Advanced Cyber-Threat Intelligence, Detection, and Mitigation Platform for a Trusted Internet of Things
Grant Agreement: 786698

D7.4 – CYBER-TRUST blockchain security analysis (I)

Work Package 7: Distributed ledger technology for enhanced accountability

Document Dissemination Level

P	Public	☒
CO	Confidential, only for members of the Consortium (including the Commission Services)	☐

Document Due Date: 30/04/2020
Document Submission Date: 30/04/2020

Co-funded by the Horizon 2020 Framework Programme of the European Union

Document Information

Deliverable number:	D7.4
Deliverable title:	CYBER-TRUST blockchain security analysis (I)
Deliverable version:	1.0
Work Package number:	WP7
Work Package title:	Distributed ledger technology for enhanced accountability
Due Date of delivery:	30/04/2020
Actual date of delivery:	30/04/2020
Dissemination level:	Public
Editor(s):	Nicholas Kolokotronis (UOP)
Contributor(s):	Nicholas Kolokotronis, Konstantinos Limniotis, Sotirios Brotsis (UOP) Clément Pavué (SCORECHAIN) Gueltoum Bendiab, Stavros Shiaeles (UOPHEC)
Reviewer(s):	Michael Skitsas (ADITESS) Simone Naldini (MATHEMA)
Project name:	Advanced Cyber-Threat Intelligence, Detection, and Mitigation Platform for a Trusted Internet of Things
Project Acronym	CYBER-TRUST
Project starting date:	01/05/2018
Project duration:	36 months
Rights:	Cyber-Trust Consortium

Version History

Version	Date	Beneficiary	Description
0.1	14/02/2020	UOP	Table of contents proposed
0.2	21/02/2020	UOP, ALL	Table of contents and work breakdown
0.4	06/03/2020	UOP, CSCAN	Added section 5, 6
0.5	27/03/2020	UOP	Added sections 3, 4
0.6	17/04/2020	UOP, SCORECHAIN	Added sections 1, 2, 7; revised section 3
0.8	21/04/2020	UOP	Deliverable submitted for review
0.9	24/04/2020	ADITESS, MATHEMA	Review comments received
1.0	28/04/2020	UOP	Final version ready for submission

Acronyms

ACRONYM	EXPLANATION
A	Actor
API	Application Programming Interface
ASF	Apache Software Foundation
BCT	Block Confirmation Time
BFT	Byzantine Fault Tolerant
BT	Blockchain Technology
BWH	Block Withholding
CA	Certificate Authority
CAP	Consistency Availability and Partition tolerance
CFT	Crash Fault Tolerant
CT	Cyber-Trust
D	Deliverable
DDoS	Distributed Denial of Service
DLT	Distributed Ledger Technology
DSL	Domain Specific Language
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
ESCC	Endorsement System ChainCode
EVM	Ethereum Virtual Machine
FAW	Fork After Withholding
FPoW	Full Proof of Work
FT	Fault Tolerance
IoT	Internet of Things
LCR	Longest Chain Rule
LVR	Leader Validation Replica
M	Month
MDP	Markov Decision Process
MitM	Man in the Middle
OS	Ordering Service
OSN	Ordering Service Node
PBFT	Practical Byzantine Fault Tolerant
PDC	Private Data Collection

PoW	Proof-of-Work
Pub/sub	Publish/Subscribe
RPC	Remote Procedure Calls
SC	Smart contracts
SMR	State Machine Replication
SPoF	Single Point of Failure
T	Task
TPS	Transactions Per Second
TTP	Trusted Third Party
VR	Validation Replica
VSCC	Validation System ChainCode
VSR	View-Stamped Replication

Executive summary

Cyber-Trust project aims at developing an innovative platform tackling the grand challenges in securing the IoT ecosystem. To this end, a *distributed ledger technology* (DLT) is being appropriately used to ensure the robustness of threat detection and mitigation methods and provide a tamper-resistant environment that is suitable for the forensic evidence collected (from the devices and the network) as regards to alleged cyber-criminal behavior. This report aims to present and analyze the security of Cyber-Trust's DLT solution. More precisely, important security issues that arise in the development and operation of Hyperledger Fabric DLT are being considered, driving the implementation/configuration approaches that need to be adopted. The security threats covered by the present deliverable are classified into the following areas covering pretty much all the aspects of any DLT solution: (a) consensus security; (b) smart contract security; (c) software security; and (d) network security. In addition, to the above, privacy issues are also being considered under the same context. In each area, a number of alternative threat mitigation options have been identified, which are classified into those that have already (by design, or during the implementation) been implemented by the project, those that will be used (e.g. the choice of the consensus, or the use of IDS), and those that seem to be quite promising and could be further explored in future research efforts.

Table of Contents

Table of Figures

Table of Tables

1. Introduction

The Cyber–Trust project aims to develop an innovative cyber–threat intelligence gathering, detection, and mitigation platform to tackle the grand challenges towards securing the ecosystem of IoT devices. To this end, a DLT is being appropriately used by the Cyber-Trust system, in order to play a decisive role in ensuring robustness to the detection and mitigation methods and provide a high-integrity and tamper-resistant environment that is suitable for the forensic evidence collected (from both the devices and the network in case of alleged cyber-criminal activities) as regards to alleged cyber-criminal behavior. The DLT also provides transparency in the context of the Cyber-Trust system, by allowing check the history of communications between organizations/LEAs that interact with the Cyber-Trust system (i.e. for establishing a chain of custody, so as to verify all previous communications).

As stated and justified in D7.2 [68], the choice of the relevant DLT technology with respect to the Cyber-Trust system is the Hyperledger Fabric.

1.1 Purpose of the document

This report aims to present in detail the security framework regarding the DLT in the Cyber-Trust system. More precisely, the present report considers important security issues that arise in developing and operating the DLT within the framework of the Cyber-Trust system. This security analysis is very important, towards making appropriate choices in the implementation/configuration approach that will be adopted, taking into account that security should be part of the design process and not considered afterwards. In addition, apart from security issues, privacy issues are also being considered under the same context. This deliverable constitutes a first security analysis report; a complementary version will be submitted in M30 that will be focusing more on cryptographic aspects.

1.2 Relations to other project activities

In D7.1 [67], we have analyzed the general security and privacy properties that any blockchain mechanism should have. In addition, the architecture of the DLT that has been chosen to be incorporated in the Cyber-Trust system is being described in D7.2 [68]; as stated therein, the Hyperledger Fabric constitutes the proper choice, taking into account the Cyber-Trust system's requirements, as well as the inherent features of the Hyperledger Fabric. Therefore, the present document performs a security and privacy analysis on the architecture described in D7.2, taking into account the general requirements stemming from D7.1.

Moreover, D7.5 presents a technical documentation of the prototype of the DLT implementing the forensic evidence, focusing also on measures that are to be taken in order to ensure compliance with the relevant legal framework [69]. Therefore, D7.5 also provided input to the present document, since specific implementation decisions have been taken therein.

Regarding privacy issues, D3.3 [66] sets some requirements that need to be fulfilled by the Cyber-Trust system, including – amongst others – the specific case of the DLT architecture. Hence, the present report takes into account the requirements presented in D3.3, in order to assess the conformity of the process performed by the DLT with the data protection legal provisions.

The implementation decisions stemming from this deliverable will provide significant input to the task T8.2 (platform integration and testing), which is related with the integration of all the tools into an operational environment that will provide the required services, as well as – as a clear consequence – to the task T8.3 concerning the pilot execution.

1.3 Structure of the document

This document consists of seven sections, including the current introductory section. More precisely, the structure of the document is as follows:

- Section 2 describes the generic security model of the Hyperledger Fabric, presenting all security goals as well as the vulnerabilities that need to be addressed. Some implementation decisions that have been already taken, due to the analysis performed in previous deliverables, are also briefly described.
- Sections 3-5 focus on specific types of security threats, aiming to provide a concrete analysis with the ultimate goal to take appropriate implementation\configuration decisions to alleviate these issues. More precisely:
    - Section 3 focuses on threats that are related with the underlying consensus mechanism.
    - Section 4 focuses on threats that are related with the smart contract's security.
    - Section 5 focuses on threats that are related with software and network security.

  In all cases, all relevant threats are being presented, whereas mitigation measures that are to be adopted are also concluded.
- Section 6 focuses on privacy issues, presenting all the relevant personal data protection risks that may occur and presenting how the Cyber-Trust DLT manages to resolve them.
- Finally, the main conclusions obtained are summarized in Section 7.

2. Hyperledger Fabric

Prior DLTs follow the order-execute architecture, in which the transactions are ordered first, via the consensus protocol and they are executed in sequence on each peer. This model has several limitations (see Deliverable D7.2) from which several threats and risks may arise. Fabric follows the execute-order-validate architecture (illustrated in Figure 2-1 along with the architecture of other DLTs) for executing untrusted code in adversarial environments.

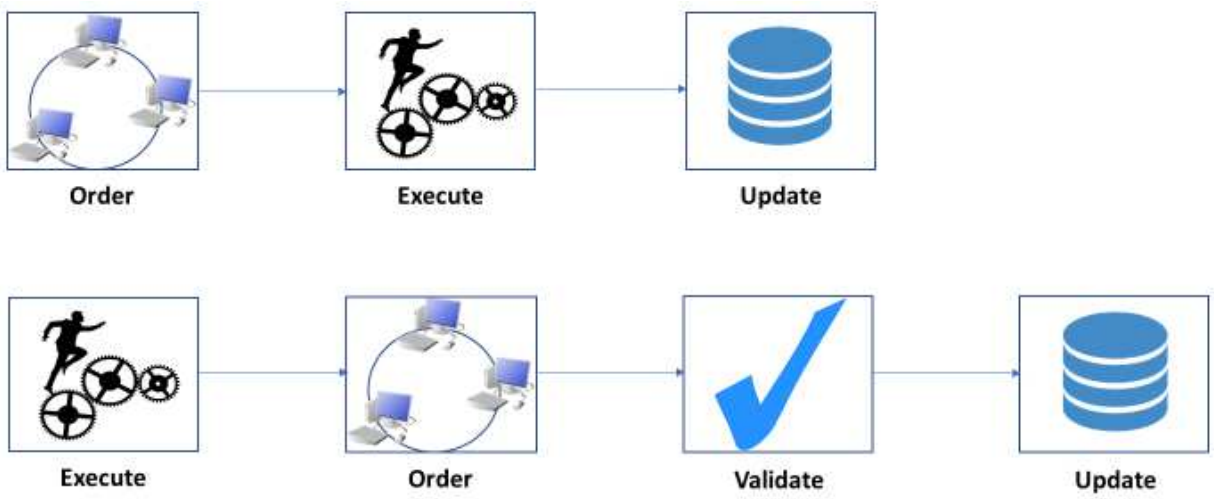

Figure 2-1: The common architecture (above) – The Fabric's architecture (below)

Hyperledger Fabric divides the transaction flow into the following phases: **a)** *execution*, in which a transaction is sent to the peers, it is simulated and as a result of this simulation the `rw-set` is created[1] — when the simulation is finished, the endorsing peer creates an endorsement and sends it to the client; **b)** *ordering*, in which the consensus process occurs; and **c)** *validation*, which is the last phase before the update of the ledger and in which the transactions are validated not from the consensus nodes but from the peers.

