


Ad Usque Fidelis Cabinet Louis Reynaud 	Reference: CLR.FE.004.E.2023 CLR-007
	Version: 1.8
EN 17640 SECURITY EVALUATION TARGET within the C.S.P.N certification scheme – Working document	Page 1 on 24

Document Status

Author	DEMASI Thibault
Current version	1.8
Date	16/11/2023

	Developers	Sponsor	Evaluator
Organization(s)	N.A.	CLR Labs as part of the CAMPUS Cyber work	CLR Labs

Copyrights

© Cabinet Louis Reynaud, 2023. This document is the property of the company Cabinet Louis Reynaud SASU. Copyright and international treaty provisions protect this document. No copies or partial productions are authorized without the written consent of the company Cabinet Louis Reynaud SASU.


Cabinet Louis Reynaud SASU - N° SIRET 83373449400010 - RCS, Marseille 833 734 494
3 rue plan cavaillon - 13420 Gémenos (France) – CLR Labs : 2 rue fougasse 13600 La Ciotat (France)
Rue de la science 14b - 1000 Brussels (Belgium) - Web : www.cabinet-louis-reynaud.fr

Restricted

Name	Organization
Anyone - Public	Public review

Contents

Introduction.....	4
1. Summary	4
1.1 Identification of the security target	4
1.2 Identification of the system to be evaluated	4
1.3 References	4
1.4 Definitions et abbreviations	5
2. Argument.....	7
2.1 General description of the system to be evaluated	7
2.2 Description of the use of the system to be evaluated	8
2.3 Description of the intended use environment	9
2.4 Description of dependencies.....	10
2.5 Description of typical users	11
2.6 Description of the scope of the evaluation	11
3. Description of the technical operating environment.....	12
3.1 Assets to protect	17
4. Environmental hypotheses.....	18
5. Threats description.....	18
6. Description of the security functions of the system to be evaluated	20
7. Threat Coverage	22

Ad Usque Fidelis Cabinet Louis Reynaud 	Reference: CLR.FE.004.E.2023 CLR-007
	Version: 1.8
EN 17640 SECURITY EVALUATION TARGET within the C.S.P.N certification scheme – Working document	Page 4 on 24

Introduction

The objective of this document is to verify if the EN 17640 evaluation methodology can be used for performing an Ethereum Nodes evaluation under the French C.S.P.N certification scheme.

This initiative has been initiated within the Crypto Asset Working Group of the French CAMPUS CYBER. CLR Labs is the editor of this evaluation Security Target and the Crypto Asset Working Group members have already provided their comments on this version of the document.

This document is put under public consultation in order to get some extra feedback from the Web 3.0 and Cybersecurity ecosystems. You can provide your comment using the commenting table associated to this document and send them to: info@cabinet-louis-reynaud.fr

1. Summary

1.1 Identification of the security target

Nodes are instantiations of Ethereum clients such as Geth, Erigon, Nethermind or Besu on servers or computers. Nodes can have several roles including maintaining a copy, updating and communicating to their peers the blockchain ledger. The nodes can also have the mission of validating transactions and executing smart contracts via the Ethereum Virtual Machine (EVM). The blockchain ledger is updated by the network of nodes via a consensus mechanism (PoS), in place since September 2022.

1.2 Identification of the system to be evaluated

Category	Identification
System trade name	Ethereum Nodes
Evaluated version number	X
System category	Blockchain, Ethereum, Crypto-assets, Smart Contract, Decentralization

1.3 References

Code	Reference	Name and Source
[1]	CSPN	First level security certification – A.N.S.S.I. https://cyber.gouv.fr/documents-applicables-la-certification-de-securite-de-premier-niveau-cspn
[2]	RGPD	General Data Protection Regulation Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of individuals with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/ 46/EC April 27, 2016

[3]	CLR.SEC.MS	CLR Labs Security Manual 28 Février 2020
[4]	DevP2P	Networking Layer : https://ethereum.org/fr/developers/docs/networking-layer
[5]	Merkle Patricia Trie	Merkle Trees : https://ethereum.org/en/whitepaper/#merkle-trees
[6]	Guide on how to join the Goerli/Prater merge testnet (Geth/Lighthouse)	Guide : https://github.com/eth-educators/ethstaker-guides/blob/main/merge-goerli-prater.md
[7]	EN 17640	EN 17640: 'Fixed-time cybersecurity evaluation methodology for ICT products' helps evaluate the cybersecurity of ICT products

1.4 Definitions et abbreviations

Name	Abbreviation	Definition
Agence Nationale de Sécurité des Systèmes d'Information	ANSSI	ANSSI has a mission to defend State information systems and is responsible for providing advice and support to administrations and operators of vital importance.
Cabinet Louis Reynaud Labs	CLR Labs	Security evaluation laboratory of Cabinet Louis Reynaud.
International Standard Organisation	ISO	International standards body.
General Data Protection Regulations	RGPD	[4]
Security Function Requirement	SFR	Security feature to provide a countermeasure to potential attacks.
Target Of Evaluation	TOE	Product to be evaluated by the laboratory within the scope defined with the customer.
Smart Contract		Programmable digital contract that automates the terms and conditions of an agreement between the parties, without the intervention of a trusted third party. They are executed in a blockchain, guaranteeing their immutability and transparency.

Blockchain		Decentralized and distributed database recording the history of all transactions carried out on its platform. The data blocks are cryptographically linked to each other to form an unbroken chain. The blocks are added to the main chain according to the consensus rules that the blockchain validators follow.
Nodes		A computer that is part of a decentralized network and manages transactions and maintains a copy of the blockchain. They can validate transactions, store data, execute smart contracts and participate in the process of maintaining the integrity of the blockchain.
Decentralized network		A computer system that operates without a single central authority controlling transactions and records. Data is managed by a network of nodes distributed in a decentralized manner.
Ethereum	Eth	Specific blockchain allowing Proof Of Stake (PoS), quasi-Turing complete, execution of smart contracts, etc. It uses its own crypto-assets (ETH) to power the network.
Virtual Machine	VM	Computer system simulating the environment of a physical machine. It provides abstraction of hardware resources such as memory, processor and I/O.
Proof Of Stake	PoS	It is a method by which a crypto asset blockchain aims to achieve distributed consensus.

