protocol buffers to connect data centers with pluggable load balancing.*

Section 6 Utils

Platform Messaging Mechanism	
Protocol Type	*gRPC;*
Description	*Further description if any*

[1] On chain storage cost much, solution/mechanism to resolve the problem of large cost of mass storage from node perspective. E.g., data maintenance, data storage and data cleaning.

Platform Crypto Libraries	
Secure Network Connection Type	*TLS; …*
Cipher Suites	*SHA; SM3; ECDSA; SM2; ECDH; AES; SM4; SM9*
Description	*Hyperchain uses SHA and SM3 for hash, supports ECDSA and SM2 for signing. ECDH is used for key agreement, and AES or SM4 for message transmission.*

Section 7 Operation & Maintenance

Platform system management – Node	
Log	*Yes*
Monitoring	*Provides* a *visualized monitoring platform named Hypervision.*
Description	*Hypervision is designed for real-time monitoring and alarms on blockchains, as well as management of smart contracts.* *IPC command can be used to manage network connections, make log-level modification, query license information and so on.*

Platform system management – Chain Network	
Permission Control	*Yes*
Auditing	*Security testing of transactions*
Supervisory Support	*The supervisor can join the blockchain network as a node.*
Description	*Do security testing on extra of the transaction before it's written into the block, failed transactions will become invalid.*

Section 8 External Resource Management

Platform External Resource Management	
Interoperation solution	*Oracle pushes third-party data to Hyperchain, and the smart contract can obtain data information from a specific blockchain address.*
Description	*Oracle provides externally trusted data sources which are authoritative, accurate, non-tamper, stable, and acceptable for auditing, such as databases, trusted timestamps, etc.*

Section 9 Extensions

Platform Extensions - optional	
[the following list can be duplicated for multiple extensions]	
Name	*Smart Contract Support*
Extension type	*Internal*
Extension mode	*Vertical*
Solution	*Hyperchain has accessed multiple virtual machines: EVM, JVM, HVM.*
Serve domain	*Smart Contract Support*
Description	*The smart contract engines support Solidity and Java, they are user-friendly. Contracts are easy to compile and deploy on Hyperchain, all contracts are compatible and portable.*

Platform Extensions - optional	
[the following list can be duplicated for multiple extensions]	
Name	*Inter Blockchain*
Extension type	*External*
Extension mode	*Horizontal*
Solution	*Heterogeneous blockchains with Application chains, side chains and etc.*
Serve domain	*Cross Chain Applications*
Description	*Inter Blockchain supports for inter-chain transactions between homogeneous and heterogeneous blockchain platforms to form blockchain internet.*

Attachment VII – Architecture Mapping of Hyperledger Fabric

Section 1 Summary

Platform summary	
Platform ID	*Hyperledger Fabric*
Status/Revision	*Proposed; V1.4* (*long term support release),*
Type	*Permissioned, Consortium*
Domain	*A wide range of industry use cases ranging from government, to finance, to supply-chain logistics, to healthcare and so much more.*
Description	*Hyperledger Fabric has been designed for enterprise use from the outset.*

Section 2 Governance & Compliance Functions

Platform governance	
Governance Type	*Permissioned;*
Chain Network Admin	*Entity (Consortium/Private)*
Pledge (cost of malicious action)	*The guilty party can be easily identified and the incident handled in accordance with the terms of the governance model. Stop service.*
Description	*Fabric operates a blockchain amongst a set of known, identified and often vetted participants operating under a governance model. All actions, whether submitting application transactions, modifying the configuration of the network or deploying a smart contract are recorded on the blockchain. So the guilty party can be easily identified and the incident handled in accordance with the terms of the governance model.*

Platform trust endorsement policy	
Type	*Law/Agreement;*
Tool	*Contract ID;*
Policy	*A fabric network can be operated under a governance model that is built off of what trust does exist between participants, such as a legal agreement or framework for handling disputes.*

	An endorsement policy which specifies the set of peers on a channel that must execute chaincode and endorse the execution results.

Economic Model (optional)	
Price Model to Deploy Contracts and do Transactions	*NA*
Who pays the costs of the network	*NA*
Monetary Policy of Tokens	*Fabric v2.0 will release the FabToken feature. FabToken is an Unspent Transaction Output (UTXO) based token management system that allows users to issue, transfer, and redeem tokens on channels.*
Rights of Tokens	*FabToken uses the membership services of Fabric to authenticate the identity of token owners and manage their public and private keys.*

Section 3 Application

Platform Smart Contract mechanism	
Language	*Go, Node.js, Java; ...*
Turing Complete?	*Yes*
Compiler	*Go; Java; …*
Runtime VM	*a secured Docker container; …*
DevTools	*Dev framework(the chaincode/(smart contract) interface, the chaincode "shim" APIs is the ChaincodeStubInterface; fabric-chaintool;…*
Extra Tool(s)	*Hyperledger Explorer (Block data view); Hyperledger Caliper (performance benchmarking); Hyperledger Burrow(Solidty smart contract migration); Hyperledger Composer(contract orchestration)*
Lifecycle	*The Hyperledger Fabric API supports package, install, instantiate and upgrade chaincode on the endorsing peer nodes.*
Description	*the chaincode interface whose methods are called in response to received transactions, the ChaincodeStubInterface is used to access and modify the ledger; fabric-chaintool is a utility to assist in various phases of chaincode development, such as compilation, test, packaging, and deployment;*

Section 4 Protocol

Platform AAA Management	
Account type	*Identity; address;*
Distributed ID	*using an established PKI structure, a digital identity encapsulated in an X.509 digital certificate*
AAA support	*Fabric CA;* *Membership Service Providers,*
Description	*Membership Services authenticates, authorizes, and manages identities on a permissioned blockchain network. A membership identity service that manages user IDs and authenticates all participants on the network. Access control lists can be used to provide additional layers of permission through authorization of specific network operations.* *Fabric provides Identity Mixer, which has a trust model and security guarantees similar to what is ensured by standard X.509 certificates but with underlying cryptographic algorithms that efficiently provide advanced privacy features such as "unlinkability" and minimal attribute disclosure.*

Platform Consensus Mechanism	
Algorithm	*CFT ;*
Consensus mode	*Event;*
Management solution	*external*
Description	*In the currently available releases, Fabric offers a CFT ordering service implemented with Kafka and Zookeeper. It also has a Raft consensus implementation. From version 1.4.2, users can migrate from Kafka to Raft. There're third parties providing other consensus for Fabric like LaSIGE implemented BFT-Smart consensus.*

Platform Ledger Management	
Model	*balance;*
Extra	*Fabric Coin (UTXO cryptocurrencies) described in a peer reviewed paper published by a team from IBM Research;*
Description	*N/A*

Section 5 Resources

Node Management	
Node Role	*There are three types of nodes in Fabric:* *1. Client or submitting-client: a client that submits an actual transaction-invocation to the endorsers, and broadcasts transaction-proposals to the ordering service.* *2. Peer: a node that commits transactions and maintains the state and a copy of the ledger. Besides, peers can have a special endorser role. The special function of an endorsing peer occurs with respect to a particular chaincode and consists in endorsing a transaction before it is committed.* *3. Ordering-service-node or orderer: a node running the communication service that implements a delivery guarantee, such as atomic or total order broadcast.*
Joining	*1.Channel administrators send out channel configuration tx to Orderer, to add an organization to the channel;* *2. Organization administrators sends a join channel proposal to one or more endorsing peers, to let a peer of the organization join the channel;*
Leaving	*Peers have no activity on the channel for a timely basis;*
Role changing	*Organization administrators send out channel configuration tx to Orderer, to update an anchor peer;*
Description	*Network MSP defines who are the members in the network; Orderer MSP list the actors or nodes it trusts;* *A peer need to be registered and enrolled to obtain the enrollment certificate signed by the Organization CA;* *Anchor peer is used by gossip to make sure peers in different organizations know about each other.*

