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).