2.1 Fabric's CA security model

The data that is stored on Fabric – especially the integrity of the data – can be considered secure by the concept of the BT itself or by the implementation of the smart contracts. However, there are prerequisites to make the previous statement valid. The first one is the security at the network level which can be ensured by means of the Hyperledger Fabric CA, see Figure 2-2:. Fabric's CA, which is responsible for the renewal of the certificates, provides the registration and authentication of actors inside the network i.e. the LEAs or the ISPs in the context of Cyber Trust's blockchain.

[1] `rw-set` is the *read-set*, which is comprised of the dependencies `(key, ver)` along with the *write-set* comprised of the state updates `(key, val)`.

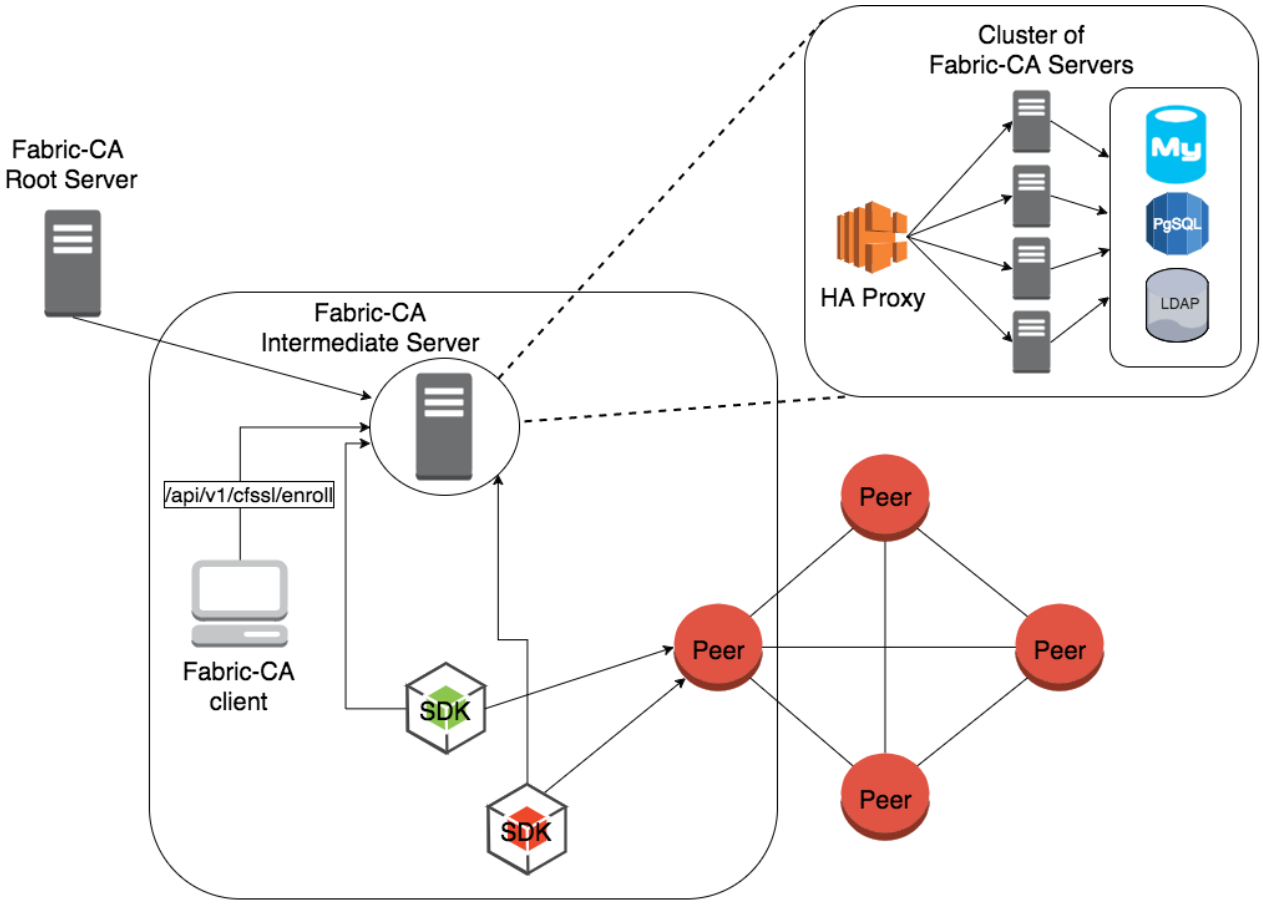

Figure 2-2:Fabric's CA architecture[2]

The central point of the architecture is the **Fabric-CA Intermediate Server**, which is actually a proxy towards the actual server responsible of the CA. It is significant to note that the **Fabric-CA Intermediate Server** can be a standalone server but it is not recommended for an optimal security of the DLT. Clients can communicate through the Fabric's SDK or the **Fabric-CA client** by means of the REST API.

At the host level, it is recommended to use data encryption. Fabric provides a native encryption to protect the stored data, which can be deactivated at any time. It is recommended to use encryption at the disk level.

2.2 Security Goals of Hyperledger Fabric

In Deliverable 7.1, we have analyzed the security properties that a blockchain mechanism should have [19] [20], namely: Common-prefix, chain-quality, chain-growth, and liveness. In this section, it is presented how these properties are applied in the DLT that we have chosen, i.e. The Hyperledger Fabric [1].

The common-prefix (i.e. consistency) is the most critical security property for blockchains and defines that all the local chains of the correct participants have a common prefix, which means that the first blocks are the same in each participant's local chain. Fabric satisfies this property, since the peers on a channel maintain the same ledger.

The chain-quality property defines an upper bound on the percentage of the blocks that have been formed by malicious parties. Generally, each block is created by a single entity and thus, the blocks can be marked as malicious or correct. However, Fabric cannot be considered amenable or vulnerable to this property due to

[2] https://fabric-ca-2.readthedocs.io/en/latest/users-guide.html

the fact that the batching of a transactions into a block, each bock's creation and signing, as well as, its dissemination it to the peers is taking place by all the OSNs in the cluster [15].

The liveness property defines that all transactions that derive from correct clients will be ultimately added to the blockchain. In Fabric, all the transactions are included into the ledger by the OS and then marked as malicious or correct.

The chain-growth property defines that as the time elapses the local chains of the correct peers will not stop growing, but this property is necessary only for blockchains that follow the LCR. In such DLTs, if the local chains of the correct peers stop growing, the network can be vulnerable to several attacks, such as double-spending, long range, etc. In such attacks, the malicious nodes, by creating an alternative longer chain, attempt to deceive the network and change a part of the ledger's history (See Section 2.3 for further details). In Fabric, all the transactions are stored in the world state and only the peers, in which a specific chaincode is installed, are the only entities capable to mark these transactions.

Apart from the chain-quality property, which is suitable to permissionless blockchains, Fabric does not also satisfy the common-prefix property in the presence of malicious behavior. For instance, an adversarial cluster can disseminate transactions to multiple OSNs in a different order. Thus, the blocks that the OSNs create from these transactions might not be the same and alternative states might occur. Even if the Kafka or the Raft cluster operates correctly, other security issues may arise. A violation of the common prefix property can occur if the OSNs are corrupted and deliver malicious blocks to multiple peers. Such behavior can be devastating in permissionless and anonymous networks, but in Fabric such actions are identified.

While, the common-prefix property cannot be satisfied in the presence of malicious behavior, accountability is a concept that makes the peers and the OSNs accountable for their actions. In Cyber-Trust case, if a malicious activity is detected, the participant who has performed this malicious action is further investigated and appropriate mitigation measures are being taken (e.g. removal from the Cyber-Trust's DLT environment, other measures as dictated by EU and national laws, etc.). Specifically, if the common-prefix property is not satisfied, which means that if some malicious actions occur in the execution of the protocol; it is possible to identify and hold accountable the parties who misbehave. Such entities could then face image loss or punishment, which averts them from performing such actions in the first place. Therefore, it is possible to ensure that the common-prefix property either holds or it is possible to hold accountable one or more entities that violated this property.

However, any violation of the liveness and the chain-growth is not so devastating in Fabric, since none of these properties can ensure agreement [19]. In a real world scenario, if the OSN, the peers or the Raft/Kafka cluster do not receive for a long period of time any update from a channel, this event will be reported and the individual connections will be monitored and analyzed in order to find where the transactions are dropped and which actions (malicious or not) caused this event. For example, in the ordering phase, some transactions might be dropped. For such an action, only the OS can be held accountable, because either the cluster is corrupted and willingly dropped these transactions or the OSNs are corrupted and they did not forward the transactions to the cluster. Since, the all the OSNs are known, such an action can be investigated and identified.

In Fabric, the atomic broadcast [21], i.e., (the total order of transactions) can only support the *deliver* and the *broadcast* events. These two events can be applied in the Fabric's case [1] and they can also be supported by the interface of the OS:

- $Broadcast(tx)$: This operation is invoked by a client in order to disseminate a transaction (tx) to the OSNs.
- $B \leftarrow Deliver(k)$: This operation can also be invoked (multiple times) by a client, where with input a positive increasing number k, it retrieves the block B by a peer. This block will contain a list of transactions and a hash, which corresponds to the previous block $(k-1)$.

The OS of Fabric ensures that the following properties, (as defined in [1], [15]) hold:

- *Agreement*: This property derives from the common-prefix property and states that for any pair of blocks (B, B'), which are delivered from honest peers by the numbers k and k', (respectively), if $k = k'$ then $B = B'$.
- *Hash chain integrity*: This property also derives from the common-prefix property [19] and states that for any set of correct peers in a channel (P, P'), if P delivers the block B with $ID = k$ and P' delivers the block B' with $ID = k + 1$, then the hash of the previous block that is included in block B' is the hash of the block B. Respectively, it is said that the peer P' violates this property if the delivered block B' does not contain the hash of the block B. Thus, any violation of the hash chain integrity by any peer can be detected and this property can be enforced to all the peers, even to the malicious, by means of the individual accountability.
- *No skipping*: When a block B_k is delivered by a correct peer P, then the peer P has previously delivered all the blocks B_i, where $i = 0, \dots, k - 1$.
- *No creation*: This property defines that if any correct peer P delivers any block B, then $\forall\, tx \in B$, each tx was previously broadcast by a client.

In Deliverable 7.1, it was also mentioned the importance of the CAP theorem [17] to the BT. Thus, it is shown here how Hyperledger Fabric achieves all the required properties to satisfy the theorem [17], [18]:

- *Consistency*: i.e. common-prefix property; which generally means agreement i.e. that there are no forks on the ledgers. It is mentioned above how Hyperledger Fabric satisfies this property.
- *Availability*: i.e. liveness; which means that the transactions created by correct clients will be included to the ledger and will be available to the peers.
- *Partition Tolerance*: i.e. fault-tolerance; this property ensures the continuous operation of the protocol, even if some of the participants fail (crash, misbehave, etc.). This property is satisfied according to the adopted consensus methods. (Further details will be provided in Section 3).