<p style="text-align: center;">Ad Usque Fidelis Cabinet Louis Reynaud </p>	Reference: CLR.FE.004.E.2023 CLR-007
	Version: 1.8
EN 17640 SECURITY EVALUATION TARGET within the C.S.P.N certification scheme – Working document	Page 7 on 24

2. Argument

2.1 General description of the system to be evaluated

There are 3 types of blockchains within the framework of Ethereum: public blockchains, private blockchains and intermediate (or so-called semi-public) blockchains.


1. **Public blockchain:** A public blockchain is an open and decentralized network, accessible to everyone. **The present security target is focused on a public blockchain.** Participants can join the network, propose transactions, deploy smart contracts and participate in the consensus process.
2. **Private Blockchain:** A private blockchain is a network controlled and managed by an entity or organization. Network access and read, write and transaction validation rights are limited to authorized participants. Quorum is an example of a private Ethereum blockchain.
3. **Intermediate blockchain (consortium blockchain):** A blockchain sits between public and private blockchains in terms of control and access. In this type of blockchain, the network is managed by a set (consortium) of organizations rather than a single entity. Consortium participants collectively control and manage the network, determine access rules and make decisions regarding its operation.

The target is dedicated to Ethereum nodes and the different tasks dedicated to them:

- **Blockchain storage:** Ethereum nodes maintain a full or partial copy of the Ethereum blockchain, which is a distributed database containing the complete history of all transactions and state changes on the network.
- **Synchronization and propagation:** Ethereum nodes synchronize with each other by sharing and verifying block and transaction information. They contribute to the propagation of data on the network.
- **Transaction Validation:** Ethereum nodes validate transactions by verifying their signatures, balances, and state data to ensure transactions are legitimate.
- **Execution of smart contracts (outside the TOE):** nodes execute smart contracts deployed on the network using the EVM in a decentralized manner, verifying and applying the resulting state changes.
- **Consensus Participation:** Ethereum nodes participate in the consensus mechanism to validate and add new blocks to the blockchain. Since September 2022, with the transition from Ethereum 1.0 to Ethereum 2.0, the network has moved from a Proof of Work (PoW) mechanism to a Proof of Stake (PoS) mechanism.

Ethereum nodes can be full nodes, light nodes, or archive nodes:

- **Full Nodes:** Nodes store the entire blockchain, including all blocks, transactions, and state data. They can validate and transmit transactions, execute smart contracts and fully participate in the consensus process.
- **Light Nodes:** Light nodes store only partial information about the blockchain, such as block headers, and rely on full nodes for state data and transactions. They are designed to work on low-capacity devices.

<p style="text-align: center;">Ad Usque Fidelis Cabinet Louis Reynaud </p>	Reference: CLR.FE.004.E.2023 CLR-007
	Version: 1.8
<p style="text-align: center;">EN 17640 SECURITY EVALUATION TARGET within the C.S.P.N certification scheme – Working document</p>	<p style="text-align: center;">Page 8 on 24</p>

- Archive Nodes: Archive nodes are full nodes that also maintain the complete history of state changes for each block.

2.2 Description of the use of the system to be evaluated

Users can create nodes and deploy smart contracts using the platform and development tools, such as Geth, Nethermind, Besu, etc.

Here Geth (go-ethereum) will be used, for several reasons:

1. Interoperability: Geth is the official Ethereum client node and is widely used, ensuring interoperability with other applications.
2. Complexity: Geth is relatively simple to use and configure
3. Support: Geth has great support from the Ethereum community
4. Features: Geth offers a number of additional features (support for decentralized applications, fast synchronization, etc.)

The creation and deployment of a node will be the main functionalities of the system that will be evaluated. A full node will be deployed using the Geth client, allowing all available functionality of the node.

For this, several steps are necessary for proper operation:

- Installation of Geth and Lighthouse (consensus client allowing the deployment of the PoS mechanism). It is recommended to deploy nodes on Linux.
- Creation of a jwt token to authenticate the connection.
- Configuring the Geth service.
- Configuring the Lighthouse service.
- Add Ethereum funds to a user account. For a user to become a validator, they must have at least 32 eth. For personal use, it is possible to use faucets to generate Ethereum on derived networks (Sepolia, etc.).
- Add a validator.

For more information and details, please refer to reference [6].

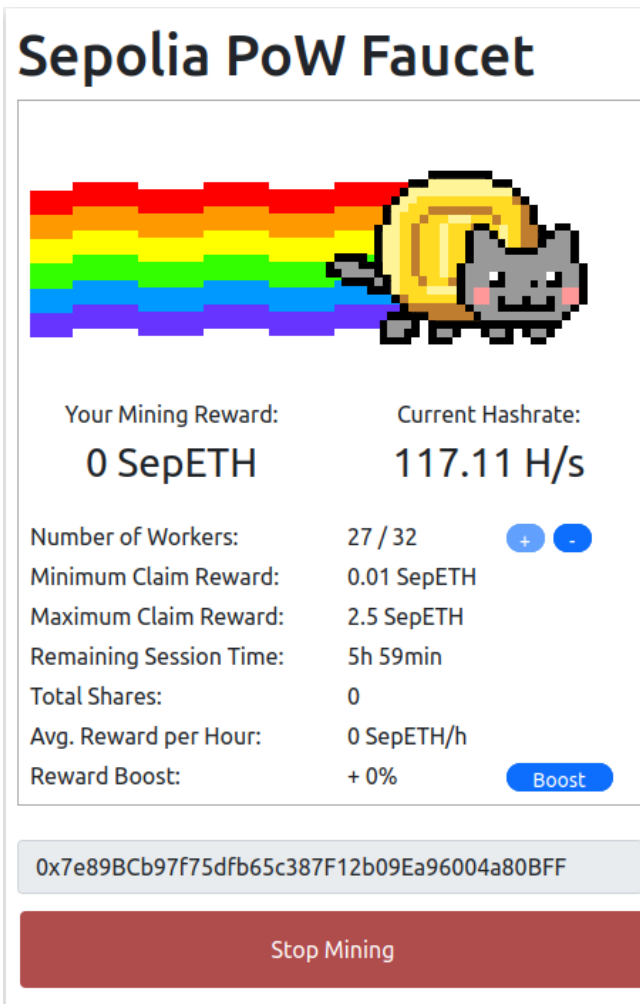


Figure 1: Example of faucet for generating eth tests

2.3 Description of the intended use environment

Ethereum nodes are used to execute and validate transactions in the Ethereum node. They are also used to deploy and execute smart contracts via the EVM, allowing the implementation of decentralized applications.