Platform Data Storage Mechanism	
Mass storage mitigation[1]	*NA…*
Decentralized Data Storage Support	*KV style databases*
Data Privacy Solution	*Fabric enables confidentiality through its channel architecture, and has added support for private data encryption. A Hyperledger Fabric channel is a private "subnet" of communication between two or more specific*

[1] On chain storage cost much, solution/mechanism to resolve the problem of large cost of mass storage from node perspective. E.g., data maintenance, data storage and data cleaning.

	network members, for the purpose of conducting private and confidential transactions.
Tamper Proof (tamper cost)	*In the currently available 1.4.0 releases, Fabric only offers a CFT ordering service. And depends on Endorsement policies.*
Description	*zero knowledge proofs (ZKP) available in the future*

Platform Network Management	
Node Scalability	*Hundreds*
Network Structure	*Distributed; Flexible;*
Network Discovery Protocol	*Gossip;*
Byzantine Node Accepted?	*No for now*
P2P?	*Yes(except Orderers)*
Data Exchange Protocol	*Gossip;*
Description	*The gossip-based data dissemination protocol manages peer discovery and channel membership, disseminates ledger data across all peers on a channel, brings newly connected peers up to speed by allowing peer-to-peer state transfer update of ledger data;*

Section 6 Utils

Platform Messaging Mechanism	
Protocol Type	*gRPC;*
Description	*Further description if any*

Platform Crypto Libraries	
Secure Network Connection Type	*TLS;*
Cipher Suites	*Hyperledger Ursa ;*
Description	*The signing key used for signing by the node (currently only ECDSA keys are supported). In Hyperledger Fabric there is no support for certificates including RSA keys for now, a new cipher library Ursa has just been added to Hyperledger projects and may provide various algorithms later.*

Section 7 Operation & Maintenance

Platform system management – Node	
Log	*Yes*
Monitoring	*provides operators with three services to monitor, logging, Health Checks, metrics from peer and orderer nodes*
Description	*Some components of the Fabric peer and orderer expose metrics that can help provide insight into the behavior of the system. Operators and administrators can use this information to better understand how the system is performing over time.* *Hyperledger Explorer project can also be used to monitor on-chain data.*

Platform system management – Chain Network	
Permission Control	*Local MSPs; Channel MSP; Network MSP; Peer MSP; Orderer MSP;*
Auditing	*N/A*
Supervisory Support	*N/A*
Description	*An MSP goes beyond simply listing who is a network participant or member of a channel. An MSP can identify specific roles an actor might play either within the scope of the organization the MSP represents (e.g., admins, or as members of a sub-organization group), and sets the basis for defining access privileges in the context of a network and channel (e.g., channel admins, readers, writers)*

Section 8 External Resource Management

Platform External Resource Management	
Interoperation solution	*NA;*
Description	*N/A*

Section 9 Extensions

Platform Extensions - optional
[the following list can be duplicated for multiple extensions]

Name	*Smart Contract Support:*
Extension type[2]	*Internal;*
Extension mode[3]	*capability (vertical)*
Solution	*Hyperledger Burrow enables one to use the Hyperledger Fabric permissioned blockchain platform to interact with Ethereum smart contracts written in an EVM compatible language such as Solidity or Vyper.*
Serve domain	*Smart Contract Support*
Description	*the Fabric EVM Chaincode(Enables Ethereum smart contracts written in an EVM compatible language such as Solidity or Vyper)*

[2] Standing from DLT system instance perspective, any extension inside the instance is marked as "internal", while any extension outside the instance is marked as "external"

[3] All extension instances are equal (with similar capability and functional features), targeting for the scalability of DLT instance, marked as "horizontal"; extensions with different functional features, targeting to enforce the capability of DLT instance, marked as vertical. Extension type and mode pair(s) is/are used to describe the extension as to the whole DLT system. E.g., sharding (internal – horizontal), lightening – BTC (external – vertical), Corda Contract (internal – vertical).

Attachment XIV – Architecture Mapping of Hyperledger Sawtooth

Section 1 Summary

Platform summary	
Platform ID	*Hyperledger Sawtooth*
Status/Revision	*V1.1.4, …*
Type	*Permissioned, Consortium*
Domain	*Financial, to supply chain, to assets marketplace and so much more.…*
Description	*Hyperledger Sawtooth is an enterprise blockchain platform for building distributed ledger applications and networks.*

Section 2 Governance & Compliance Functions

Platform governance	
Governance Type	*Permissioned (private); …*
Chain Network Admin	*Entity (Consortium/Private)…*
Pledge (cost of malicious action)	*Blacklisting; stop service*
Description	*The blockchain stores the settings that specify the permissions, such as roles and identities, so that all participants in the network can access these informations.*

Platform trust endorsement policy	
Type	*Trusted execution environment;*
Tool	*Contract ID*
Policy	*Offers a solution to the Byzantine Generals Problem that utilizes a TEE (trusted execution environment).Cheating is prevented through the use of a TEE, identity verification and blacklisting based on asymmetric key cryptography. Using TEE provided by intel chips implies endorsement from Intel corporation.*

Economic Model (optional)

Price Model to Deploy Contracts and do Transactions	*NA*
Who pays the costs of the network	*NA*
Monetary Policy of Tokens	*NA*
Rights of Tokens	*NA*

Section 3 Application

Platform Smart Contract mechanism	
Language	*Go; JavaScript; Python;Rust; ...*
Turing Complete?	*Yes*
Compiler	*Go; Rust; …*
Runtime VM	*Process or Docker; …*
DevTools	*Smart Contract Templates (transaction family), …*
Extra Tool(s)	*Hyperledger Caliper (performance benchmarking); Hyperledger Burrow(smart contract migration)*
Lifecycle	*Manually managed*
Description	*A transaction family includes a transaction processor to define the business logic for your application, a data model to record and store data and a client to handle the client logic for your application.*

Section 4 Protocol

Platform AAA Management	
Account type	*Identity; address; …*
Distributed ID	*The private key's associated public key be formatted as a hexadecimal string to prove your identity on the blockchain.*
AAA support	*Transactor key permissioning,* *Validator key permissioning ,*

Description	*Transactor key permissioning controls who can submit transactions and batches, based on signing keys.* *Validator key permissioning controls which nodes are allowed to establish connections to the validator network.* *The data of state is accessed using an addressing scheme that an address begins with a namespace prefix and the hex-encoded hash values of the string or strings that make up the address elements.*

Platform Consensus Mechanism	
Algorithm	*PoET; …*
Consensus mode	*Event;*
Management solution	*Internal; external*
Description	*PoET(Proof of Elapsed Time), the peer with the smallest sample wins the election, relies on secure instruction execution, a Nakamoto-style consensus algorithm that is designed to be a production-grade protocol capable of supporting large network populations.*

Platform Ledger Management	
Model	*balance;*
Extra	*Sawtooth Private UTXO, allows for assets to be tracked and traded on the Ledger;*
Description	*Sawtooth represents state for all transaction families in a single instance of a Merkle-Radix tree on each validator*

Section 5 Resources

Node Management	
Node Role	*Validator (node in Sawtooth) is the component ultimately responsible for validating batches of transactions, combining them into blocks, maintaining consensus with the network, and coordinating communication between clients, other validators, and transaction processors.*
Joining	*Validator Registry transactions are sent to add new validators to the network.*

Leaving	*Validator Registry transactions are sent to let validators leave the network.*
Role changing	*NA*
Description	*Validator key permissioning controls which nodes are allowed to establish connections to the validator network.*

Platform Data Storage Mechanism	
Mass storage mitigation[1]	*NA…*
Decentralized Data Storage Support	*Custom formatted file;*
Data Privacy Solution	*NA*
Tamper Proof (tamper cost)	*51% of the nodes decide to tamper (PoeT is a Nakamoto-style consensus algorithm).*
Description	*Further description if any*

Platform Network Management	
Node Scalability	*Node scale*
Network Structure	*Distributed; Flexible; …*
Network Discovery Protocol	*Kademlia-like; Private …*
Byzantine Node Accepted?	*Yes*
P2P?	*Yes*
Data Exchange Protocol	*Gossip; ...*
Description	*Further description if any*

Section 6 Utils

Platform Messaging Mechanism

[1] On chain storage cost much, solution/mechanism to resolve the problem of large cost of mass storage from node perspective. E.g., data maintenance, data storage and data cleaning.