2.3 Classification of security vulnerabilities

Despite the fact that the technology behind blockchains inherits most of its security properties from cryptographic primitives (i.e. hash-functions); several threats, security vulnerabilities and limitations have been detected, identified and reported in blockchain deployments [22]. Given the potentials that this technology offers, there are multiple concerns regarding the blockchains' robustness [23] and multiple reports have been conducted regarding the cyber-security of blockchains. Thus, in this Section, it is provided a taxonomy about the identified threats and security risks in blockchains. This taxonomy is based on findings from different events and produces a classification structure, see Figure 2-3, about most of the risks, the threats and the limitations of blockchains:

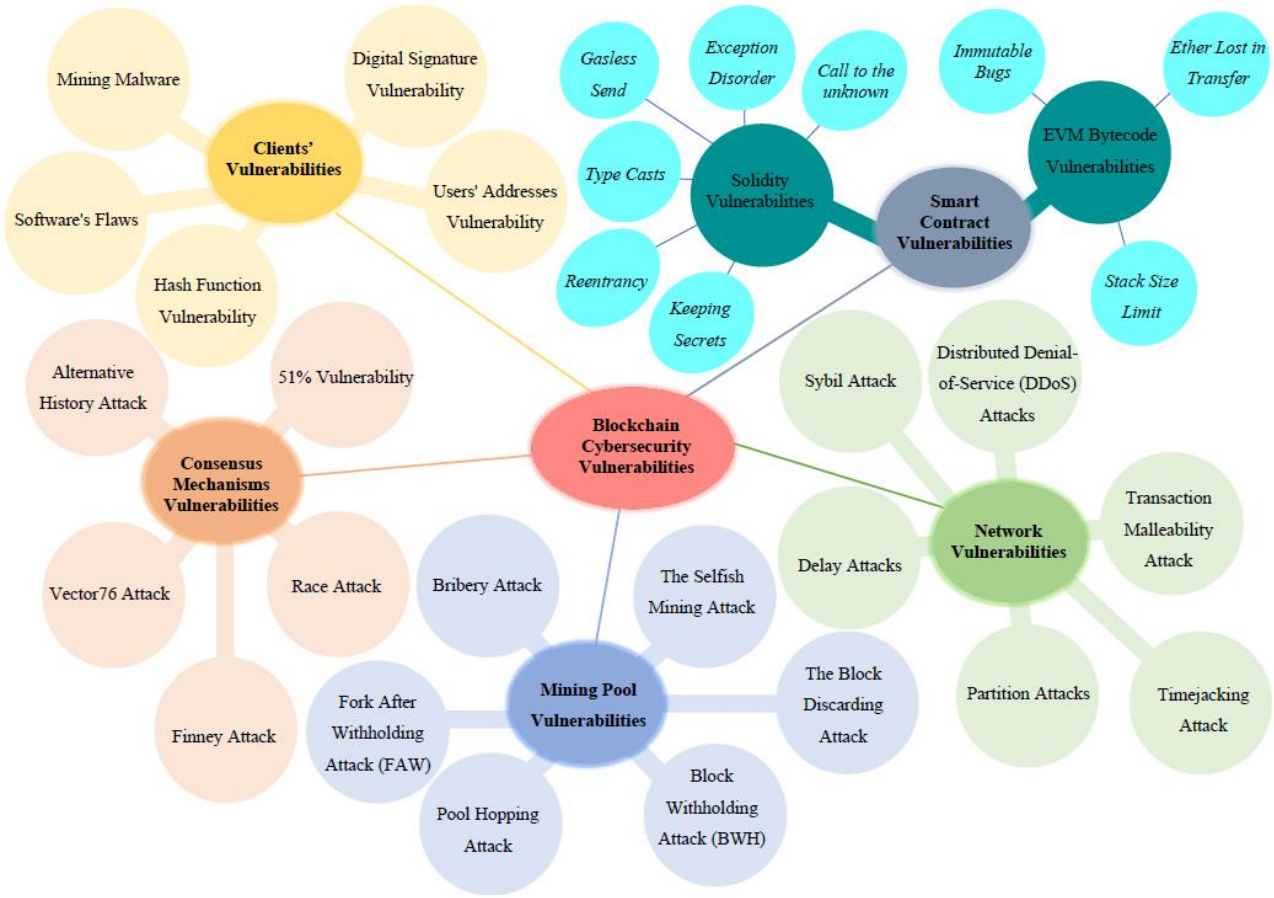

Figure 2-3: A taxonomy of blockchain attacks and risks [22]

1. *Consensus protocols' risks:* These risks concern the implemented consensus method on the blockchain mechanism and the protocol's tolerance against malicious actions. Such risks are:
   - 51% Attack: If the adversary has more than 50% of the "total power" in the network (computational, stake, etc.), then it can alter the protocol and execute other attacks such double spending, DdoS, etc.
   - Double-spending: This attack is the most common attack in blockchains, where the adversary attempts to double-spend (coins, ether, etc). In this attack are included the alternative history attack, the Race attack, the Finney attack [25] and the Vector76 Attack, which arise from the BitcoinTalk forum[3] and it is a combination of Finney and Race attacks[4].
   - Game theoretic risks: This category is part of the consensus protocols' risks and consists of the risks and strategies that malicious consensus nodes exploit and perform in order to maximize their revenue. Game theory can be also used to create incentives for the consensus nodes and avert them from misbehaving and executing attacks. The attacks that belong to this category are game theoretical and are:
     - BWH attack: In this attack, the attacker leaves the pool in which he is loyal and maliciously joins the targeted pool to mine blocks that he does not publish [26]. By withholding blocks the attacker does not gain any reward from the mined blocks but from the shared revenue of both pools. The unfortunate event is that if both pools attack each other, then the revenues are less if none of them had attacked.

[3] https://bitcointalk.org/index.php?topic=36788.msg463391#msg463391
[4] https://www.reddit.com/r/Bitcoin/comments/2e7bfa/vector76_double_spend_attack/

- Bribery attack: In several consensus mechanisms bribing other nodes in order to gain profit is permittable. Thus, the attacker can validate malicious transactions, since a part of the consensus nodes have been bribed to accept them [27].
- Pool hopping attack: By monitoring the shares and the block creation rate, the attacker decides if it is best for him to leave the pool [28]. If the attacker assumes that its earnings are not enough, leaves the current pool and joins another.
- Block discarding attack: By creating slave nodes, the attacker attempts to increase its superiority in the network. When new blocks are created by correct nodes, the attacker discards the correct blocks by disseminating its hidden and private chain [28].
- Selfish strategy: When the attacker acquires 25% of the "total power" in the network, executes this strategy [29] to take more nodes on its side and run other attacks such as DdoS and double-spending. By performing a block discarding attack and taking into account the pool hopping attack, the attacker disseminates malicious blocks and lures other nodes to hop into its pool.
- FAW attack: This attack is a combination of the selfish strategy [29] and the BWH attack [26]. This attack can be conducted by performing BWH attack on either a single pool or multiple. If the attacker solves a FpoW, the block is kept hidden. Depended on the chain's outcome the attacker either discharges the hidden block or publishes it and creates a situation similar to the selfish mining strategy.

2. *Smart contract risks:* The adoption of smart contracts in blockchains has led to several exploits and bugs (such as the DAO exploit and the Parity bug). In this category belong the Solidity and the EVM Bytecode risks.
3. *Client's software risks:* This category consists of the risks that can be exploited by an adversary at the user's software or hardware. Such risks are:
   - Digital signature risks: The validity of the transactions and the authentication of each user are based on the ECC, (such as the ECDSA), where the digital signature is created with the user's private key. Using the ECC is not adequate due to the fact that it does not provide the necessary randomness and the user's private key might be compromised.
   - Hash function risks: To ensure the correctness of transactions and the immutability of blocks, the BT is based on SHA-256 (Bitcoin), which is vulnerable to multiple threats, such as collision and preimage risks [22].
   - Mining malware: The attackers install malware on victim's machines and force the targeted CPUs to spend their computational power in mining cryptocurrency for the attacker5.
   - Software's risks: There are risks regarding the exposure of the user's private keys, (such as runtime and concurrency). An example is the software update of Blockchain.info in 2014, where the private keys were compromised only by viewing the public addresses [24].
4. *Network's Risks:* This category consists of risks in blockchains that occur in the peer-to-peer network layer, such risks are:
   - Partitioning attack: If the attacker isolates a set of nodes from the rest, the network can be partitioned via BGP hijacking and thus, hard forks may occur [30], [31]. This attack was demonstrated in the bitcoin network and showed that if one path attackers interfere with a minor number of messages, then the block propagation time can be significantly delayed. This action results to enforce the correct nodes to waste their mining power.

[5] https://www8.hp.com/h20195/V2/getpdf.aspx/4AA7-3873ENW.pdf

- Delay attack: In contrast to the partitioning attack the attacker does not have to gain full access to the target's traffic, but intercepting only one connection, he can introduce significant delays to the BTC.
- DdoS attack: This attack is one of the most common attacks not only on the internet but also on the BT. An example of a DdoS attack is the Mirai botnet in 2016 [32].
- Sybil attack: This attack is mostly executed in trust/vote-based consensus methods, in which the multiple fake identities that the attacker is able to creates can alter the protocol [33]. A sybil attack [34] on PoW-based protocols has no effect because the attacker will have to divide its computational power. In the context of Hyperledger Fabric, this is assumed to be an example of the more generic type of insider attacks.
- Transaction malleability attack: In the cryptocurrency-based networks, the attacker exploits this vulnerability to modify the identifier of a transaction (TX_{id}) in order to deceive the victim. An example of this attack is the Mt. Gox hack where coins were stolen from the exchange [35].
- Users' addresses risks: Such as identity theft and MitM can be executed by an attacker to obtain rewards that were intended for the supposed target. Such attacks are disastrous in cryptocurrency-based systems for the apparent reasons.

There are also other types of more typical network-oriented attacks, like wormhole, man-in-the-middle and SSL stripping attacks, which can be applied to Hyperledger Fabric. These are all analyzed in Section 5.

5. *Privacy risks:* Data privacy risks concern the data leakage by attackers via analysis. This analysis could expose valuable data regarding the data that have been stored on the DLT. This Such risks are:
   - Loss of confidentiality of personal data: An attacker manages to get access to personal data, without being authorized for such an access.
   - Transaction linkability: An entity that has access to the DLT manages to combine information concerning individuals so as to derive some conclusions (e.g. profiling of the users) for non-legitimate purposes and/or without informing users for this process
   - Private keys management. Compromisation of private keys results in identity thefts, as well as in privacy issues.
   - Malicious smart contracts: There is possibility that a smart contract might be able, either deliberately (due to a malicious programmer) or not, to leak personal information.
   - Non-erasable data: Due to the inherent immutability property of blockchains, data that have been put into a blockchain cannot be deleted. This gives rise to privacy issues, which are strongly related to the so-called right to erasure (or right to be forgotten).
   - Privacy interoperability across different blockchain-enabled scenarios. A blockchain can be applied in different kind of scenarios, which may demand diverse privacy requirements.

2.4 Cyber-Trust's implementation decisions

In order to fulfill the technical or legal requirement imposed by the ecosystem of Cyber-Trust. Most of the implementation were about data retention and data privacy.

Regarding data privacy, only the metadata of the actors are stored. This metadata consists of non-sensitive information such as name of organizations and members of the Cyber Trust platform. The nature of these data enables the sharing through all the peers of the network. To match the end user requirements of Cyber Trust, we also need to store the sensitive data, which is our case for the forensic evidence preservation. In this case, we use the private feature of Fabric. This feature allows us to partition the network into sub-networks. This means that the only participants of a sub-network can have access to the data that is published

in this particular sub-network. So, once an ISP publishes private data to its private network, he is the only one who can see data's content. A LEA can gain access to the forensic evidences by simply issuing a transaction. Once its demand is accepted, the LEA is added to the subnetwork. As a direct consequence of this addition, the content of the subnetwork is copied to the peer of the LEA. Only then he has access to the private data of the forensic evidences.