Ethereum nodes can be used to store and access the Ethereum database, allowing some transparency and immutability of the information stored in the blockchain to be maintained.

Here is the flow diagram in a situation where a user wants to check their Eth balance:

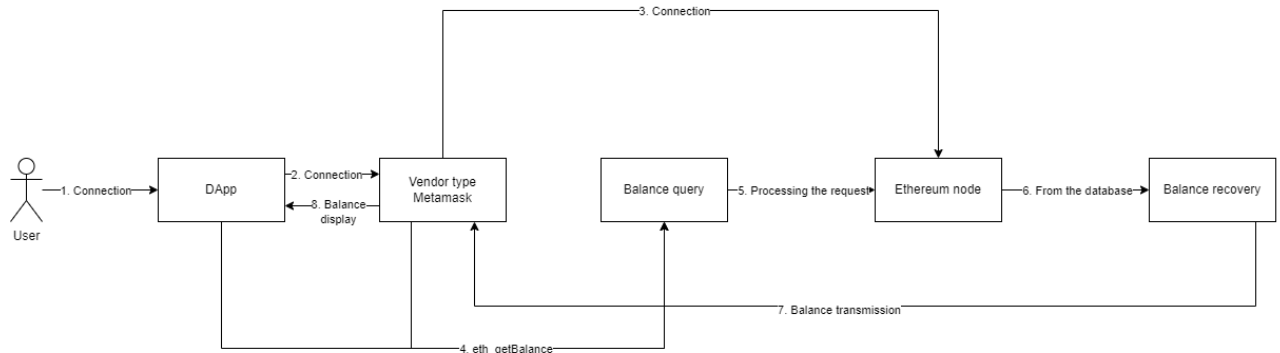


Figure 1 Flow when a user wants to check their balance.

Here is another flow diagram where a user makes a transaction and then executes a smart contract:

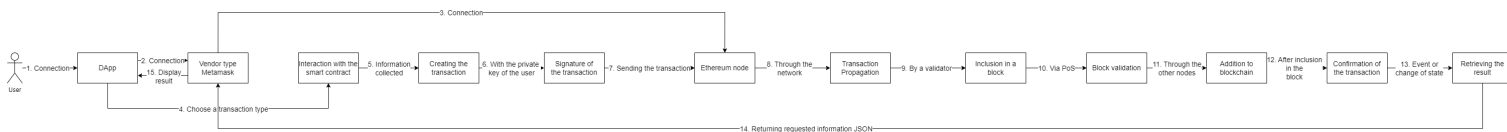


Figure 3 : Description of the evaluation scope

2.4 Description of dependencies

To function correctly, the system to be evaluated is dependent on all the elements of its environment, namely:

- **Ethereum client:** It is necessary to have an Ethereum client to interact with the node (Geth for example).
- **Consensus client:** It is necessary to have a consensus client (here Lighthouse) to allow a user to become a validator.
- **Platform:** Geth and Lighthouse are compatible with several OS, such as Windows, Linux or Mac. It is recommended to use Linux.
- **Programming language:** The keystores, storing the encrypted private key, are encoded in JSON format.
- **Blockchain:** Ethereum is a blockchain and its presence is necessary for the creation and execution of smart contracts, for the validation of smart contracts, etc.
- **Private/public key pair:** A user has an asymmetric key pair. The public key allows the user to generate the Ethereum address for the account, used to receive funds. The private key allows transactions to be signed and access to funds associated with the application. The algorithm used for key generation is Elliptic curve digital signature algorithm on the secp256k1 curve. The private key is stored in a keystore file.



2.5 Description of typical users

Users likely to interact with the system to be evaluated are:

- Developer of smart contracts
- Users of decentralized development platforms
- dApp developers
- Ethereum blockchain user
- Investors in Ethereum-based projects.
- Network Architect
- Designer of equivalent EVM solutions (Layer 2, Rollups)

2.6 Description of the scope of the evaluation

The evaluation focuses on the Ethereum node, its operation as well as certain implementations that it defines, such as data storage, the keystore and the software allowing consensus execution.