Protocol Type	*ZeroMQ Message Transfer Protocol;*
Description	*ZeroMQ includes a TLS-like certificate exchange mechanism and protocol encryption capability*

Platform Crypto Libraries	
Secure Network Connection Type	*ZeroMQ TLS-like;*
Cipher Suites	*ECDSA;*
Description	*Sawtooth uses OpenSSL Toolkit. ECDSA key using the secp256k1 curve. Also it will share more algorithms provided by Hyperledger Ursa in the future.*

Section 7 Operation & Maintenance

Platform system management – Node	
Log	*Yes*
Monitoring	*Display Sawtooth metrics with Grafana, using InfluxDB to store the metrics data.*
Description	*Sawtooth network allows nodes with different versions co-exist and can set the Allowed Transaction Types;*

Platform system management – Chain Network	
Permission Control	*local validator configuration and network-wide on-chain permissioning*
Auditing	*NA*
Supervisory Support	*NA*
Description	*Sawtooth network allows nodes with different versions co-exist and can set the Allowed Transaction Types;*

Section 8 External Resource Management

Platform External Resource Management

Interoperation solution	*NA*
Description	*NA*

Section 9 Extensions

Platform Extensions - optional	
[the following list can be duplicated for multiple extensions]	
Name	*Smart Contract Support:*
Extension type[2]	*Internal;*
Extension mode[3]	*capability (vertical)*
Solution	*Ethereum Contract Compatibility with Seth*
Serve domain	*vertical: Smart Contract*
Description	*Seth, extends the interoperability of the Sawtooth platform to Ethereum. EVM (Ethereum Virtual Machine) smart contracts can be deployed to Sawtooth using the Seth transaction family.*

[2] Standing from DLT system instance perspective, any extension inside the instance is marked as "internal", while any extension outside the instance is marked as "external"

[3] All extension instances are equal (with similar capability and functional features), targeting for the scalability of DLT instance, marked as "horizontal"; extensions with different functional features, targeting to enforce the capability of DLT instance, marked as vertical. Extension type and mode pair(s) is/are used to describe the extension as to the whole DLT system. E.g., sharding (internal – horizontal), lightening – BTC (external – vertical), Corda Contract (internal – vertical).

Attachment IX – Architecture Mapping of LACChain

Section 1 Summary

Platform summary	
Platform ID	*LACChain*
Status/Revision	*Pro Test-net*
Type	*Public-permissioned*
Domain	*Multi-sector, multi-community*
Description	*LACChain is a public-permissioned infrastructure developed by a global alliance led by the IDB Lab [1] to offer public, authenticated and regulated networks in Latin America and the Caribbean with digital identity and tokenized regulated fiat money as native layers. LACChain has three main techno-legal components: LACChain DLT, LACChain ID and LACChain TFM. [2]* *LACChain DLT: A set of public-permissioned, decentralized, transparent, interoperable, zero transaction fee, regulated and quantum-safe blockchain networks. Using Pantheon [3] and Quorum [4] as core software, LACChain DLT is developing specific taxonomy composed by validator, gear, writer and observer nodes; smart-contract-based permissioning managed by the Satellite Permissioning Committee (SPC) and the Core Permissioning Committee (CPM); a DAPP for managing permissioning; cloud integration for node deploy and maintenance; a smart-contract-based gas schema to prevent network collapse; and tools to monitor nodes activity and data analytics tools, among others features.* *LACChain ID: LACChain ID is a native layer of the LACChain infrastructure that enables and requires the identification and authentication of every node, using LACChain standards aligned with international standards from W3C, EEA and IEEE. LACChain ID is also developing an application for end-user self-sovereign identity (SSI) to be used to authenticate in any platform.* *LACChain TFM: LACChain TFM is a native layer of the LACChain infrastructure that enables the issuance and transference of tokenized fiat money. This tokenized money will be issued and controlled by regulated and authorized financial institutions, backed-up by fiat money and represented by digital tokens.* [1] https://www.iadb.org/en/news/global-alliance-promote-use-blockchain-latin-america-and-caribbean [2] https://medium.com/@lacchain.official/lacchain-la-internet-del-valor-74bdb649095

	[3] https://github.com/PegaSysEng/pantheon [4] https://github.com/jpmorganchase/quorum

Section 2 Governance & Compliance Functions

Platform governance	
Governance Type	*Public-permissioned*
Chain Network Admin	*NA/Community of authenticated and regulatory compliant entities (public)*
Pledge (cost of malicious action)	*Stake*
Tamper Proof (tamper cost)	>50%
Description	*LACChain is developing a techno-legal framework to operate the infrastructure at two layers. The first layer consists of core nodes that maintain the layer, permission new nodes and apply the consensus protocols. These nodes will guarantee services under SLAs and other legal agreements. Only the LACChain Partners will be permissioned as core nodes. LACChain Partners are those that contribute to the LACChain program. [5]* *The second layer consists of satellite nodes that run applications and generate transactions. Parties operating these nodes will be legally responsible for their activity and the services they provide. Any entity can deploy and maintain satellite nodes.* *The admission of new nodes is responsibility of the Satellite Permissioning Committee (SPC) and the Core Permissioning Committee (CPC). [6]* *At an infrastructure level, LACChain is enabling smart-contract-based permissioning. This allows the SPC and CPC to manage the permissioning of new nodes via DAPP. Once new nodes are permissioned, a DID and a set of verifiable credentials is issued to the node owner. [7]* [5] https://medium.com/@lacchain.official/lacchains-networks-taxonomy-651138a70346 [6] https://medium.com/@lacchain.official/lacchains-permissioning-protocols-f18e6290949a [7] https://medium.com/@lacchain.official/lacchains-networks-roadmap-600b58872e43

Platform trust endorsement policy

Type	*Legal agreements*
Tool	*SLAs, digital identity, smart contracts, off-chain legal agreements*
Policy	*LACChain techno-legal framework* *National regulations from Latin America and the Caribbean countries*

Economic Model (optional)	
Price Model to Deploy Contracts and do Transactions	*Zero transaction fee* *Membership model for participating in LACChain, which will include not only access to the infrastructure but also to additional services as training or data visualization*
Who pays the costs of the network	*Users, developers, nodes, memberships*
Monetary Policy of Tokens	*Non-zero gas model to manage the use of the infrastructure in a responsible way*
Rights of Tokens	*To be defined*

Section 3 Application

Platform Smart Contract mechanism	
Language	*Solidity*
Turing Complete?	*Yes – Solidity*
Compiler	*Solcjs - Solidity*
Runtime VM	*EVM – Ethereum Virtual Machine*
DevTools	*Development: Visual Studio Code; Sublime; Remix;* *Build framework: Truffle, Embark, Remix, Web3j, Web3js, ethersjs* *Test framework: Truffle, Embark, Remix, Web3j, Web3js, ethersjs*
Extra Tool(s)	*Any Ethereum compatible Blockchain Explorer.* https://github.com/Councilbox/cbx-quorum-explorer https://github.com/gobitfly/etherchain-light
Lifecycle	*LACChain techno-legal framework will address the death of the network* *The developers of the applications and the satellite nodes providing services will be responsible for the services provided*
Description	*LACChain techno-legal framework*

	Github repository

Section 4 Protocol

Platform AAA Management	
Account type	*Authentication mandatory at a node level and recommended at a user level*
Distributed ID	*There are two types of accounts which share the same address space: externally owned accounts and contract accounts. Externally owned accounts are controlled by public-private key pairs and have no code. Contract accounts are controlled by the code stored together with the account – the smart contract code* *User should generate an externally owned account using a local software/hardware in order to keep the private key private* *Contract accounts are created during deploy* *LACChain ID will provide a suite of standards and recommendations in the different areas composing digital identity, as DIDs, Verifiable Claims, Verifiable Presentations, Levels of Assurance, Key Management System or Data Privacy* *LACChain ID will provide an application following the LACChain ID standards and recommendations, for users to generate and manage DIDs and certificates. This application provided will be intended to be used by other applications or platforms*
AAA support	*LACChain ID*
Description	*LACChain ID*