Regarding to the data retention, the private data have a TTL, which is given to the DLT at the creation time, since each jurisdiction has potentially a different length for the data detention. This TTL corresponds to the time that the data will exist inside the sub-network before being deleted.

During the development of Cyber Trust's DLT we also took the decision to use as consensus the default method, which is Solo. Afterwards, while examining the security analysis of Solo, we figured out that the Solo consensus method is not the best option for our DLT, despite the fact that Hyperledger Fabric is the best option for our case. Indeed, Solo is very suitable during the development phase as it necessitates no configuration but it has several drawbacks. In the latest version of Fabric, the consensus has even been deprecated by the developers as according to them too many projects used Solo despite its weaknesses.

One consequence of storing only metadata is the size of the transaction generated by inserting data to the DLT. Indeed, the size of the transaction has a direct impact on the throughput. As we are in the IoT ecosystem, we needed to design the DLT in order to achieve a high throughput of transactions.

3. Consensus security

The consensus in distributed ledgers is a state where all the participating nodes agree upon the same messages in a specific order [1], i.e. atomic broadcast [21]. In Hyperledger Fabric the OSNs atomically broadcast endorsements and achieve agreement on the total order of transactions. Once the agreement is achieved, even in the presence of failing nodes, it can be certified the consistency (i.e. the common-prefix property, see Section 2.2) and the availability of the distributed ledger.

3.1 Consensus methods in Hyperledger Fabric

An important differentiation in each blockchain platform is the support of pluggable consensus methods to enable customization in order to achieve trust models or other use cases [1]. If a consensus mechanism is operated by a small enterprise, it is deployed in a small network or it is managed by a single trusted authority, complicated protocols might be considered as an exaggerated drag on energy consumption or the protocol's performance. Thus, adopting a single node to order the clients' request and provide substantial consensus, such as the centralized protocols do, the overall system might work quite as well as other complicated consensus mechanisms.

3.1.1 Solo

Solo is a centralized protocol that is being used for testing purposes. This consensus approach necessitates only one node in the network in order to acquire the incoming messages and it provides a substantial consensus, allowing developers to concentrate on other matters, such as the development of the smart contracts. However, the drawback of the Solo protocol is that it becomes a SPoF and endangers the security of the network.

3.1.2 Kafka

Apache Kafka [5] is a distributed pub/sub system for log processing, written in SCALA and JAVA. The distributed message system of Kafka is developed for transferring high volumes of information with very low latency. The key concepts of Kafka, as defined in [5] are the topics, the producers, the brokers and the consumers. The records are published by the producers to a topic, which is in a form of stream of messages [5]. A topic, in which none or more consumers subscribe to consume information, is partitioned into different segment files, as shown in Figure 3-1.

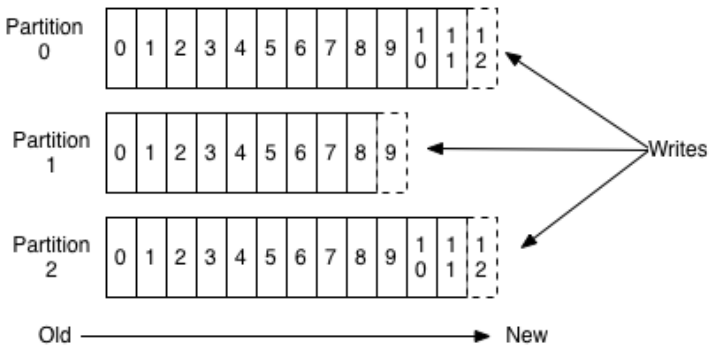

Figure 3-1: A screenshot of a topic[6]

[6] https://sookocheff.com/post/kafka/kafka-in-a-nutshell/

In Kafka, the messages are stored in an ordered partition with a unique identifier and when a producer publishes a message to a partitioned log, the brokers – which are a set of servers – store the message to the latest segment file. Then, only the consumers, who are subscribed to this topic can only consume the messages in sequence from the partition by issuing pull requests to the brokers.

The FT properties of Kafka are provided from the ZooKeeper. These properties can be achieved by replicating the partitions among the brokers. In simple words, Kafka follows the leader-follower model, where in each partition there is a leader and the rest of the brokers act as followers by passively replicating its actions. Thus, if the ongoing leader crashes or fails to submit a request, then one of the followers takes its place.

Hyperledger Fabric [1] deploys Kafka and its ZooKeeper as an ordering service to provide consensus about transactions' ordering. Since, the OSNs can operate as proxies among the network's peers and Kafka; the cluster itself can be executed on physical nodes independent from the OSNs. If an OSN receives a new transaction from a client, then this transaction is injected to the Kafka broker. When the maximum size of bytes is attained or a specific amount of time is elapsed or when the maximum amount of transactions is received, then a block is cut [1].

3.1.2.1 Kafka's Performance:

The performance metrics of Fabric is measured in [1]. Kafka showed significant results. In their experiment the authors used a) 5 peers as endorsers with each peer to belong to a different organization, b) a typical OS with one Kafka OSN and c) the default 256-bit ECDSA signing scheme. The evaluation showed that Hyperledger Fabric can achieve the magnificent transactions' throughput of 3500 TPS with latency less than a second, as shown in Figure 3-2. This result brought a significant revolution in the area of blockchains and particularly in the area of permissioned and business oriented blockchains. For this reason, Kafka, outperforming Solo, is the recommended protocol in version 1.0 of Hyperledger Fabric.

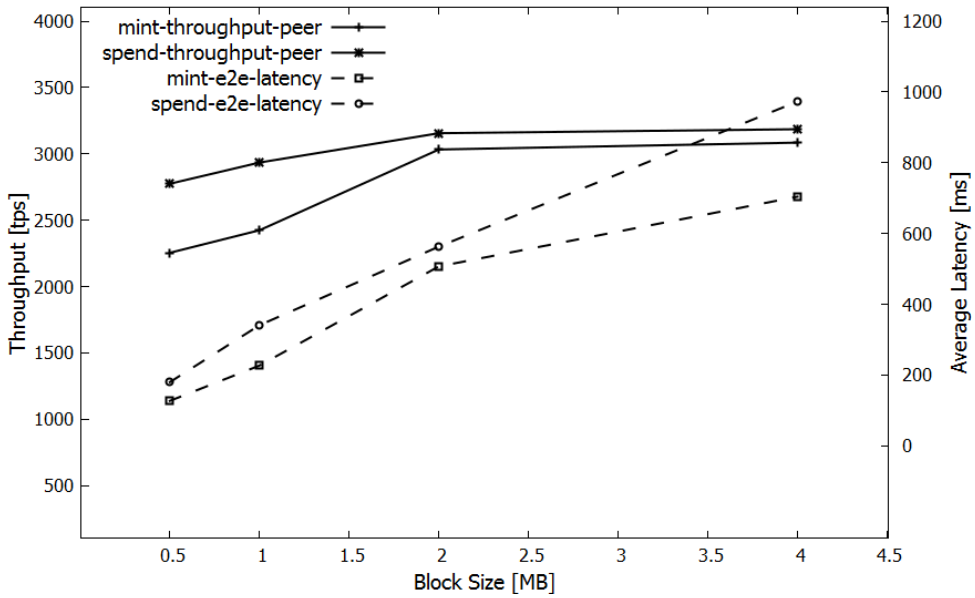

Figure 3-2: Kafka performance [1]

3.1.3 Raft

Raft [6], [7] is a consensus method equivalent to (multi-)Paxos, similar to Oki and Liskov's VSR protocol [8]; that follows the "leader-election" algorithm. The protocol achieves consensus by electing a leading node, which has the responsibility to accept and replicate the new entries from the clients without consulting other peers. The consensus algorithm is split into the following subroutines. Namely "leader election, log

replication and safety" to provide a strong leader and coherency. Raft is a consensus method that can endure crash failures. For example, if there are five nodes in a channel, which is the typical number in a Raft cluster, the protocol can tolerate the failure of two. While the protocol proceeds in terms, a node can be in one of the following three conditions "leader, candidate or follower". A term is an arbitrary time period and each term is characterized by a monotonically increasing integer.

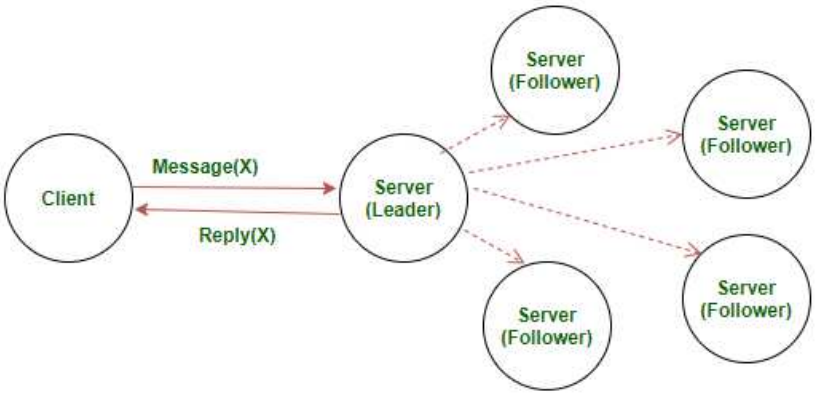

Figure 3-3: States in Raft consensus[7]

The leader is the principal entity capable to interact with the clients and replicates log entries to its followers. Each follower synchronizes with the leader. If the leader fails, the follower that has detected this divergence, casts a vote to the network and contests a new leader-election process in order to take its place. With the initiation of a new election process in each term, the contesting nodes may ask for support. Thus, nodes that request votes from others are considered candidates. Then, winner becomes a leader for this term. The above-mentioned states in the Raft are shown in Figure 3-3.

The protocol has to make sure that only a single node can act as a leader. If there is a split up, the ongoing term will finish with no leader and a new term will begin with a new election [7]. When the nodes communicate with each other, the ongoing term number is exchanged and a node may not realize when the term is altered. In this case, the node revises its outdated number and if it is a leader or a candidate then it falls into the follower state. The leader also broadcasts arranged heartbeats to its followers. If a follower does not receive any for a particular time-period then it presumes that the leader has crashed and casts a vote for a new election. Figure 3-4 shows in a simple manner all the transitions, in which an OSN might be. The basic communication between the nodes is happening with RPC calls and concerns the initiation of a new election process, in which the candidates request for votes; and the replication of log entries, in which the leader broadcasts a heartbeat and compels the followers to append new log entries.