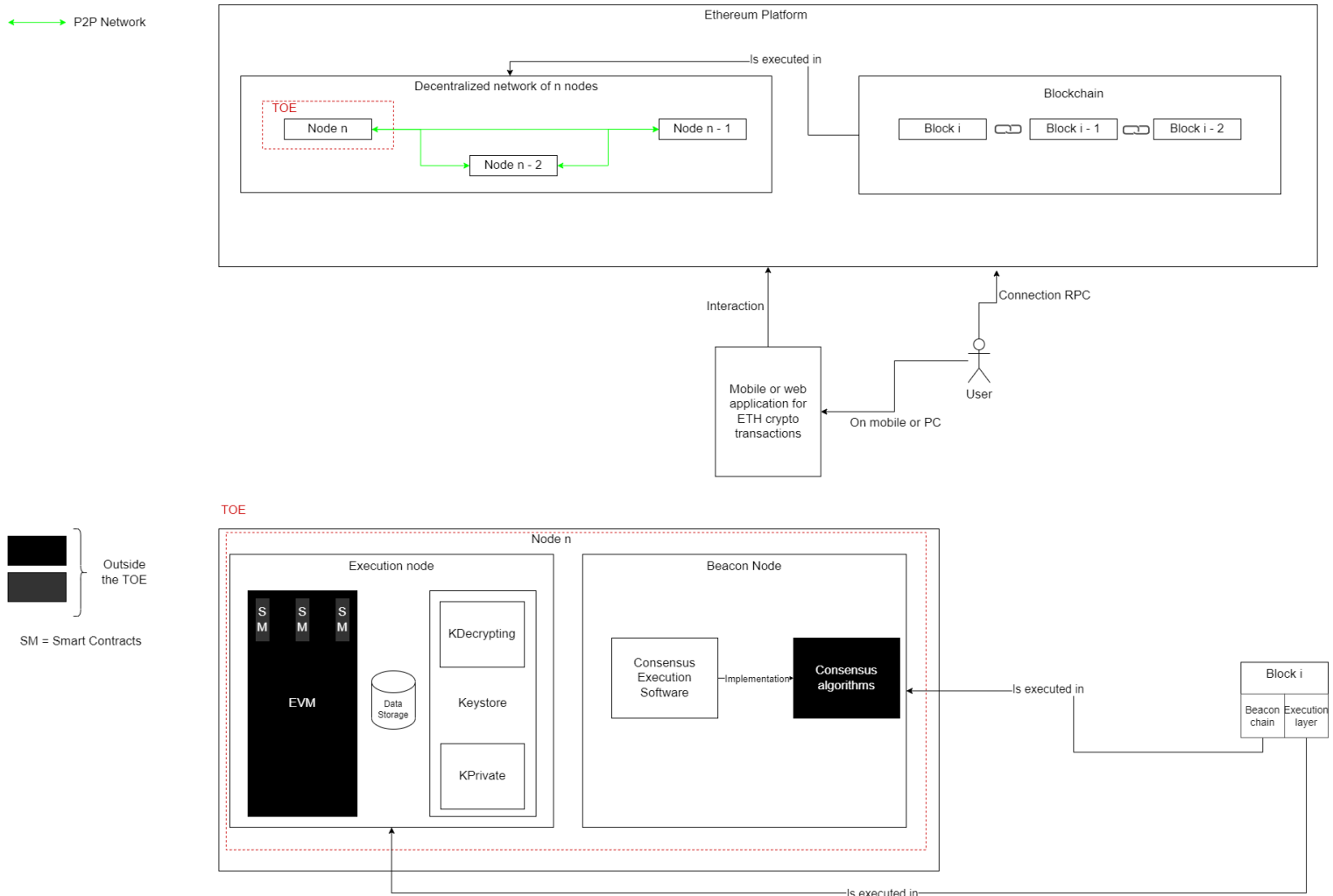



Figure 4: Description of the evaluation scope

<p style="text-align: center;">Ad Usque Fidelis Cabinet Louis Reynaud </p>	Reference: CLR.FE.004.E.2023 CLR-007
	Version: 1.8
<p style="text-align: center;">EN 17640 SECURITY EVALUATION TARGET within the C.S.P.N certification scheme – Working document</p>	Page 12 on 24

3. Description of the technical operating environment

Communication between nodes and with the EVM:

Communication between the nodes and the EVM is done via a protocol called DevP2P. It allows nodes to connect, share information, and synchronize network status. It uses UDP and also a mechanism called Distributed Hash Table to allow nodes to locate and connect to each other.

Specifically, nodes will use the sub-protocol called the Ethereum Wire Protocol to exchange specific information, such as transactions, blocks, and network state information via “messages.” Here are some main messages:

1. Status: This message is exchanged during the initial connection between nodes and to share information about their protocol version, the blockchain they follow, and their sync status.
2. NewBlockHashes: This message is used to inform other nodes about the discovery of new blocks.
3. Transactions: This message is used to transmit transactions between nodes.
4. GetBlockHeaders, BlockHeaders: These messages are used to request and share block headers during blockchain synchronization.
5. GetBlockBodies, BlockHeaders: These messages are used to request and share block bodies during blockchain synchronization.

For more information on the protocols, please follow the reference heading [4].

Data storage and organization:

Data is stored in nodes. Each node contains a complete copy of the blockchain, containing all transactions, account addresses, balances and deployed smart contracts. The EVM is only responsible for executing the associated code and updating the network state based on the results of that execution. Nodes are responsible for managing data storage and synchronization. There are different types of nodes: full nodes, lightweight nodes, and archive nodes.

- Full nodes store the entire blockchain.
- Lightweight nodes only store block headers
- Archive nodes maintain the complete state history for each block.

The database is organized using a structure called Merkle Patricia Trie, a variation of the Merkle Tree. This structure makes it possible to search for, insert and delete elements and to prove, using cryptographic calculations, the existence or non-existence of the elements. Trie items are stored using a key-value database, similar to LevelDB or RocksDB. Storage in the nodes is done in several parts:

1. Blockchain: blocks are organized in a linear manner and each block is linked to its predecessor by a hash. Each block contains a header and a list of transactions. Block headers are stored separately from block bodies.



2. Overall state: this is the current state of the network, which includes account balances, deployed smart contracts and their internal data. The global state is organized so that each node in the Trie represents a state element. Each element is indexed by a key derived from its address or identifier.
3. Transactions: Transactions are organized in a Trie within the block itself. Each block contains a field called TransactionRoot, which is the root hash of the Transaction Trie. This allows you to check whether a specific transaction is part of a given block using a Merkle proof (see SFR-13).
4. Receipts: Transaction receipts are also organized in a Trie. Receipts are records of the results of transaction execution such as event logs and success or failure status. They are indexed in the same way as transactions.