Platform Consensus Mechanism	
Algorithm	*IBFT2.0*
Consensus mode	*Event, state*
Management solution	*Internal*
Description	*LACChain techno-legal framework*

Platform Ledger Management	
Model	*balance*
Extra	*MPT support - modified Merkle Patricia tree (trie)*

Description	*Each account has a storage, a persistent memory area. A contract can neither read nor write to any storage apart from its own.* https://github.com/PegaSysEng/pantheon/tree/master/ethereum/trie/src/main/java/tech/pegasys/pantheon/ethereum/trie https://github.com/jpmorganchase/quorum/tree/master/trie

Section 5 Resources

Node Management	
Node Role	*Validator node: Apply consensus protocol. Maintain the ledger.* *Gear node: Connect validator nodes with satellite and observer nodes. Set up new nodes.* *Writer node: Generate transactions. Provide services to end-users.* *Observer nodes: Read the blockchain.*
Joining	*Validator and gear nodes will be required to sign SLAs and prove technical capabilities. Only LACChain Partners will be allowed to run these nodes.* *Writer and observer nodes will be legally accountable for the activity of their node and the transactions it generates.* *Every node must be identified and operate authenticated.*
Leaving	*Validator and gear nodes have to notify as agreed in the SLAs.* *Writer and observer nodes can leave at any time without notification.*
Role changing	*A node can change the role as long as it satisfied the conditions for performing the new role.*
Description	*LACChain techno-legal framework*

Platform Data Storage Mechanism	
Mass storage mitigation[1]	*Concept of Gas* *Some operations may have negative gas cost, for example kill a contract.*
Decentralized Data Storage Support	*IPFS, cloud-services*
Data Privacy Solution	*LACChain techno-legal framework will specify the policies related to data privacy*

[1] On chain storage cost much, solution/mechanism to resolve the problem of large cost of mass storage from node perspective. E.g., data maintenance, data storage and data cleaning.

Description	*The fundamental unit of computation is called "gas"; The fee system is to require a person to pay proportionately for every resource that they consume, including computation, bandwidth and storage; However, gas fee will not be associated with a monetary fee; Gas will be distributed at no cost for the users, ensuring the correct functioning of the network*

Platform Network Management	
Node Scalability	*Thousands*
Network Structure	*Distributed*
Network Discovery Protocol	*Kademlia-like;*
Byzantine Node Accepted?	*Yes*
P2P?	*Yes*
Data Exchange Protocol	*RLPx*
Description	*RLPx transport protocol, a TCP-based transport protocol used for communication among Ethereum nodes. The protocol carries encrypted messages belonging to one or more 'capabilities' which are negotiated during connection establishment.* https://github.com/jpmorganchase/quorum/tree/master/rlp https://github.com/PegaSysEng/pantheon/tree/master/ethereum/rlp https://pdos.csail.mit.edu/~petar/papers/maymounkov-kademlia-lncs.pdf

Section 6 Utils

Platform Messaging Mechanism	
Protocol Type	*RPC* *GraphQL*
Description	*JSON-RPC is a stateless, lightweight remote procedure call (RPC) protocol. Primarily this specification defines several data structures and the rules around their processing. It is transport agnostic in that the concepts can be used within the same process, over sockets, over HTTP, or in many various message passing environments. It uses JSON (RFC 4627) as data format.* https://github.com/PegaSysEng/pantheon/blob/master/docs/Pantheon-API/JSON-RPC-API.md https://github.com/jpmorganchase/quorum/blob/master/docs/Security/Framework/Quorum%20Network%20Security/Node.md

Platform Crypto Libraries

Secure Network Connection Type	*Communication via public Internet (TCP + UDP).*
Cipher Suites	*ECDSA (Elliptic Curve Digital Signature Algorithm) for it's public-key cryptography and KECCAK-256 for hashing* *New quantum-safe cipher suites will be introduced*
Description	*Geth (modified) and Pantheon Enterprise Ethereum Client uses UDP connection to exchange information about the P2P network. After establishing peer connections, nodes exchange blockchain information via encrypted and authenticated TCP connections.* https://github.com/PegaSysEng/pantheon/blob/master/docs/index.md https://github.com/jpmorganchase/quorum/tree/master/ethclient

Section 7 Operation & Maintenance

Platform system management – Node	
Log	*Yes*
Monitoring	http://netstats.testnet.lacchain.io/
Description	*Network status allows anyone to see the performance and number of nodes and where they are located.* *There is no special nodes (masternodes, special block producers etc) in the network.* https://github.com/PegaSysEng/pantheon/blob/master/docs/Monitoring/Monitoring-Performance.md https://github.com/jpmorganchase/quorum/blob/master/docs/Privacy/Tessera/Usage/Monitoring.md

Platform system management – Chain Network	
Permission Control	*Smart contract-based*
Auditing	*Information public available in blockchain explorers* *Public information from DIDs, verifiable credentials and similar will be captured and used to measure the social impact achieved through the use of LACChain*
Supervisory Support	*N/A*
Description	*N/A*

Section 8 External Resource Management

Platform External Resource Management

Interoperation solution	
Description	

Section 9 Extensions

Platform Extensions – optional	
[the following list can be duplicated for multiple extensions]	
Name	
Extension type[2]	
Extension mode[3]	
Solution	
Serve domain	
Description	

[2] Standing from DLT system instance perspective, any extension inside the instance is marked as "internal", while any extension outside the instance is marked as "external"

[3] All extension instances are equal (with similar capability and functional features), targeting for the scalability of DLT instance, marked as "horizontal"; extensions with different functional features, targeting to enforce the capability of DLT instance, marked as vertical. Extension type and mode pair(s) is/are used to describe the extension as to the whole DLT system. E.g., sharding (internal – horizontal), lightening – BTC (external – vertical), Corda Contract (internal – vertical).

Attachment X – Architecture Mapping of Masterchain

Section 1 Summary

Platform summary	
Platform ID	*Masterchain*
Status/Revision	*Masterchain 0.3.0*
Type	*Private, Consortium*
Domain	*Financial*
Description	*Masterchain is a blockchain platform developed for banks and other financial institutions by forming a consortium. It allows for the data and information to be exchanged between the parties. It is developed by FinTech Association of Russian financial institutions.* *It is built using a fork of Ethereum blockchain.*

Section 2 Governance & Compliance Functions

Platform governance	
Governance Type	*Permissioned*
Chain Network Admin	*FinTech Association, a consortium that includes Bank of Russia and other financial institutions in Russia (e.g. Alfa Bank, Bank Otkritie, Tinkoff Bank, and Qiwi)*
Pledge (cost of malicious action)	*Computer Power – measured by hash rates*
Tamper Proof (tamper cost)	>50%
Description	

Platform trust endorsement policy	
Type	*Law, Consensus agreement by the consortium*
Tool	*Operation fully depends on the FinTech Association's central server that controls mining and consensus* *https://www.coindesk.com/russias-largest-bank-is-quitting-central-banks-blockchain-project*

Policy	*N/A*

Economic Model (optional)	
Price Model to Deploy Contracts and do Transactions	*Smart contracts are deployed in the ecosystem. They are charged per transaction.* *Transaction fees are paid in gas.*
Who pays the costs of the network	*Network participants/Application providers*
Monetary Policy of Tokens	*Unlimited supply, all held by FinTech Association, provided to consortium members upon request* *Tokens for paying gas are distributed by the association's node. They are distributed among participants for free, and wallets are refilled from time to time automatically.* *Network participants do not have to rely on AFT for tokens, they can share it with each other.* *https://www.coindesk.com/russias-largest-bank-is-quitting-central-banks-blockchain-project*
Rights of Tokens	*N/A*