[7] https://www.geeksforgeeks.org/raft-consensus-algorithm/

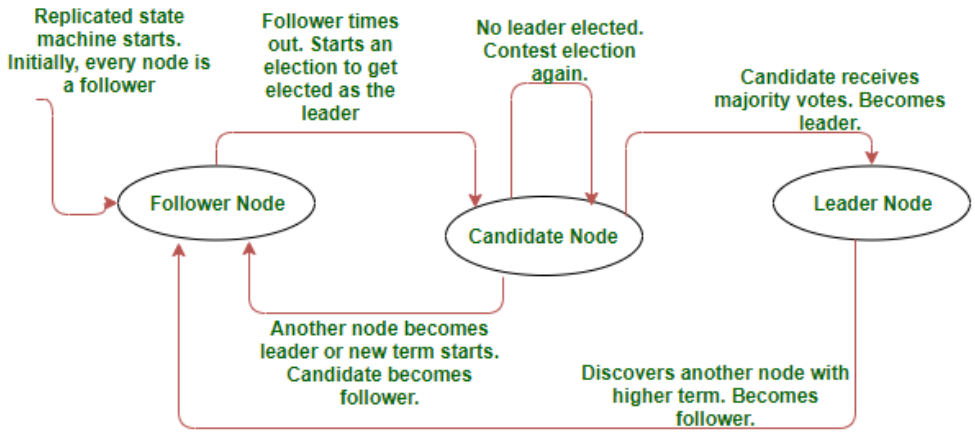

Figure 3-4: OSN status transitions[7]

3.1.3.1 Raft's performance:

The performance of the Raft consensus is measured in [9] prior to the adoption of Raft in Hyperledger Fabric, by an IBM distinguished engineer. In the experiment, it was used the same topology of Hyperledger Fabric v1.4.1 to compare the two protocols. The results, which are illustrated in Figure 3-5, showed that Raft outperforms Kafka with almost twice the transaction throughput and even less latency. Thus, Raft outperforming Kafka becomes the recommended protocol in the latest version of Hyperledger Fabric.

Figure 3-5: The throughput of Kafka (in the left) and Raft (in the right) [9]

3.1.4 BFT-SmaRt

BFT-SmaRt [10] is a modular SMR that can be used to provide an OS for Hyperledger Fabric [1]. In the absence of malicious validation replicas (VRs), the BFT-SmaRt's message pattern is similar to PBFT's and it is illustrated in Figure 3-6. The time in BFT-SmaRt is separated into *instances*, where in each instance, a *leader validation replica* (LVR) proposes a sequence of requests to be categorized and ordered. This is happening by disseminating a *propose* message to all the other VRs. When the *propose* message is received by a VR a security check occurs in order to be verified that the sender is indeed the leader and that all the transactions included in it are valid. Then the VRs store the proposed batch and disseminate a *write* message to all the other VRs. If a VR collects $\lceil (n + f + 1)/2 \rceil$ write messages for a batch, makes it public by disseminating an *accept* message. Then, if a VR collects $\lceil (n + f + 1)/2 \rceil$ accept messages it delivers the batch by denoting its decision for the ongoing instance. In case where the network does not achieve consensus, then a new leader is elected and a new instance begins [10].

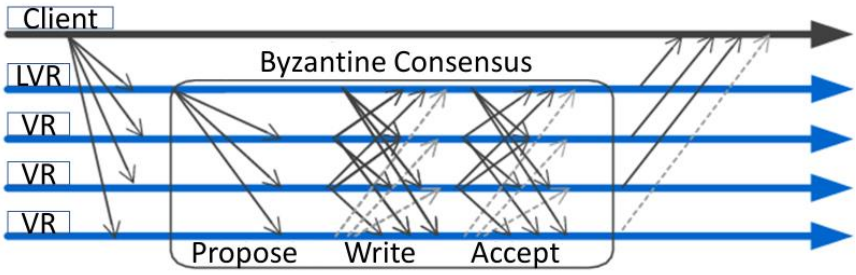

Figure 3-6: The message pattern of BFT-SmaRt [11]

BFT-SmaRt also implements WHEAT to provide faster replication, reduced latency and an efficient vote assignment scheme for the OS, without imperiling the security properties of the consensus method (i.e. the agreement and the liveness, see Section 2.2). The OS is comprised by the frontends and the cluster. In the cluster, in which malicious nodes may exist, the OSNs execute the protocol and order the transactions, which are received from the frontends. The transactions are in a form of envelopes and when a predetermined number of envelopes is collected a new block is created. The envelopes are transferred from the clients to the frontends, which in their turn relay them to the OSNs. When a new block is created, it contains except from the envelops, the digital signature of the block and the hash of the previous block. After its creation, the new block is disseminated from the OSNs to the frontends, which in the worst-case scenario can assemble $2f+1$ matching blocks. After receiving the generated blocks from the OSNs, the frontends send the generated blocks only to the peers in charge to manage the DLT. The design of the OS in BFT-SmaRt is shown in Figure 3-7 and demonstrates how the frontends communicate with the OSNs and the clients.

Each frontend is comprised by two major components; namely “the Fabric codebase” and the “BFT-shim”. The former acts as an interface for the clients to submit requests in a form of envelops and the later possesses the components to interact with the clients and the OSNs. These components are illustrated in the left part of Figure 3-7 and they mentioned below:

1. *The client thread pool* (Client Threads), which collects the envelops and sends them to the OSNs.
2. *The receiver thread* (Recv Thread), which receives the new created blocks from the OSNs.
3. *The BFT-SmaRt proxy* through which the envelops are sent to the OSNs and the blocks are received from them, and
4. *The Fabric’s SDK*, which confirms that the appropriate computations are executed regarding the invocation of transactions, the queries of blocks and the monitoring of events on a channel.

On the right part of Figure 3-7 are illustrated the OSNs. The OSNs are part of the cluster and each one manages a *block-cutter*, in which the envelopes are placed before the creation of a block. This component handles all the envelopes correlated with each channel in the DLT and creates batches of envelopes in order to be added in a block for a ledger correlated with a specific channel. When a pre-arranged number of envelops is collected for a channel, then the *block-cutter* alerts the *node thread* to create a new block.

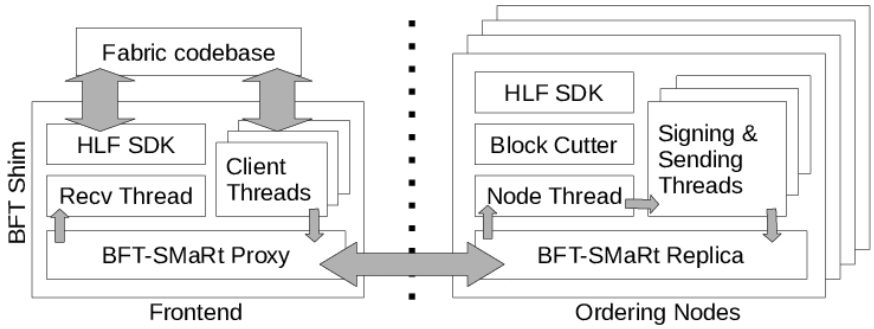

Figure 3-7: The OS architecture [11]

When a new block is created the *block-cutter* drains itself and a new integer is appointed for the next block. This increasing number is given to the *signing and sending thread* component accompanied by the specific header. The block-header consists of the foregoing increasing integer, the hash of the envelops and the hash of the previous header [11]. Commonly to the frontends, the *Fabric's SDK* is utilized to create and manage the data structures of the network. Furthermore, the ordering nodes also use the SDK to create all the cryptographic hashes of the blocks and their digital signatures via the ECDSA. When a new block is created, it is signed and delivered to all the operating frontends by BFT-SmaRt's API. The *BFT-SmaRt replica* apart from receiving, ordering and executing the disseminated transactions from the frontends, also provides supplementary capabilities. In short, these capabilities concern the state machine replication, and the configuration of the ordering nodes and the durability of a state in the extreme case of all the ordering nodes collapse.

BFT-SmaRt's validation process differs from the original that is presented in [1]. Hyperledger Fabric is flexible to blocks that include invalid transactions and for this reason the OSNs in Kafka can order the transactions without any validation checks. The validation process in Kafka, is taking place in the validation phase where the peers mark each transaction as valid or invalid. In BFT-SmaRt's case, the transactions can be checked for double spending by the *signing and sending threads* component prior to their dissemination to the peers. Thus, invalid transactions can be removed from the blocks before creating the blocks' signatures.

3.1.4.1 BFT-SMaRt's Performance:

The performance of BFT-SMaRt is measured in [11] and showed that the protocol can provide transaction throughput more than 10 thousand TPS and BCT less than 1sec. In their experiments, they used different sets of ordering nodes capable to withstand misbehavior. In each block are included 10 or 100 envelops, where each envelop is either 40 bytes, 200 bytes, 1 kbytes or 4 kbytes. The results are illustrated in Figure 3-8. In this experiment the authors used 4 orderers and they also stored 10 envelops per block. It can be observed that when one or two receivers (frontends) exist in the system, the maximum throughput is more than 50 thousand TPS for small envelop sizes.

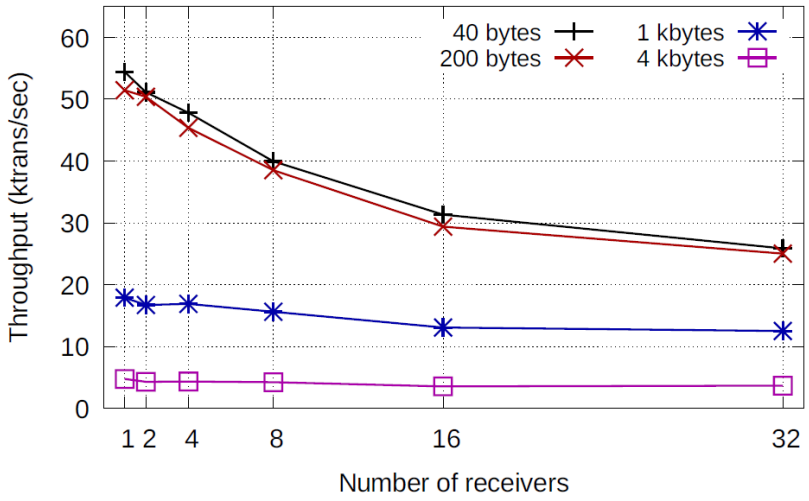

Figure 3-8: The performance of BFT-SMaRt

3.1.5 Comparison

Solo is not tolerant against faults and therefore it does not satisfy the CAP theorem. Furthermore, it can become a SpoF and for this reason it should not be deployed. Both Kafka and Raft use the "leader and follower" design to withstand failures and provide significant throughput. Although Kafka is very popular, various tricky components have to be managed in order to be implemented. In Raft, all these components are embedded into the OSNs, which means that there are less components that might fail. Kafka and ZooKeeper are designed to be implemented in a small network comprised by a few OSNs and the Kafka cluster has to be run by single entity. This makes all the nodes to attract this entity and does not provide much to escalate the decentralization of the system. On the other hand, Raft is more decentralized and provides much more throughput than Kafka does [12]. In addition, Solo and Kafka are deprecated in v2.0 of Hyperledger Fabric. BFT-Smart is also a protocol that provides significant transactions throughput, and it can also withstand malicious actions outperforming all the previous protocols. Unfortunately, BFT-SMaRt is a Java-based library and it is not yet adopted by the Hyperledger Project[8]. In addition, the protocol does not provide a very stable OS. In particular, it requires to operate two processes; the first build in Java and the second in GO; with a network socket bound among them, which might act as bottleneck in the system.

3.2 Threat's overview

The consensus protocol that is going to be implemented is one of most fundamental part to consider during the development of a blockchain system. The consensus method is the assurance for the reliable operation of the system, due to the fact that the nodes agree upon the validity of a transaction via this protocol. The 51% and the double spending attacks differ according to consensus protocol that is being used but they can cause huge irreparable damages in any case. The 51% attack intends to create an alternative chain in order to perform double-spending. Similarly, the BWH attack, the Selfish strategy and the other game theoretic risks derive from the consensus protocol that is being used and the incentives that it provides.