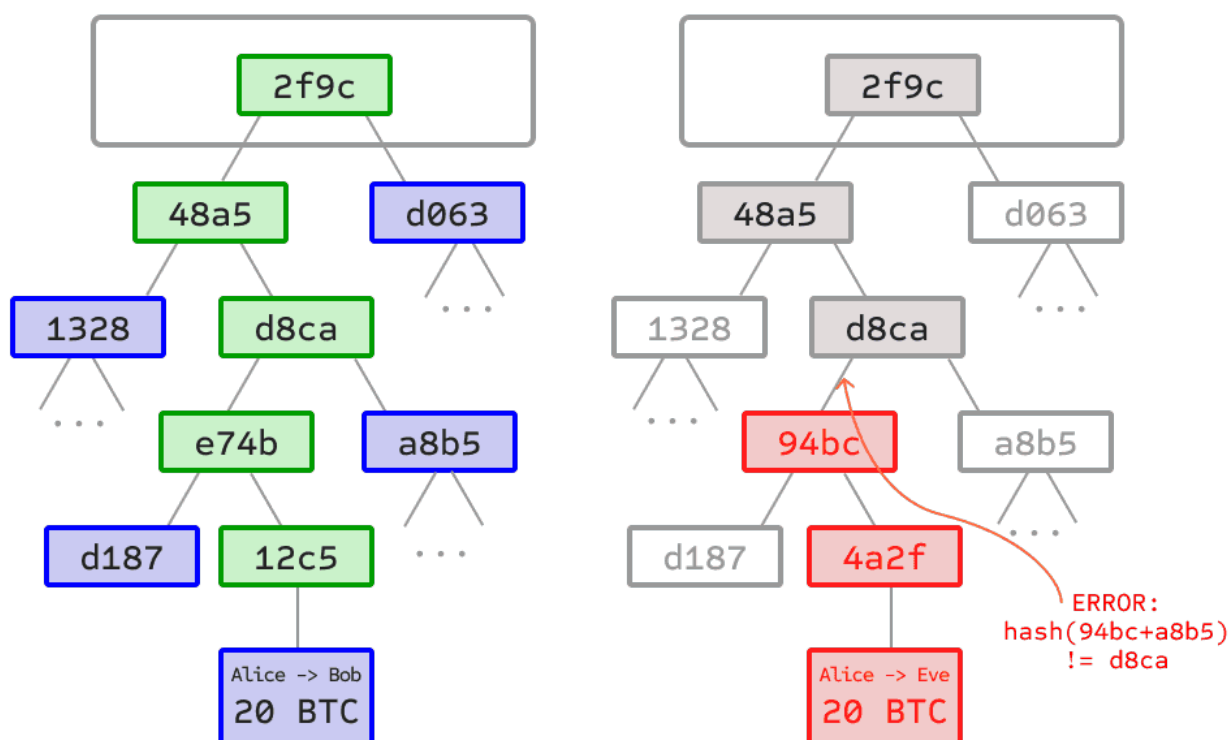


Figure 4 : How Merkle Patricia Tree works with example transactions..

For more information, see the heading [5] of the references.

Confirmation process for creating a new block

The process of confirming and sharing information when a block is created takes place in several stages:

1. Creation of a block: a validator is selected to propose a block (Proof-Of-Stake). They then gather a set of pending transactions, execute them, create a new state of the network and construct a block with these transactions, the result state and additional information (hash of the previous block, root hash, etc.)

2. Block Broadcast: The validator then broadcasts the new block across the network, sending it to neighboring nodes. Neighboring nodes validate the block, and, if it is valid, add it to their own copy of the blockchain and transmit the block to their neighbors. This propagation process continues until the majority, or even all, of the nodes have received and validated the block.
3. Block Confirmation: Once a block is added to the blockchain, it is considered confirmed. To reduce the risks of blockchain reorganization, a number of additional confirmations must be made before considering a transaction as definitively confirmed.

Information contained in a keystore

```
["address": "7e89bcb97f75dfb65c387f12b09ea96004a80bff", "crypto": {"cipher": "aes-128-ctr",
"cyphertext": "430511ba31807c1799850cfeea"}, "kdf": "scrypt", "kdfparams": {"dklen": 32, "n": 262144, "p": 1, "r": 8, "salt":
430511ba31807c1799850cfeea"}, "id": "430511ba31807c1799850cfeea", "version": 3}
```

Figure 5 part 1 : see Figure 5 part 2

```
er", "cyphertext": "430511ba31807c1799850cfeea"}, "kdf": "scrypt", "kdfparams": {"dklen": 32, "n": 262144, "p": 1, "r": 8, "salt":
430511ba31807c1799850cfeea"}, "id": "430511ba31807c1799850cfeea", "mac": "430511ba31807c1799850cfeea", "version": 3}
```

Figure 5 part 2 : Contents of the keystore file for a user (only readable by a root user)

For each Ethereum account, a keystore file is associated with it. Below is the breakdown of the file:

- “address”: the Ethereum address associated with the private key.
- “crypto”: the encryption information used to protect the private key.
 - “cipher”: the encryption algorithm used, here “aes-128-ctr” (AES with a counter mode and a 128-bit key).
 - “cyphertext”: the private key encrypted as a hexadecimal string.
 - “cypherparams”: the encryption parameters, in particular the initialization of the “iv” vector.
 - “kdf”: the key derivation algorithm used to encrypt the private key, here “scrypt”.
 - “kdfparams”: parameters of the key derivation algorithm, such as derived key length (“dklens”), memory cost (“n”), parallelism cost (“p”), CPU cost (“r”) and salt (“salt”).
 - “mac”: an integrity check value to verify that the passphrase used to decrypt the private key is correct.
- “id”: A unique identifier for the keystore file
- “version”: the version of the keystore format, here 3.

Steps for creating the keystore file

1. Private key generation: The private key is generated locally on the user's device.
2. Generation of the public key and Ethereum address: From the private key, a public key is generated using elliptic curve cryptography on the secp256k1 curve. The Ethereum address is then derived from the public key by taking the last 20 bytes of the keccak-256 (sha-3) hash of the public key.

3. Private key encryption: The private key is encrypted with the AES-128-CTR algorithm with a key derived from the passphrase. The passphrase is transformed into an encryption key using a key derivation function (aka KDF) which is scrypt, with specific parameters.
4. Creating the keystore file: Information to decrypt the private key, such as the Ethereum address, encryption parameters, KDF parameters, and an integrity check (MAC) value is stored in JSON format.