Section 3 Application

Platform Smart Contract mechanism	
Language	*Solidity*
Turing Complete?	*Yes*
Compiler	*Solcjs - Solidity*
Runtime VM	*EVM;*
DevTools	*Development: Visual Studio Code; Sublime; Remix;* *Build framework: Truffle, Embark, Remix* *Test framework: Truffle, Embark, Remix*
Extra Tool(s)	*Explorer (Block data view): Masterchain Explorer*
Lifecycle	*The developer has to code if the contract can stop or be killed. It is not possible to update the deployed smart contract, but there are recommendations to that.*

Description	

Section 4 Protocol

Platform AAA Management	
Account type	*Address;*
Distributed ID	*There are two types of accounts which share the same address space: externally owned accounts and contract accounts. Externally owned accounts are controlled by public-private key pairs and have no code. Contract accounts are controlled by the code stored together with the account – the smart contract code.* *User should generate an externally owned account using a local software/hardware in order to keep the private key private;* *Contract accounts are created during deploy.*
AAA support	*N/A*
Description	*The rational is that there are so many possible addresses that the probability of collision is negligible.*

Platform Consensus Mechanism	
Algorithm	*PoW;*
Consensus mode	*Event;*
Management solution	*Internal;*
Description	

Platform Ledger Management	
Model	*balance;*
Extra	*MPT support - modified Merkle Patricia tree (trie)*
Description	*Each account has a storage, a persistent memory area. A contract can neither read nor write to any storage apart from its own.* *From a block header there are 3 roots from 3 MPT: stateRoot, transactionsRoot and receiptsRoot.*

Section 5 Resources

Node Management	
Node Role	*Full Nodes and Full archiving nodes.*
Joining	*The node has to sync with the network and start to participate with the permission: must be included in the node whitelist smart-contract, managed by FinTech Association.*
Leaving	*Nodes can stop working at any time.*
Role changing	*FinTech Association manages nodes' roles based on consortium decisions.*
Description	*Consortium's participants who need more gas are not able to mine more tokens, or it will be switched off from the network.* *https://www.coindesk.com/russias-largest-bank-is-quitting-central-banks-blockchain-project*

Platform Data Storage Mechanism	
Mass storage mitigation	*Concept of Gas* *Some operations may have negative gas cost, for example kill a contract.*
Decentralized Data Storage Support	*MCMS – Masterchain Confidential Messaging System*
Data Privacy Solution	*N/A*
Description	*The fundamental unit of computation is called "gas"; The fee system is to require a person to pay proportionately for every resource that they consume, including computation, bandwidth and storage;*

Platform Network Management	
Node Scalability	*Thousands*
Network Structure	*Distributed*
Network Discovery Protocol	*Kademlia-like;*
Byzantine Node Accepted?	*Yes*
P2P?	*Yes*
Data Exchange Protocol	*RLP over TLS*
Description	https://github.com/ethereum/wiki/wiki/Kademlia-Peer-Selection

	RLP transport protocol, a TCP-based transport protocol used for communication among Ethereum nodes. The protocol carries messages belonging to one or more 'capabilities' which are negotiated during connection establishment. Messaging security is provided with a TLS connection layer.

Section 6 Utils

Platform Messaging Mechanism	
Protocol Type	*RPC*
Description	*JSON-RPC is a stateless, lightweight remote procedure call (RPC) protocol. Primarily this specification defines several data structures and the rules around their processing. It is transport agnostic in that the concepts can be used within the same process, over sockets, over HTTP, or in many various message passing environments. It uses JSON (RFC 4627) as data format.* https://github.com/ethereum/wiki/wiki/JSON-RPC

Platform Crypto Libraries	
Secure Network Connection Type	*Communication via public Internet (TCP with TLS + UDP).*
Cipher Suites	*Russian GOST 34-10.2012 for it's public-key cryptography and GOST 34-11.2012 for hashing*
Description	*Meth (The official Masterchain client node software) uses UDP connection to exchange information about the P2P network. After establishing peer addresses, Meth nodes exchange blockchain information via encrypted and authenticated TLS connections.*

Section 7 Operation & Maintenance

Platform system management – Node	
Log	*Yes*
Monitoring	*Masterchain Explorer*
Description	

Platform system management – Chain Network	
Permission Control	*Whitelist of nodes which are allowed to connect to the network and which are allowed to produce blocks*
Auditing	*N/A*
Supervisory Support	*N/A*

Description	*Masterchain Explorer shows information about blocks, transactions, tokens, smart contracts, addresses and the history of its transactions. Masterchain Explorer is operated and developed by FinTech Association and can be accessed by consortium members.*

Section 8 External Resource Management

Platform External Resource Management	
Interoperation solution	*Masterchain Confidential Messaging Service*
Description	*The system of smart contracts describing network participants, their roles and data objects which can be accessed by them*

Section 9 Extensions

Platform Extensions – optional	
Name	*Masterchain Confidential Messaging Service*
Extension type[1]	*internal*
Extension mode[2]	*vertical*
Solution	*The web service implementing GraphQL protocol to provide access to read and write data objects based on the rules of MCMS smart contract system.*
Serve domain	*Network/Application*
Description	

[1] Standing from DLT system instance perspective, any extension inside the instance is marked as "internal", while any extension outside the instance is marked as "external"

[2] All extension instances are equal (with similar capability and functional features), targeting for the scalability of DLT instance, marked as "horizontal"; extensions with different functional features, targeting to enforce the capability of DLT instance, marked as vertical. Extension type and mode pair(s) is/are used to describe the extension as to the whole DLT system. E.g., sharding (internal – horizontal), lightening – BTC (external – vertical), Corda Contract (internal – vertical).

Attachment XI – Architecture Mapping of Monero

Section 1 Summary

Platform summary	
Platform ID	*Monero/XMR*
Status/Revision	*Mainnet – v0.14.1.0*
Type	*Public*
Domain	*Peer to peer payments, Financial*
Description	*Monero is an open-source, decentralised, permissionless, fungible cryptocurrency.* https://github.com/monero-project/monero

Section 2 Governance & Compliance Functions

Platform governance	
Governance Type	*Permissionless*
Chain Network Admin	*Community (public)*
Pledge (cost of malicious action)	*Resources (hardware + electricity) – measured by hash rate (H/s)*
Tamper Proof (tamper cost)	>50% of network H/s
Description	*Monero is an open-source project to which everybody is allowed to contribute. There are no restrictions on participation. There is no formal entity (neither for-profit nor foundation) governing the project.* *Improvements to the protocol are discussed on public communication platforms such as GitHub, IRC, Reddit. Regular development meetings, accessible to all, take place on IRC and allow to discuss relevant topics or gauge interest for specific features.* *As one of the defining characteristics of the project since the beginning, Monero is set to undergo regular "protocol upgrades" (also called hard forks) allowing to deploy protocol upgrades.* *https://getmonero.org*

Platform trust endorsement policy	
Type	*Elliptic curve cryptography*
Tool	*Curve Ed25519*
Policy	*CryptoNote protocol, cryptographic signature, one-dimensional distributed acrylic graph (DAG), Schnorr-style multilayered linkable spontenerous anonymous group signatures (MLSAG), Schnorr-style Borromean ring signatures, amounts concealed with Pedersen commitments* *https://getmonero.org/library/Zero-to-Monero-1-0-0.pdf*

Economic Model (optional)	
Price Model to Deploy Contracts and do Transactions	*Minimum fee per kB: fkB =* $f_b^{kB} * (300kB/M100) * (B^{actual}/10)\ b$ Transaction size is round up to the nearest kB.
Who pays the costs of the network	*User*
Monetary Policy of Tokens	*Block rewards uses a formula: [(L-M) >> 19] /* 10^{12} *Steady rate 'tail emissiom' supply of 18.132 million coins by ca. end of May 2022; and 0.3 XMR per minute thereafter.* *Monero is mined with respect to 'coinbase rewards'. Miners receive the coinbase reward along with write permissions for appending the next block to the ledger.*
Rights of Tokens	*N/A*

Section 3 Application

Platform Smart Contract mechanism	
Language	*N/A*
Turing Complete?	*N/A*
Compiler	*N/A*
Runtime VM	*N/A*
DevTools	*N/A*
Extra Tool(s)	*N/A*
Lifecycle	*N/A*