In Fabric, the transaction flow is different than the other blockchain mechanisms. The 51% attack and the aforementioned game theoretic attacks (see Section 2.3) are not even applicable to Fabric. The only attack that is applicable in our case is the double spending attack which is addressed by Fabric with a special way.

Fabric follows the execute-order-validate model (see Section 2), in which one or more malicious transactions can be sent from a client to the OS for ordering and hence these transactions can be included in a block. The

[8] https://www.hyperledger.org/

detection of double spending occurs, not only in the execution phase, but also when the transactions and the blocks have been ordered in the validation phase, where the peers check and update the read-write sets of the world state. All the transactions that are included in a block are marked as valid or invalid [1]. To perform that action, the peers check if the current version number that there is on the ledger is the same with that on the read-set. Then, in the validation phase, marks as valid only the first transaction that changes the version of the state, all the others are marked as invalid [1]. Thus, **double spending and similar attacks cannot be executed in Fabric, regardless of the consensus protocol that is being used.**

3.3 Threat's mitigation

Generally, in blockchains the most important threats can be dealt with the implementation of the appropriate consensus method. The consensus methods are characterized by their resilience against faults and these are relied on CFT and BFT design principles are capable of thwarting the most prominent threats.

3.3.1 Crash fault tolerant protocols

An atomic broadcast [21] is indicated by the $broadcast$ and the $deliver$ events, which may occur in the protocol multiple times. The broadcast event means that each node can disseminate a message with the $broadcast(m)$ event and the protocol via the $deliver(m)$ event outputs the message m. More precisely, considering a set of n nodes in a network, the following properties should be fulfilled [4]:

Validity: If any legitimate node n_i disseminates a message m_j, then ultimately n_i delivers m_j.

Agreement: If any message m_j is delivered by any legitimate node, then m_j is ultimately delivered by all the legitimate nodes.

Integrity: Any message m_j is not delivered more than once by any legitimate node. Furthermore, if any legitimate node delivers the message m_j and the sender n_i of the message is also legitimate, then m_j was formerly broadcast by n_i.

Total order: For any pair of messages (m_1, m_2) are delivered by a pair of legitimate nodes (n_1, n_2), it is said that n_1 delivers m_1 before m_2, if and only if, n_2 delivers m_1 before m_2.

The most prominent and successful way to reach consensus (i.e. implement atomic broadcast [21]) in DLTs which are vulnerable to $t < n/2$ node crashes, (where t denotes the number of the crashed nodes), is to implement a family of protocols which are known today as crash-fault-tolerant (CFT). The security against crashes that these protocols provide, has led to their adoption in several critical environments and it is model by identical rules for different implementations. In the CFT family of protocols, the time is parted into a sequence of views – such as in Snow White [3] – or epochs – such as in Ouroboros [2] – where in each time interval, a leader is voted, elected, or randomly selected to create, disseminate and propose a block. The security of the system is based on the most important property that CFT protocols possess: "If the leader collapses, or even if a set of nodes in the network speculate that the leader has collapsed, then a new epoch or view begins with a new leader".

3.3.2 Byzantine fault tolerant protocols

Recently, several consensus methods have been emerged to counter the capability of Byzantine nodes, often called adversaries, whose goal is to manipulate the network. The family of Byzantine-Fault-Tolerant (BFT) consensus methods aims to reach consensus knowing that a set of participating nodes expresses malicious/byzantine behavior. Similar to the CFT protocols, these protocols utilize a leader-election algorithm to elect a node, which will order the incoming messages in each epoch, view or round. Moreover, BFT protocols by their nature are CTF, which means that they can tolerate not only a set of byzantine failures but also the same number of crashes. Generally, a consensus method attains the BFT requirements if the malicious nodes, here denoted with f, do not exceed a percentage of 33% of the network, meaning that the following condition holds: $n \geq 3f + 1$.

3.3.3 Adoption of incentives

A blockchain mechanism requires a set of nodes to participate in the consensus process. Several rational nodes by working individually may perform actions to maximize their benefit against the common goal of the network. Other malicious nodes by working collectively may launch attacks or perform strategies to take advantage of the consensus process. However, different approaches have been used to optimize and examine the strategies and the actions that are being performed by blockchain nodes in order to identify and thwart their malicious behaviors, such as the Markov Decision Process (MDP) [13]. Recently, the mathematical tools of game theory have been utilized as a different solution in blockchains to analyze the interactions and misbehaviors between rational nodes. By utilizing the game theoretical analysis in a blockchain mechanism, it is possible to develop incentive mechanisms that prevent malicious acts and analyze the actions of rational players that mishandle the network.

Current state in Fabric: The Fabric can facilitate malleable trust assumptions. In particular any client might fail or presumed conceivably malicious. The peers are gathered into organizations and each organization creates a trust domain, in which each peer trust only the other peers of the same organization. The integrity of the network relies in the OSN, which consider the clients and the peers as potentially malicious. In Fabric, the trust assumptions for distributed applications are detached from the trust design of the consensus method. In simple words, a distributed application can settle its own trust model, which is transferred via the endorsement policy, and it is separated from the trust model that is implemented on the consensus method. Thus, attaching a reputation to the nodes' identities, incentivizes the clients, the peers and the OSNs to adhere to the protocol and not deviate.

3.4 Comparative evaluation of (semi-)permissioned blockchain solutions

Apart from Fabric, several other blockchain solutions have been proposed to secure IoT networks. Permissionless blockchains, by their nature, cannot be used in an environment where the IoT devices share critical information. The computational power that the PoW-based implementations need, can be considered as a bad choice in an IoT ecosystem where IoT devices must execute computations.

In contrast to permissionless, other blockchain solutions are marked by the fact that the participating entities that regulate the critical information are identified and can be considered accountable for misbehaving. Permissioned blockchains can provide the necessary transaction throughput and latency, without adopting the security and the design thinking of permissionless [16]. The design of public and cryptocurrency based DLTs (such as Bitcoin's) can act as bottleneck in a network where the performance of the ledger is the most important aspect.

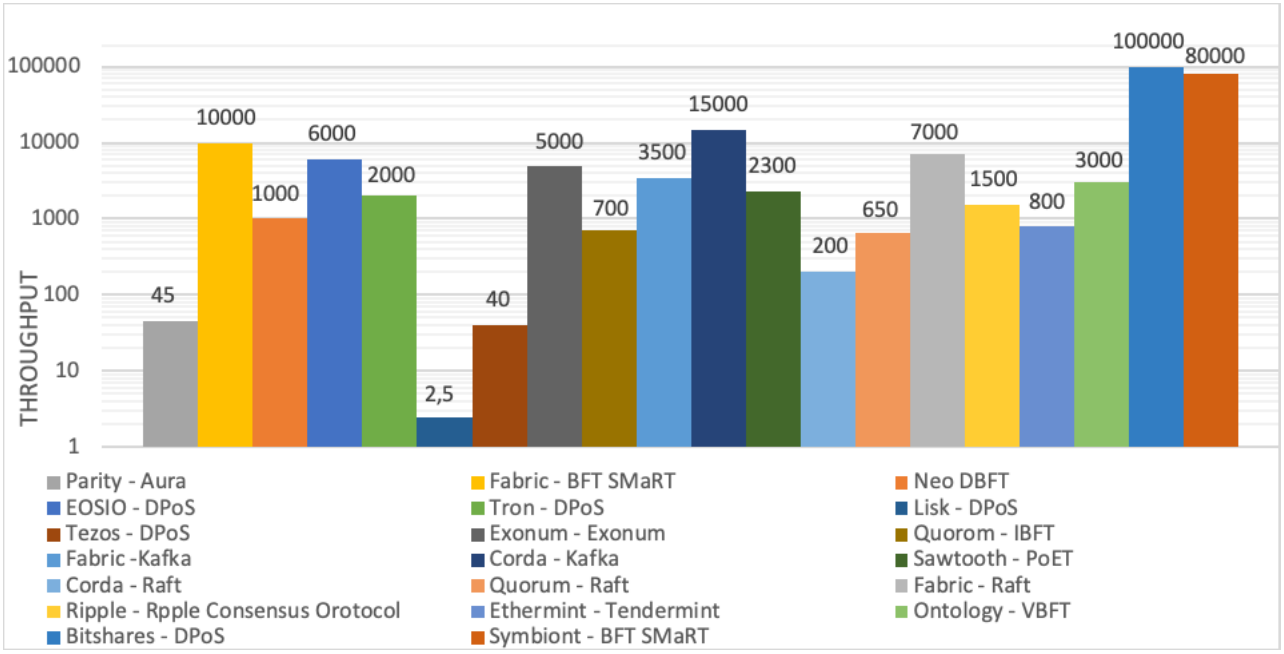

Figure 3-9: The transaction's throughput of each (semi-)permissioned blockchain platform

The transaction's throughput of the most prominent (semi-)permissioned blockchains according to the adopted consensus method is illustrated in Figure 3-9. It can be observed that the throughput of the most of them is high and some platforms outperform Hyperledger Fabric. The corresponding latency of each one of these DLTs and their implemented consensus methods are illustrated Figure 3-10, where only the transaction latency less or equal to 1sec can be considered acceptable to the IoT ecosystem.

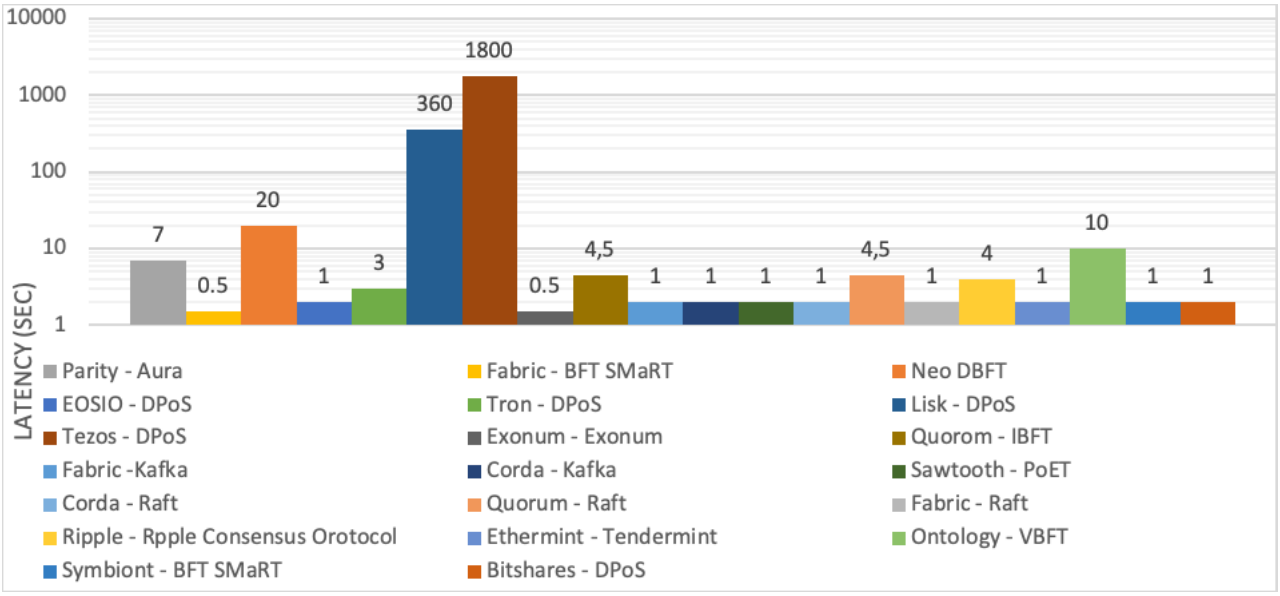

Figure 3-10: The BTC of each (semi-)permissioned blockchain platform

The transactions' fees, the low network's throughput and the high BCT are not the only options that we had to take into account in order to choose which DLT and which consensus method we are going to implement. The idea of misbehaving without consequences is not a good option for Cyber-Trust's case. Thus, in **Error! Reference source not found.** it is illustrated the performance and the tolerance (CFT or BFT) of several permissioned and semi-permissioned blockchain solutions regarding their implemented consensus methods.