Steps for using the keystore

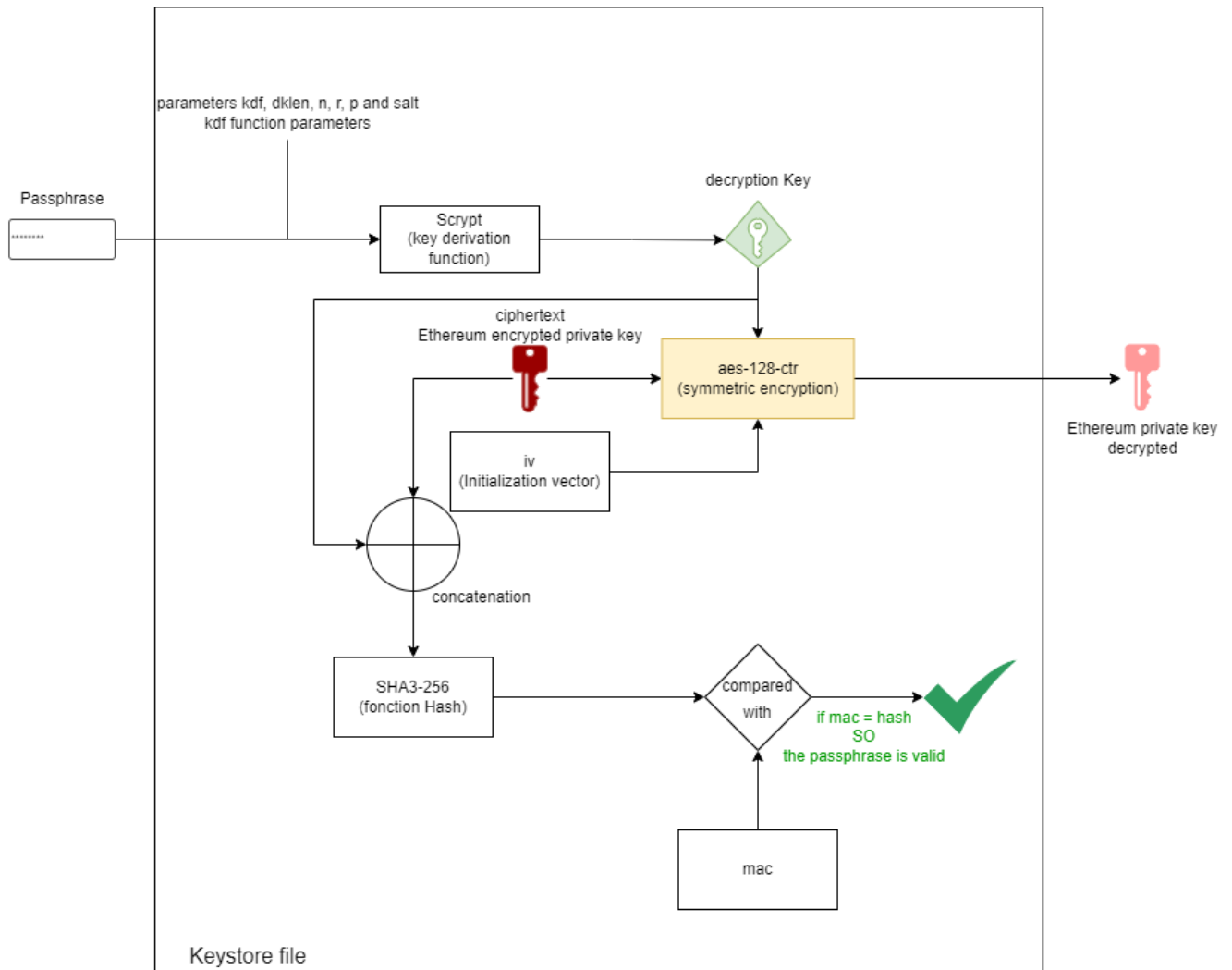



Figure 6: Keystore usage flow

1. Use of the passphrase: The user enters their passphrase to carry out a transaction or to manage their Ethereum account.

<p style="text-align: center;">Ad Usque Fidelis Cabinet Louis Reynaud </p>	Reference: CLR.FE.004.E.2023 CLR-007
	Version: 1.8
<p style="text-align: center;">EN 17640 SECURITY EVALUATION TARGET within the C.S.P.N certification scheme – Working document</p>	<p style="text-align: center;">Page 16 on 24</p>

2. Extracting keystore information: Necessary information is extracted from the keystore, including encryption settings, key derivation algorithm (KDF) settings, and integrity check value.
3. Deriving the encryption key: The passphrase and KDF parameters extracted from the keystore are used to derive the encryption key. This is done by applying the KDF (scrypt) algorithm with the specified parameters.
4. Verifying keystore integrity: Before decrypting the private key, the integrity of the keystore is verified using the MAC value. The MAC is calculated by concatenating the derived key and the encryption result of the private key, then applying a hash algorithm (Keccak-256). If the calculated MAC does not match the MAC stored in the keystore, it means the passphrase is incorrect or the keystore is corrupt.
5. Decryption of the private key: If the integrity check passes, the derived encryption key and encryption parameters are used to decrypt the private key.
6. Signing Transactions: Once the private key is decrypted, transactions are signed using elliptic curve cryptography (ECC) on the secp256k1 curve. The signature guarantees that the user is the owner of the Ethereum address associated with the private key.
7. Sending signed transactions: Signed transactions are then sent to the Ethereum network for processing and inclusion in the blockchain.

Transition from Proof of Work to Proof of Stake

Since September 2022, Ethereum has moved from a Proof Of Work system to a Proof Of Stake system, offering an alternative consensus mechanism. In the PoS system, the node with the most Ether staked is authorized to generate blocks unlike PoW nodes where it is the nodes with the greatest computing power that do so.