Description	*N/A*

Section 4 Protocol

Platform AAA Management	
Account type	*UTXO*
Distributed ID	*There is no identification system attached to wallet addresses or transactions. The project is entirely permissionless and anyone is free to participate.* *Addresses are based on a dual-key mechanism allowing them to never appear on the blockchain. An address is used by senders of transactions to generate a derived one-time destination. These one-time destinations cannot be related by observers to a particular address or linked together even if they are generated from the same address.*
AAA support	*N/A*
Description	*This dual-key mechanism (also called "stealth addresses") is one of the mechanisms used by Monero to protect users' privacy and ensures token fungibility. More specifically, this mechanism obfuscates senders, recipients and amount of transactions.*

Platform Consensus Mechanism	
Algorithm	*Currently CNv4 (scheduled update to RandomX)*
Consensus mode	*Proof of Work (PoW)*
Management solution	*Internal*
Description	*Monero has committed to aim at Application Specific Integrated Circuit (ASIC) resistance to increase the decentralisation properties of the network. As the ASIC industry tends to be centralized, and there are many more CPUs in the world than ASICs, a CPU friendly PoW algorithm should lower the barrier to entry for miners. As such, the project has developed a new PoW algorithm. In particular, the next protocol upgrade (~October) should include "RandomX", a state-of-the-art algorithm in that domain.* *https://github.com/tevador/RandomX*

Platform Ledger Management	
Model	*Balance*
Extra	*Merkle tree*

Description	*Block structure consists of three parts: block header, base transaction body, and a list of transaction identifiers. The block header has information about the major (protocol version)/minor (miner voting mechanism, albeit currently unused) header version, block creation time, identifier of previous block, and nonce.*

Section 5 Resources

Node Management	
Node Role	*Full Nodes and Pruned Nodes.*
Joining	*Anyone can join the network. A new node simply needs to synchronized with the current state of the chain in order to fully participate.*
Leaving	*Any node can leave at any time.*
Role changing	*N/A*
Description	*N/A*

Platform Data Storage Mechanism	
Mass storage mitigation	*Blockchain pruning - non-critical information removed from local storage. Pruned nodes remove approximately 2/3 of the total blockchain size.*
Decentralized Data Storage Support	*N/A*
Data Privacy Solution	*Ring Signatures, Stealth Addresses, Ring Confidential Transactions (incl. Bulletproofs).*
Description	*A number of privacy-preserving mechanisms are deployed within the Monero protocol. This ensures as little distinguishing information is published to the public ledger as possible whilst maintaining information verifiability, accountability, and audibility.* *This mitigates against PII (personal identifying information) encroachments, breaches of current information and data legislation (GDPR), corporate data harvesting, information surveillance, and mitigates against any destabilization of the fungible nature of the value exchange mechanism.* *These mechanisms are critical to Monero's aim at preserving privacy and constitute a major part of its technical design. The project is constantly pushing the state-of-the art in that domain to ensure these fundamental characteristics.*

Platform Network Management

Node Scalability	*No upper bound. (current ~2000 nodes).*
Network Structure	*Distributed*
Network Discovery Protocol	*Levin Protocol (https://github.com/xmrdsc/py-levin)*
Byzantine Node Accepted?	*Yes*
P2P?	*Yes*
Data Exchange Protocol	*TCP*
Description	*Nodes request and share list of peers from other nodes.*

Section 6 Utils

Platform Messaging Mechanism	
Protocol Type	*RPC*
Description	*JSON-RPC is a stateless, lightweight remote procedure call (RPC) protocol. Primarily this specification defines several data structures and the rules around their processing. It is transport agnostic in that the concepts can be used within the same process, over sockets, over HTTP, or in many various message passing environments. It uses JSON (RFC 4627) as data format.* *https://github.com/monero-project/monero/wiki/Daemon-RPC-documentation* *https://github.com/monero-project/monero/wiki/Wallet-RPC-Documentation*

Platform Crypto Libraries	
Secure Network Connection Type	*P2P; RPC; DNS*
Cipher Suites	*ECDHE-ECDSA-CHACHA20-POLY1305-SHA256; ECDHE-ECDSA-CHACHA20-POLY1305; ECDHE-ECDSA-AES256-GCM-SHA384; ECDHE-ECDSA-AES128-GCM-SHA256; ECDHE-RSA-CHACHA20-POLY1305; ECDHE-RSA-AES256-GCM-SHA384; ECDHE-RSA-AES128-GCM-SHA256*
Description	*Current code uses CHACHA20-POLY1305 by default (with option of above suites)*

Section 7 Operation & Maintenance

Platform system management – Node

Log	*Yes*
Monitoring	*monerod*
Description	*monerod is the daemon client that manages all interactions with the Monero network. It offers necessary features to act as a node. Additionally, it also acts as the interface between wallet software and the Monero network. This allows the wallet software to be completely decoupled from the node management daemon.*

Platform system management – Chain Network	
Permission Control	*N/A*
Auditing	*Auditing mechanisms are self contained within each wallet and pertains to each wallet address managed by the wallet software. There is no network wide auditing system. However, an important auditing mechanism is associated to addresses (or individual transactions). A 'view key' is associated to an address, which gives anyone holding it auditing functionality -- while not allowing any wallet spend authority.* *This functionality would allow, for example, an NGO to publish its view key along with a donation address, so that anyone who wishes can audit received donations. It also allows businesses and individuals to comply with certain regulations, if they are required, to provide visibility over wallet information.*
Supervisory Support	*N/A*
Description	*N/A*

Section 8 External Resource Management

Platform External Resource Management	
Interoperation solution	*N/A*
Description	*N/A*

Section 9 Extensions

Platform Extensions – optional	
[the following list can be duplicated for multiple extensions]	
Name	*N/A*

Extension type[1]	*N/A*
Extension mode[2]	*N/A*
Solution	*N/A*
Serve domain	*N/A*
Description	*N/A*

[1]Standing from DLT system instance perspective, any extension inside the instance is marked as "internal", while any extension outside the instance is marked as "external"

[2]All extension instances are equal (with similar capability and functional features), targeting for the scalability of DLT instance, marked as "horizontal"; extensions with different functional features, targeting to enforce the capability of DLT instance, marked as vertical. Extension type and mode pair(s) is/are used to describe the extension as to the whole DLT system. E.g., sharding (internal – horizontal), lightning – BTC (external – vertical), Corda Contract (internal – vertical).

Attachment XII – Architecture Mapping of Ontology

Section 1 Summary

Platform summary	
Platform ID:	*ONT*
Status/Revision:	V 1.7.0
Type:	*Public – Ontology* *Consortium – Ontology 2B*
Domain:	Blockchain infrastructure

Section 2 Governance & Compliance Functions

Platform governance - Ontology	
Governance Type:	*Permissionless*
Chain Network Admin:	*Ontology community*
Pledge (cost of malicious action):	*Stake*
Description:	*Node with Stake can contribute to consensus network, the stake can be frozen on malicious actions, community vote to decide upon malicious actions.*

Platform trust endorsement policy - Ontology	
Type:	*Tokenomics*
Tool:	ONT/ONG
Policy:	*ONT as stake to become node, and ONG as basic DLT service fee. ONT can be frozen on malicious actions.*

Platform governance – Ontology 2B	
Governance Type:	*Permissioned*
Chain Network Admin:	*Entity*
Pledge (cost of malicious action):	*Liquidated damages*

Description:	*Agreement will be placed to buy in the node into network, and any malicious action will go to law process*

Platform trust endorsement policy – Ontology 2B	
Type:	*Law / Agreement*
Tool:	*ONT ID (with CA) + agreement*
Policy:	-

Economic Model (optional)	
Price Model to Deploy Contracts and do Transactions	*ONG as utility token for gas inside Ontology.* *Ref., fee model[1], deployment[2].*
Who pays the costs of the network	*Users*
Monetary Policy of Tokens	*1 billion ONT total, 1 billion ONG bound with ONT, ONG unbinding curve equation manage ONG unbinding per second[3].*
Rights of Tokens	*ONT as Stake and ONG as utility token inside Ontology multi-chain network*