Tx-class and Lt-class denote the given characterization according to each platform's transaction throughput and latency, (respectively). These two factors are the most crucial to be considered in an IoT ecosystem where a transaction should be disseminated and stored into a ledger in matter of seconds or even less. Thus, in **Error! Reference source not found.** it is mapped a three-valued scale (Low, Med and High) to denote the Tx-class and Lt-class, as well as the overall performance of various permissioned and semi-permissioned blockchains. The overall performance characterization is derived from the combined characterization of the Tx-class and the Lt-class. The transaction throughput (Tx-class) is defined to be:

- Low: If the transaction throughput is less than 500 TPS,
- Med: If the transaction throughput is between 500 and 1500 TPS, and
- High: If the transaction throughput is more than 1500 TPS.

Respectively, the Lt-class is defined to be:

- High: If the transaction's latency is less than 1sec,
- Med: If the transaction's latency is between 1 and 10sec, and
- Low: If the transaction's latency is more than 10sec.

Table 3-1: Comparative evaluation of (semi-)permissioned blockchain solutions

Consensus	Fault Tolerance	Platforms	Throughput	Tx – Class	Latency (BCT)	LT – Class	Overall Performance
Aura	BFT: $2f+1$	Parity	35-45	Low	3-7sec	Med	Low
BFT-SmaRT	BFT: $3f+1$	Fabric	> 10000	High	0.5sec	High	High
		Symbiont	80000	High	<1sec	High	High
DBFT	BFT: $3f+1$	Neo	< 1000	Med	15-20sec	Low	Low
DpoS	BFT: $3f+1$	EOSIO	1000 – 6000	High	< 1sec	High	High
		Bitshares	100000	High	1sec	High	High
		Tron	>2000	High	3sec	Med	Med
		Lisk	2.5	Low	6min	Low	Low
		Tezos	40	Low	≈30min	Low	Low
IBFT	BFT: $3f+1$	Quorum	400 – 700	Med	4.5sec	Med	Med
Kafka	CFT: $2t+1$	Fabric	3500	High	<1sec	High	High
		Corda	3000 – 15000	High	1sec	High	High
PoET	BFT: $\theta\left(\frac{\log\log n}{\log n}\right)$	Sawtooth	1000-2300	High	<1sec	High	High
Raft	CFT: $2t+1$	Corda	100 – 200	Low	1 sec	High	Low
		Quorum	≈ 650	Med	4-4.5 sec	Med	Med
		Fabric	≈ 7000	High	< 1 sec	High	High
RPCA	BFT: $4f+1$	Ripple	1500	High	4 sec	Med	Med
Tendermint	BFT: $3f+1$	Ethermint	200-800	Med	1 sec	High	Med
VBFT	BFT: $3f+1$	Ontology	>3000	High	5-10 sec	Med	Med
YAC	BFT: $3f+1$	Iroha	several thousands	High	< 5 sec	Med	Med

An implementation of the same consensus method on different DLTs can provide different performance. For example, Kafka can provide 3500 TPS in Fabric [1] and 3 to 15 thousand TPS in Corda. It is obvious that in the

case of the Kafka, Corda outperforms Fabric. Unfortunately, the protocol is only experimentally implemented in Corda and it does not provide a stable notary service[9]. Bitshares, EOSIO and Symbiont can provide 100000, 6000 and 80000 TSP respectively, with each transaction to be appended in a block in 1sec or even less. Bitshares and Symbiont have a tremendous throughput, but unfortunately, these consensus methods are not deigned to be deployed in an IoT ecosystem and the monetary concept that they possess cannot be easily substituted with IoT-based criteria. BFT-SMaRt seems to be a quite appealing protocol for our case, but there are several issues that might arise, (see Section 3.1.5).

Fabric's performance is not the only aspect that led us to our option. The aforementioned (semi-)permissioned platforms follow the Order-execute model [1] and thus, suffer from various restraints that affect the security of the system [16]. When the smart contracts are executed, they are run consecutively and on each peer that participates in the consensus process, which in its turn is hard-coded within the system. Fabric follows the execute-order-validate model, addresses the above restraints [1], [16] and strengthens the necessary security properties that a secure blockchain mechanism should have.

Taking into account the above research and the comparison in Section 3.1.5 it is clear that Cyber-Trust's decision to use Hyperledger Fabric with the Raft consensus protocol is by far the best option.

3.5 Conclusions and next research steps

The consensus implementation can be considered as one of the most important components of a DLT. A CFT consensus method might initially seem to be at odds with a secure DLT, but this is rather not the case in Fabric. **CFT protocols are amenable to permissioned networks for specific blockchain use cases in which the need for a secure distributed database is more visible.** The permissioned networks consist of identified entities that collaborate to achieve consensus even in the presence of malicious peers.

The fact that all the peers in Fabric can be held accountable for their actions provides them an incentive to follow the consensus method. Regardless of the implemented type of consensus (i.e. CFT or BFT), the Hyperledger Fabric can address not only the most common types of consensus-oriented attacks, i.e. the double-spending attacks, but also more sophisticated ones by its design. **The combination of Fabric and a CFT consensus protocol is considered to be ideal in an enterprise environment [16].**

BFT protocols, like BFT-SMaRt, are left for future research (as they are still being researched and they have not currently been tested in a production environment) and it remains to be seen if this protocol is indeed going to be adopted by the Hyperledger project. Another BFT protocol that has been implemented on Fabric is the HoneyBadgerBFT [36], [37]. This particular protocol experimentally achieves a tremendous throughput of 100 thousand TPS in Fabric; unfortunately, this throughput has not been validated in a real-world environment yet. Thus, it is left to see how such an idea can be implemented and if it is possible to add "internal" incentives and trust assumptions to all the peers, clients and OSNs in Fabric in order to make them follow the protocol.

[9] https://docs.corda.net/docs/corda-enterprise/3.3/design/kafka-notary/design.html

4. Smart contract's security

Smart contracts are executable programs, which implement operations that have been agreed among the entities on a DLT network (see Figure 4-1). Each entity executes the contract locally and submits the result to the system, while they all work together to select the result to put into the DLT's ledger.

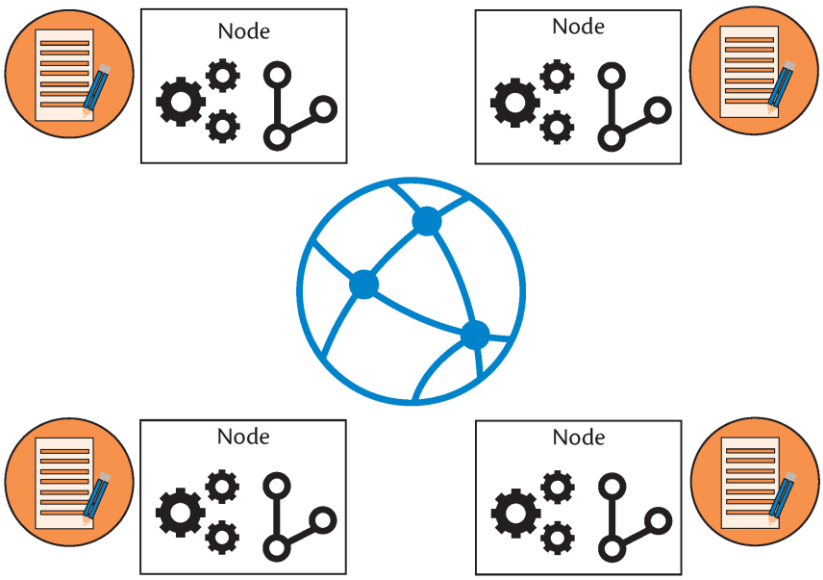

Figure 4-1: Smart contract system based on blockchain technology [61]

The persistence of smart contracts can induce various programming errors, which can eventually lead to malicious behavior and exploits. Therefore, the smart contracts can be considered as an easy target for malicious programmers who observe the program's code in order to discover exploitable bugs.

4.1 Potential risks

Most of the smart contracts' vulnerabilities are typically caused by programs' defects (e.g. design flaws, coding errors, etc.), as indicated in **Error! Reference source not found.**; According to [63], a large number of smart contracts (nearly 50% from a sample of 19,366 available ones) have been found to be vulnerable. As we are interested in the permissioned setting and the details of smart contracts for Hyperledger Fabric (referred to as *chaincode*), our subsequent discussion focuses on them.

Table 4-1: Taxonomy of vulnerabilities in smart contracts [62]

Vulnerability	Cause	Vulnerability	Cause
Call to the unknown	The called function does not exist	Immutable bug	Alter a contract after deployment
Out-of-gas send	Fallback of the callee is executed	Ether lost	Send ether to an orphan address
Exception disorder	Irregularity in exception handling	Stack overflow	The number of values in stack exceeds 1024
Type casts	Type-check error in contract execution	Unpredictable state	State of the contract is changed before invoking
Reentrancy vulnerability	Function is re-entered before termination	Randomness bug	Seed is biased by malicious miner
Field disclosure	Private value is published by the miner	Timestamp dependence	Timestamp of block is changed by malicious miner

Legend

Source code

EVM bytecode

Blockchain mechanism

Smart contracts that have been developed in Hyperledger Fabric [1] use general-purpose programming languages (i.e., Java, Node.js and Go). In general, smart contracts are programmed using DSLs, which have been defined by precise features and restrictions for blockchains. Hyperledger Fabric adopts the above programming languages, but in the absence of features and restrictions, the potential risks related with the platform might differ from the risks that other permissioned DLTs face. This occurs due to the fact that these platforms are related with smart contracts that are written by DSLs. In this section, a list of potential risks [14], [64] associated with the Hyperledger Fabric's chaincode is provided. The indicative risks have been mentioned below according to their philosophy.

4.1.1 Non-determinism ascending from programming languages

The determination of the chaincode is a crucial concern, since it is conducted in identified and independent peers [1], [14]. If the chaincode is non-deterministic, the endorsing results from the peers to the clients in the execution phase might be inconsistent. Inconsistencies may arise from several reasons, such as the following:

- *Global Variables:* Can change innately and cause non-determinism
- *Generating Random Number:* Can be used in various cases and lead to dissimilar behaviours.

```
1  // Something like a lottery application
2  // Users predict a number
3
4  // User's prediction
5  pred := arg[0]
6
7  // Answer
8  rand.Seed(seed)
9  sel := rand.Intn(10)
10
11 if pred == sel {
12    PayPrizeToUser(user, prize)
13 }
```

Figure 4-2: An example of non-deterministic risk [14]

In Figure 4-2, an example of non-determinism is represented, where the users guess a randomly generated number. If the number is true then the users gain their prize. The chaincode, during the endorsing phase is simulated in each endorsing peer. Thus, the generated numbers might be dissimilar in each peer. In Go, the seed is settled as 1 and it is easy to predict. Thus, in adversarial environments it should be set as: `rand.Seed(time.Now().UnixNano())`.