The advantages of the PoS system over the PoW system are as follows:


- Reduction of significant investments in equipment and energy, more respectful of the environment.
- Technological and architectural improvements.

PoS introduces the concept of shard chain, smaller chains that manage part of the network data:

- Ethereum plans to have 64 of these chains.
- Validators only need to process data on the chain they are validating, reducing hardware requirements and improving security through greater decentralization.

To become a validator, users stake 32 eths. The validators propose and validate the blocks, those who are chosen to propose blocks are called the proposers and those who confirm the attestations. Validators risk losing part or all their stake if they disconnect, fail to validate, or attempt to manipulate the system.

A new component, the beacon chain, ensures coordination between shard chains and maintains network synchronization. The beacon chain also receives information about the state of the shard chains, which allows the entire network to maintain its state. It records attestations, which are confirmations that a block appears valid, rather than recording the transactions themselves.

 Ad Usque Fidelis Cabinet Louis Reynaud	Reference: CLR.FE.004.E.2023 CLR-007
	Version: 1.8
EN 17640 SECURITY EVALUATION TARGET within the C.S.P.N certification scheme – Working document	Page 17 on 24

A minimum of 128 validators are required to attest to each block, thus forming a committee. The committee has a deadline or window to propose and validate a block. After each series of 32 slots (epoch), the committees are randomized to maintain network security. Validators are rewarded for proposing and attesting blocks, but they can lose their stake if they attest a malicious or invalid block.

When a shard block receives enough attestations, a crosslink is formed, confirming the block's inclusion in the beacon chain. Cross-links allow the beacon chain to follow the head of the shard chain, allowing all shards to stay in sync.


Beacon node and Execution engine

2 main parts structure the Ethereum network:

1. Beacon node:
The Beacon node is responsible for managing the Ethereum PoS consensus. It takes care of the coordination and management of PoS validators, as well as the creation and proposal of new blocks for the blockchain. The beacon node follows the Beacon Chain, which is a separate blockchain dedicated to PoS consensus. Validators are selected to propose blocks and attest to proposed blocks based on their stake and other parameters. Beacon nodes also manage rewards and penalties for validators based on their performance.
2. Execution engine:
The execution engine is responsible for executing transactions, smart contracts, and processing network state changes. In other words, it manages all operations related to transactions and smart contracts on the Ethereum blockchain. The Execution Engine uses the EVM to perform these operations. It works with shard chains designed to increase network capacity and performance.

3.1 Assets to protect

Sensitive assets to protect	Assets attached to the application	Location	Confidentiality	Integrity	Availability	Asset Reference
Funds	Ether funds and assets	Node	Yes	Yes	Yes	A1
Smart contract	Transactions	EVM	No	Yes	Yes	A2
User public key	“Public” user information	Node	No	Yes	Yes	A3
Transactions	Eth, crypto assets	Nodes, EVM	No	Yes	Yes	A4

 Ad Usque Fidelis Cabinet Louis Reynaud	Reference: CLR.FE.004.E.2023 CLR-007
	Version: 1.8
EN 17640 SECURITY EVALUATION TARGET within the C.S.P.N certification scheme – Working document	Page 18 on 24

Source code	Node, Ethereum	Github, Ethereum Platform	No	Yes	No	A5
Node private key	Ethereum Node	Keystore	Yes	Yes	Yes	A6
Node public key	Ethereum Node	Node, Ethereum Platform	No	Yes	Yes	A7

4. Environmental hypotheses

Hypotheses	Description
H1	The identity of the user of a dApp is verified beforehand by a third party.
H2	Nodes are deployed in a decentralized network.
H3	Consensus algorithms are considered secure.
H4	The EVM is considered a trusted environment.
H5	The cryptographic algorithms used, if they comply with ANSSI recommendations, are considered secure.
H6	The evaluation is focused within the framework of the public Ethereum blockchain.
H7	The identity of the user is not entered in the nodes and will therefore not need to be protected.

5. Threats description

Threats	Description	Achievement difficulty levels to achieve the threat
T1	An attacker carries out a Denial of Service (DoS) attack with the aim of bringing down nodes and making them unavailable. Objectives: network slowness, data loss, service interruption, loss of trust.	Basic
T2	An attacker uses the node's private key to validate the transactions he wants. Objective: Theft of funds	Substantial
T3	An attacker carries out a Race Condition attack with the aim of accessing a resource accessible to several threads or processes at the same time whose final behavior depends on the order in which these threads or processes access the resource and whose instructions are not correctly synchronized.	Substantial

<p>Ad Usque Fidelis Cabinet Louis Reynaud </p>	Reference: CLR.FE.004.E.2023 CLR-007
	Version: 1.8
EN 17640 SECURITY EVALUATION TARGET within the C.S.P.N certification scheme – Working document	Page 19 on 24

	Objectives: Competitive advantage, control of transactions, modification of data, interference with the normal functioning of the blockchain.	
T4	An attacker carries out a front-running attack by manipulating the order of transactions to prioritize a specific transaction over other transactions initially scheduled to be processed first. By monitoring pending transactions, the attacker identifies an opportunity to buy or sell assets at a price lower than their true value. Then, the attacker creates and broadcasts a competing transaction with higher gas fees, tricking validators into including this transaction in the next block before the one originally intended. Objectives: illicit access to assets, price manipulation, theft of sensitive information, loss of trust.	Substantial
T5	An attacker usurps a user's account to redirect funds to divert attention in the event of an investigation (ransomware, etc.). Objective: Diversion of attention	Basic
T6	Attackers can corrupt data stored in Ethereum nodes. Objectives: Modification of the logic of smart contracts, modification of the state of smart contracts, alteration of blockchain data, interference with the normal operation of the blockchain, theft of funds	Substantial
T7	Attackers can compromise data integrity by intercepting communications between Ethereum nodes. Objectives: commercial sabotage, infiltration into other systems, preparation for subsequent attacks.	Substantial
T8	An attacker conducts a 51% attack by taking control of the majority of the blockchain's computing power, allowing the attacker to control transaction validation and create double spending. Objectives: Double spending, reversal of transactions, blocking of transactions, reduction of market confidence.	Substantial
T9	An attacker who has retrieved a user's keystore file can bruteforce it to recover the victim's private key. Objective: Theft of funds	Substantial
T10	An attacker can use a malicious node and make fraudulent transactions or fund their account. Objective: Theft of funds, competitive advantage, illicit access to assets	Substantial