Section 3 Application

Platform Smart Contract mechanism	
Language	*NEOVM: Python; C#; Javascript;* *WASM: C++; Rust*
Turing Complete?	*Yes*
Compiler	*NEOVM: Python; C#; Javascript;* *WASM: C++; Rust*
Runtime VM:	*NEOVM; WASM*

[1] https://github.com/ontio/ontology-smartcontract/blob/master/smart-contract-tutorial/feemodel_en.md

[2] https://ontio.github.io/documentation/Smart_Contract_Deployment_en.html#calculate-the-gas-consumed-by-deploying-a-smart-contract

[3] https://medium.com/ontologynetwork/triones-node-incentive-model-dbcb175f4728

DevTools	*SDK[4], SmartX[5], Punica Suite[6]*
Extra Tool(s):	*Explorer[7]*
Lifecycle	*Live within an app call*
Description:	*Support NEOVM and WASM with multiple programming languages compiler, as well as some language translators, from bytecode to NEOVM/WASM bytecode.*

Section 4 Protocol

Platform AAA Management	
Account type:	*Identity; address*
Distributed ID:	*ONT ID*
AAA support:	*ONT ID suite (ONTID[8], OntPass[9], TrustAnchor connector[10])*
Description:	*ONT ID as identifier of entities.*

Platform consensus mechanism	
Algorithm:	*VBFT (Byzantine Fault Tolerance with Verifiable Randomness)*
Consensus mode:	*Event*
Management solution:	*Internal*
Description:	*VBFT achieves chain scalability by consensus node selection with VRF, anti-attack ability by randomness and PoS, and fast state finality with BFT.* *Plus, in Ontology 2B, use predefined stake in agreement as PoS in consensus*

[4] https://dev-docs.ont.io/#/docs-en/Punica/punica

[5] https://dev-docs.ont.io/#/docs-en/SmartX/00-overview

[6] https://dev-docs.ont.io/#/docs-en/SDKs/00-overview

[7] https://explorer.ont.io/

[8] https://pro-docs.ont.io/#/docs-en/ontid/overview

[9] https://pro-docs.ont.io/#/docs-en/ontpass/overview

[10] https://pro-docs.ont.io/#/docs-en/taconnector/overview

Platform ledger management			
Model:	*balance*	**Extra:**	*MPT on sub-chain and sharding*
Description:	*By default, Ontology uses balance model to store data. Can support UTXO in sub-chain(s). To support SPV, apply MPT in sub-chain and sharding.*		

Section 5 Resources

Node Management	
Node Role	*Candidate node; consensus node*
Joining	*synchronized node with hardware and software installed* - *certain ONT as stake in address, peer admin address/wallet* - *address for node operating, peer runtime address/wallet* - *ONT ID combine with addresses above* *register candidate, have ONT staked, approved by operator role (manually by Ontology Foundation for first network size, delegate to AI robot contract later)*
Leaving	*Quit node and withdraw ONT staked*
Role changing	*Stake to certain rank and upgrade from candidate node to consensus node; for lower rank, downgrade from consensus node to candidate node*
Description:	-

Platform data protection - core	
Mass storage mitigation[11]	*Pay on data storing*
Decentralized Data Storage Support	*No*
Data Privacy Solution	*ZKP POC done, MPC in research*
Tamper Proof (tamper cost):	*stop service, average PoS * 1/3 network scale (nodes)* *tamper,*

[11] On chain storage cost much, solution/mechanism to resolve the problem of large cost of mass storage from node perspective.

	*average PoS * 2/3 network scale for data tamper*
Description:	*Ontology-crypto lib, supports multiple signature schemas[12] and anonymous credential[13]*

Platform Network hypothesis	
Node Scalability:	*Up to 50,000 nodes*
Byzantine Node Accepted? :	*Yes;*
Network Structure	*Flexible*
P2P? :	*Yes*
Network Discovery Protocol	*DHT*
Data Exchange Protocol	-
Description:	*Theoretically there's no limitation of node count. However, to satisfy Byzantine failure tolerance. Node scalability shall satisfy hypergeometric distribution.* *Consensus node count per block < 200, error rate < 0.00000001.*

Section 6 Utils

Platform Messaging Mechanism	
Protocol Type	*RPC; RESTful*
Description:	-

Platform Crypto Libraries	
Secure Network Connection Type	*TLS*
Cipher Suites	*Key types: ECDSA; SM2; EdDSA* *Signature schemes: SHA224withECDSA; SHA256withECDSA; SHA384withECDSA; SHA512withECDSA; SHA3-224withECDSA; SHA3-*

[12] https://github.com/ontio/ontology-crypto

[13] https://github.com/ontio/ontology-crypto/wiki/Anonymous-Credential

	256withECDSA; SHA3-384withECDSA; SHA3-512withECDSA; RIPEMD160withECDSA; SM3withSM2; SHA512withEdDSA
Description:	*Cryptography Library for Ontology Network is referenced*[14]

Section 7 Operation & Maintenance

Platform system management – node	
Log	*yes*
Monitoring	*explore*[15]
Description:	*[Operation and Maintenance] -*

Platform system management – chain network	
Permission Control:	*Yes*
Auditing:	*N/A*
Supervisory Support:	*N/A*
Description:	*[Operation and Maintenance] Native auth and global parameter contract.*

Section 8 External Resource Management

Platform external data exchange – application service	
Interoperation solution:	*ONT ID + data token solution to map data with token.* *Semantic web solution (ontology data model) to support data interoperability*
Description:	-

Section 9 Extensions

Platform Extensions
[the following list can be duplicated for multiple extensions]

[14] https://github.com/ontio/ontology-crypto

[15] https://explorer.ont.io/

Name	*Ontology sharding*
Extension type[16]	*Internal*
Solution	*sharding*
Extension mode[17]	*horizontal*
Serve domain	*Computing capability*
Description:	*Ontology sharding supports shard on state, shard on transaction and shard on network* [18]
Name	*Ontology sidechain / ecochain*
Extension type	*External*
Solution	*side-chain*
Extension mode	*horizontal*
Serve domain	*Cross domain (chain) applications*
Description:	*Ontology ecochain serves the requirement of multiple domain requirement with different governance model*
Name	*Ontology oracle and state channel*
Extension type	*External*
Solution	*Layer 2 + oracle*
Extension mode	*Horizontal and vertical*
Serve domain	*Non-DLT applications and hybrid storage system*

16 Standing from DLT system instance perspective, any extension inside the instance is marked as "internal", while any extension outside the instance is marked as "external"

17 All extension instances are equal (with similar capability and functional features), targeting for the scalability of DLT instance, marked as "horizontal"; extensions with different functional features, targeting to enforce the capability of DLT instance, marked as vertical. Extension type and mode pair(s) is/are used to describe the extension as to the whole DLT system. E.g., sharding (internal – horizontal), lightening – BTC (external – vertical), Corda Contract (internal – vertical).