Table 1: Description of threats

6. Description of the security functions of the system to be evaluated

SFR	TOE subsystem providing SFR	Description
SFR – 1	Sandbox	Virtual environment in which smart contracts execute to limit the potential effects of programming errors.
SFR – 2	Gas limitation	Limited quantity of gas allocated to a contract for its execution
SFR – 3	Checking the stack size	Check to ensure that the stack size does not exceed the defined limit.
SFR – 4	Checking data types	Verification ensuring that the input data is of the correct type and complies with the contract.
SFR – 5	Verification of the subdivision	Check on mathematical operations preventing the generation of negative values.
SFR – 6	Verification of over-allocation	Check on mathematical operations preventing the generation of values greater than a predefined maximum value.
SFR - 7	Secure storage	Storing data across multiple nodes, ensuring this information is secure in the event of an outage or failure.
SFR - 8	Integrity and authenticity of nodes	<ul style="list-style-type: none"> • Consensus Proof-of-Stake (PoS): Validators are chosen to propose and validate blocks based on their stake. Validators are incentivized to act honestly, earning rewards for correctly validating blocks and losing slashes for malicious behavior. • Cryptographic chaining of blocks: blocks are linked together via hashes. Each block contains the hash of the previous block, creating the blockchain chain going back to the genesis block. Modifying an earlier block would mean calculating the hashes of all subsequent blocks. • Merkle Trees: Transactions, receipts, and overall state are organized using a Merkle Patricia Tree structure. The hashes of these Tries are included in the block headers. Nodes can then verify the existence or non-existence of transactions, receipts or state elements using Merkle proofs (see SFR – 13). • Transaction signing: Transactions are signed by the sender using a private key. The corresponding public key is used to verify the authenticity of the signature. • Node validation: Full nodes validate all transactions and blocks before adding them to their local copy of the blockchain. Light nodes can also verify transactions using Merkle proofs provided by full nodes.
SFR - 9	Merkle's proofs	Cryptographic mechanism for verifying that a specific item is part of a data set. Merkle proofs are developed using Merkle trees. These are constructed by hashing the elements of the entire data set and grouping the hashes into pairs. The hashes of these peers are then

		combined and hashed again, creating a new set of hashes. This process is repeated until only one hash remains, called the root hash. To verify that a specific element is part of the dataset, a Merkle proof is constructed by providing the hashes of sibling nodes (sibling) along the element's path to the root hash. By recalculating the hashes following the path and comparing the result with the root hash, we can confirm whether the element is part of the dataset.
SFR - 10	Transaction validation	Transactions validated using Proof-of-Stake (PoS). PoS is a consensus method used to validate transactions and maintain network integrity. This method does not rely on processing power, but on the active participation of users as transaction validators.
SFR - 11	Private key storage	Storage of the private key is done in a keystore file on the computer hosting the node. This file is encrypted with a password to ensure the security of the private key. When a node starts, it decrypts the private key using the provided password. The keystore uses specific settings to strengthen security against bruteforce.


Table 2: Security function (SFR)

7. Threat Coverage

		Sandbox	Gas limitation	Checking the stack size	Checking data types	Verification of the subdivision	Verification of over-allocation	Secure storage	Integrity and authenticity of nodes	Merkel's evidence	Transaction validation	Private key storage
		SFR 1	SFR 2	SFR 3	SFR 4	SFR 5	SFR 6	SFR 7	SFR 8	SFR 9	SFR 10	SFR 11
Threats	Assets concerned											
T1	A4			*	*	*	*					
T2	A1, A4, A6, A7											*
T3	A4			*								
T4	A4		*								*	
T5	A1, A3, A4									*		
T6	A1, A2, A4	*										
T7	A1, A2, A4								*			
T8	A4								*	*		
T9	A1, A5											*
T10	A1, A4, A6, A7								*		*	

TBble 3: Threat coverage summary

END OF DOCUMENT

Ad Usque Fidelis Cabinet Louis Reynaud 	Reference: CLR.FE.004.E.2023 CLR-007
	Version: 1.8
EN 17640 SECURITY EVALUATION TARGET within the C.S.P.N certification scheme – Working document	Page 23 on 24


Form reference: CLR.FE.004

Version: 1.8

Written by:

Function	Name	Date (jj/m/aaaa)	Signature
Pentester mobile et IT	DEMASI Thibault	10/02/2023	

Approved by:

Function	Nom	Date (jj/m/aaaa)	Signature
Directeur Laboratoire	MOUILLE Stéfane	16/11/2023	

History of the form:

Version	Redactor	Date (jj/m/aaaa)	Modification
0.1	DEMASI Thibault	16/02/2023	Creation
1.0	DEMASI Thibault	24/02/2023	Focus on nodes and not EVM and Smart contracts
1.1	DEMASI Thibault	13/03/2023	Updated threat coverage
1.2	DEMASI Thibault	24/03/2023	Insertion of the keystore in the technical environment

**EN 17640 SECURITY EVALUATION TARGET within the
C.S.P.N certification scheme – Working document**

1.3	DEMASI Thibault	03/04/2023	TOE update and various corrections
1.4	DEMASI Thibault	30/05/2023	Modification of the TOE, addition of information on the PoS and the beacon Chain and various modifications.
1.5	DEMASI Thibault	08/06/2023	Miscellaneous changes
1.6	DEMASI Thibault	04/07/2023	Correction on threat coverage
1.7	DEMASI Thibault	30/08/2023	Integration of some comments
1.8	MOUILLE Stefane	16/11/2023	Translation in English language

FIN DE DOCUMENT