18 https://github.com/ontio/documentation/tree/master/sharding

Description:	*Ontology oracle and state channel provides the requirement of on-chain / off-chain applications and extend the performance of on-chain applications with lower cost*

Attachment XIII – Architecture Mapping of Quorum

Section 1 Summary

Platform summary	
Platform ID	*QUORUM: Go Ethereum (Geth)*
Status/Revision	*Soft-fork of Ethereum, Last Stable version v2.2.4*
Type	*Public-permissioned*
Domain	*Many sectors, e.g., Supply chain; Finance; Retail, etc…*
Description	*QUORUM is an Ethereum-based distributed ledger protocol with transaction /contract Privacy and new consensus mechanisms. That can bring the best from both worlds, every node on the network can validate every transaction on list but only exposing to relevant parties.* https://github.com/jpmorganchase/quorum

Section 2 Governance & Compliance Functions

Platform governance	
Governance Type	*It is modular with a base of BFT by validators nodes and regular nodes.*
Chain Network Admin	*Brainchild of JP Morgan*
Pledge (cost of malicious action)	*Free adoption by permission (OPEN SOURCE)*
Tamper Proof (tamper cost)	No gas by private forks.
Description	***Quorum is an Ethereum-based is an open-source platform for decentralized applications to support enterprise requirements such as privacy.*** https://github.com/jpmorganchase/quorum/blob/master/docs/Quorum%20Whitepaper%20v0.2.pdf

Platform trust endorsement policy	
Type	*Permissioned*
Tool	*Signature Validation*
Policy	*Open Source under LGPL 3.0 License*

Economic Model (optional)	
Price Model to Deploy Contracts and do Transactions	*Private control through automation. For Public state or Private State.*
Who pays the costs of the network	*Stakeholders and memberships.*
Monetary Policy of Tokens	*Non-zero gas model to manage the use of the infrastructure in a responsible way*
Rights of Tokens	*To be defined*

Section 3 Application

Platform Smart Contract mechanism	
Language	*Solidity*
Turing Complete?	*Yes – Solidity*
Compiler	*java, Solidity;*
Runtime VM	*EVM – Ethereum Virtual Machine; ABI, OVM, WAR;*
DevTools	*Quorum Blockchain explorer, Quorum Genesis, Quorum Maker, QuorumNetworkManager, ERC20 REST servie, Nethereum Qourum, web3j-quorum, Apache Camel, Quorum API. Cakeshop, quorum cloud.*
Extra Tool(s)	*Tessera is implemented in Java and it is never than Costellation (implemented in Haskell)*
Lifecycle	*Privacy Manager (Constellation/Tessera) binomial: transaction manager + Enclave.*
Description	*Cakeshop as a set of tools and APIs for working with Ethereum-like ledgers. Supports private transactions and private contracts through public/private state separation. Although private contracts work better than public ones as there is less overhead when it comes to handling private contracts. This means that Quorum private blockchain is effective.*

Section 4 Protocol

Platform AAA Management	
Account type	*Identity validation by signature.*
Distributed ID	*DID, ERC721, ERC725 and others non-fungible identities.*
AAA support	*EIP1812*
Description	*Although it is anonymous state the network has to be identified in order to peer-to-peer maintenance.*

	DIDs are welcome in Quorum and compatible with other solutions for four kinds of digital identiy: People, legal entities (NGOs, Public, Private sector, etc…) things and processes.

Platform Consensus Mechanism	
Algorithm	*PoW; PoS, BFT; HBBFT, PoA*
Consensus mode	*Pluggable RAFT, IBFT and Clique PoA*
Management solution	*Internal; external*
Description	https://github.com/jpmorganchase/quorum/blob/master/docs/Quorum%20Whitepaper%20v0.2.pdf

Platform Ledger Management	
Model	*Private control through automation. For Public state or Private state.*
Extra	*MPT support - modified Merkle Patricia tree (trie)*
Description	Smart Contracts play a crucial role whereby can be customized by business themselves.

Section 5 Resources

Node Management	
Node Role	*Validator node: Validates transactions proposals and create new block on ledger, and they keep a copy of the ledger itself (Validators Nodes only exists on IBFT mode).* *Regular Node: Exists on Raft or in IBFT (this last case considered as Non-validators nodes), responsible to store a copy of ledger and make new proposals to Validators Nodes as responsible to spread updated ledger to non-validator nodes over the network itself.* *Boot node: Permission new nodes.*
Joining	*Create a node key (enode) by using the Bootnode tool, then make a copy of static-nodes.json file into node folder, then copy the enode into the permissioned-node.json file (where all enodes of network are listed), initialize the node trough the "geth" tool , last is through an already active running node, use "addpeer()" command on "geth" tool so the node can make part of the network.*
Leaving	*Through an existing node. You could run a command called removePeer() which then will remove a node trough hes enode number..*
Role changing	*N/A*
Description	https://github.com/jpmorganchase/quorum-examples

Platform Data Storage Mechanism	
Mass storage mitigation[1]	*Off-chain.*
Decentralized Data Storage Support	*Blockchain explorer for Quorum. Swarm is also capable with Quorum and IPFS.IPFS, cloud-services*
Data Privacy Solution	*ZKP; MPC; IPFS; ZSL, ZSC and Anonymous Zether,*
Description	*Privacy by design.*

Platform Network Management	
Node Scalability	*Thousands*
Network Structure	*Distributed*
Network Discovery Protocol	*Kademlia-like;*
Byzantine Node Accepted?	*Yes*
P2P?	*Yes*
Data Exchange Protocol	*RLPx*
Description	*RLPx transport protocol, a TCP-based transport protocol used for communication among Ethereum nodes. The protocol carries encrypted messages belonging to one or more 'capabilities' which are negotiated during connection establishment.* https://github.com/jpmorganchase/quorum/tree/master/rlp https://pdos.csail.mit.edu/~petar/papers/maymounkov-kademlia-lncs.pdf

Section 6 Utils

Platform Messaging Mechanism	
Protocol Type	Transaction Manager, Peers, and Enclave use traditional TCP/UDP transport layer to communicate.
Description	*JSON-Remote Procedure Call (RPC) is a stateless, lightweight remote procedure call (RPC) protocol. Primarily this specification defines several data structures and the rules around their processing. It is transport agnostic in that the concepts can be used within the same*

[1] On chain storage cost much, solution/mechanism to resolve the problem of large cost of mass storage from node perspective. E.g., data maintenance, data storage and data cleaning.

	process, over sockets, over HTTP, or in many various message passing environments. https://github.com/jpmorganchase/quorum/blob/master/docs/Security/Framework/Quorum%20Network%20Security/Node.md

Platform Crypto Libraries	
Secure Network Connection Type	*Communication via public Internet (TCP + UDP).*
Cipher Suites	*ECDSA (Elliptic Curve Digital Signature Algorithm) for it's public-key cryptography and KECCAK-256 for hashing*
Description	https://github.com/jpmorganchase/quorum/tree/master/ethclient

Section 7 Operation & Maintenance

Platform system management – Node	
Log	*Modular and privacy by design*
Monitoring	Quorum Blockchain Explorer and others.
Description	*Network status allows anyone to see the performance and number of nodes and where they are located.* https://github.com/jpmorganchase/quorum/blob/master/docs/Privacy/Tessera/Usage/Monitoring.md

Platform system management – Chain Network	
Permission Control	*Peer Permissioning, only known parties can join the network.*
Auditing	*Public or Private.*
Supervisory Support	*N/A*
Description	

Section 8 External Resource Management

Platform External Resource Management	
Interoperation solution	*Sharding: Raiden , state channel; IPFS; Swarm. IoT Gateways and Non-DLT system interoperation solution like AWS and Oraclize*
Description	*The schema is designed by the peer-to-peer approaching on the Smart Contracts and can contain different dependencies for their transactions which some are off-chain by obliterability.*

Section 9 Extensions

Platform Extensions – optional	
[the following list can be duplicated for multiple extensions]	
Name	*Contributor License Agreement (CLA) at info@goquorum.com*
Extension type[2]	
Extension mode[3]	
Solution	
Serve domain	
Description	*Quorum is built on open source.*

Platform Extensions – optional	
[the following list can be duplicated for multiple extensions]	
Name	*Anonymous Zether*
Extension type	*Internal*
Extension mode	*capability (vertical)*
Solution	*Zether is an anonymous private payment system extension based on zero-knowledge proof protocol.*
Serve domain	*Smart Contract Support*
Description	*After Zether is deployed at a network it allows users to transfer their EC20 balances to other Zether accounts in a private (amounts) and anonymous (identity) way. At this moment Zether is only enabled to Raft consensus mode.*

Standing from DLT system instance perspective, any extension inside the instance is marked as "internal", while any extension outside the instance is marked as "external"

[3] All extension instances are equal (with similar capability and functional features), targeting for the scalability of DLT instance, marked as "horizontal"; extensions with different functional features, targeting to enforce the capability of DLT instance, marked as vertical. Extension type and mode pair(s) is/are used to describe the extension as to the whole DLT system. E.g., sharding (internal – horizontal), lightening – BTC (external – vertical), Corda Contract (internal – vertical).