- *System Timestamp:* It is difficult to confirm if the timestamps are executed concurrently in each peer.
- *Map Structure Iteration:* Due to the hidden implementation details of the Go programming language, when an iteration with map structure is used, non-determinism may arise and the order of key values might differ.
- *Concurrency of the Program:* In Go, concurrent programs are supported (by using goroutine). If a program is executed concurrently with another, non-determinism might occur and a double spending problem might arise.

4.1.2 Non-determinism ascending from accessing peripheral of the DLT

Non-determinism might also arise from accessing information outside of the DLT, where such risks may arise from in the following cases:

- *Web Services:* In several cases, the business logic that characterizes the smart contracts, can provide information peripheral of the blockchain. In such cases a trusted entity called "oracle" needs to be accessed. A common practice is to call APIs in order to implement functionalities by a *Trusted Third Party* (TTP). If a web service returns divergent results, then the ledgers might fork.
- *Calling External Libraries:* In a similar way to web service, the external libraries should not be trusted.
- *External Command Execution*: Useful external commands can be executed via the *os/exec* package, but sometimes the results are not the same between the endorsing peers.
- *Accessing External Files:* In a similar way to the *External Command Execution,* if the chaincode accesses external files, the results on the endorsing peers might differ.

4.1.3 Fabric specification

The risks in this category are depended on Fabric's implementation.

- *Chaincode Invocation Between Different Channels:* If two or more chaincodes are executed on the same channel, then no risk will arise by invoking one chaincode [14] from another. On the other hand, if the `InvokeChaincode()` method is used and the invoked chaincode is on a separate channel, then all the calls from the `PutState` method will have no effect to the world state.
- *Range Query Risks*: In order to access and obtain the results from the state database about the private data, the key's history, etc., the peers execute queries with the `GetHistoryForKey()`, `GetQueryResult()`, and `GetPrivateDataQueryResult()` commands. These queries are not executed again in the validation phase, which means that phantom reads cannot be detected.
- *Chaincode sandboxing:* The Hyperledger Fabric's chaincode runs in a secured Docker container (as also presented in project's deliverable D7.2 [68]) that is isolated from the endorsing peer process. The chaincode however has sufficient privileges that could be exploited in a malicious manner to e.g. install arbitrary software, perform port scanning, exploit vulnerable hosts, etc. Such aspects are treated in more detail in Section 5 that deals with software and network security).

4.2 Threat's mitigation

Cyber-Trust is aware of aforementioned risks, since these risks are the only ones that have been reported on Fabric's documentation[10], discussions on GitHub[11], research papers, as well as blogs[12]. As a general rule-of-thumb, a number of possible proactive measures can be taken to increase CTB chaincode's security [64]:

- Execute chaincode with privileges other than *root*;
- Define time limits while executing chaincode (to also deal with denial of service attacks);
- Ensure that chaincode cannot achieve persistence by other; and
- Check chaincode's input arguments for escape characters.

[10] https://hyperledger-fabric.readthedocs.io/en/latest/index.html
[11] https://github.com/hyperledger-archives/fabric/issues/1273
[12] https://gist.github.com/arnabkaycee/d4c10a7f5c01f349632b42b67cee46db

These are typical counter-measures that have been considered during the CTB implementation. However, there are also a number of other approaches for mitigating threats in smart contracts that depend on some formalization technique [70].

Symbolic execution techniques. Are being used by software testing tools that mostly perform static analysis of smart contracts' security and privacy to identify weaknesses in their source code. Tools like Oyente[13], Securify[14], and GASPER [71] are among those providing this type of functionality[15], but they are focusing on Ethereum smart contracts. Other more generic tools include SmartCheck[16] that can check a smart contract for vulnerabilities and a multitude of bad practices, highlighting them in the source code and also giving a detailed explanation of the problem; this solution's drawback is that it is provided as a service, requiring the source code to be uploaded in an online platform.

Formal modelling techniques. The smart contract is modelled by using a precise set of statements defining the relationship between the smart contract's internal components, and resulting into an unambiguous communication, and replicable/reproducible results [72]. As an example, the world state in Fabric, which is in the form of `(key, value, version)` [1] maintains a cache of the ledger's latest states, which are in the form of `(key, value)` pairs[17]. An application could invoke a chaincode that can provide e.g. (`get`, `put` or `delete`) operations on the state via Fabric's API methods. These methods can be specified as:

- `GetState(key)`: With a key string as input, it outputs the latest entry regarding this key. If such an entry does not exist, it outputs empty.
- `PutState(key, value)`: This method takes as input an entry (a key string and a value array) and puts it at the end of the world state's sequence.
- `DeleteState(key)`: With input a key string, deletes the respective state.

This type of modelling can be used to detect design flaws and deficiencies in the chaincode's source. Formal modelling is also used to properly analyze other properties of a blockchain such as scalability, performance, and privacy.

Model or property checking. This is related to the previous technique and models the smart contract as a finite-state system that behaves exactly according to its formal specification or correctness properties (and likewise, it has also been used to analyze other properties of a blockchain). In many cases, the process is assisted by model checker, i.e. an automated tool that verifies the correctness and important properties of smart contracts; SPIN[18] and UP-PAAL[19] are examples of such open-source tools.

4.3 Conclusions and future research steps

There is a recent growing need for provable security (and privacy) properties of blockchain solutions' smart contracts which could be analyzed based on formalization techniques. This approach seems to be promising and could be examined as future research work.

[13] https://github.com/melonproject/oyente
[14] https://securify.chainsecurity.com
[15] https://consensys.github.io/smart-contract-best-practices/security_tools/
[16] https://tool.smartdec.net
[17] https://hyperledger-fabric.readthedocs.io/en/release-1.4/ledger/ledger.html
[18] http://spinroot.com/spin/
[19] http://www.uppaal.org/

5. Software and network security

The rise of Hyperledger Fabric blockchain is accompanied by security risks and concerns (some of which might not have been explored yet) that can be detrimental to the DLT operation and performance if they are not accounted for [47]. This section presents a brief overview of such attacks along with possible countermeasures that could be adopted for their mitigation.

5.1 Threat's overview

Most of the possible threats related to the software of Hyperledger Fabric along with any network-related attacks that might be exploited by intruders to break down the Hyperledger Fabric's network are described below.

Insider threats. Hyperledger Fabric relies on some trusted parties and centralized services (i.e. Membership Service Provider (MSP) admins and the OS) to provide a generalized platform for establishing a permissioned blockchain, which makes these services vulnerable to Insider threat [47, 49], when a peer become malicious and behave incorrectly, for instance, when it tries to maximize its own profit or becomes corrupted by an attacker. Insider threat could allow for further attacks such as DDoS attacks, Crash faults, 51% attack, Sybil attacks and Man in the Middle Attacks.

- *Compromised Membership Service Provider (MSP)*: MSP is one of the defining aspects of Hyperledger Fabric because it is responsible for managing identities of all participants in the network including clients, peers and ordering nodes (Figure 5-1). Therefore, in case that the MSP is compromised or becomes malicious because of a security breach, it can cause catastrophic damage to the network [49], by spreading false information to the network, adding (or blacklisting) identities and nodes to (from) the blockchain as desired, submitting new transactions that did not happen or should not be submitted, manipulating the amount and type of access a given identity has to the blockchain network [47], which open the possibility for further attacks such as 51 % attack, Sybil attack, Invalid ID Attack, Boycott Attacks, etc.

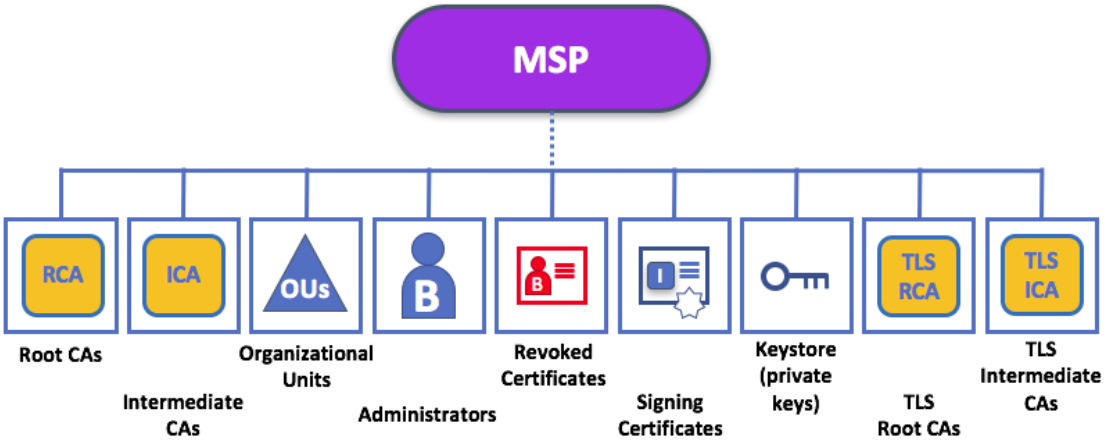

Figure 5-1: Local MSP structure[20]

- *Compromised Ordering Service (OS)*: OS consists of ordering nodes (also called orderer nodes) that establish the total order of all transactions and broadcast them as blocks of transactions to all connected peers in the network, while peers which are not connected to the ordering nodes get

[20] https://hyperledger-fabric.readthedocs.io/en/v1.1.0-rc1/membership/membership.html

blocks from other peers via gossip protocol [51]. Because Fabric's design relies on deterministic consensus algorithms, the blocks generated by the OS are guaranteed to be final and correct [47]. Therefore, a compromised OS can cause significant damage to the Hyperledger Fabric members by blocking transactions from certain peers for updating into the ledger (i.e. Sabotage attacks), withholding certain blocks, or manipulating the release of certain blocks which would favour certain ordering nodes. Further, a malicious OS can corrupt the all network by sending different versions of the blocks in response to the broadcast and deliver requests [47]. The OS is also responsible for setting the channels configurations including the Batch Size (i.e. A set number of pending transactions before catting the block) and the Batch Timeout (i.e. the amount of time to wait after the first transaction arrives for additional transactions before cutting a block) [52]. Thus, a malicious OS can change the value of Batch Size to an extremely small/extremely large value or manipulate the Batch Timeout by decreasing it too much and therefore harm the throughput of the network [52].

- *Malicious Validating Nodes and clients:* validating nodes are responsible for validating the transactions according to the "Validating System Chaincode" and updating the ledger state. Therefore, malicious validating nodes could authorise double spends (i.e. Double spending attacks) and break the integrity of the ledger. Similarly, malicious clients can pollute the blockchain and increase dramatically its size by keeps sending a constant stream of invalid unendorsed transactions to the OS. Since the client has "write access" to the ledger, ordering nodes pack the invalid transactions into blocks and sends them to the validating nodes. Even though the validating nodes mark the transactions as invalid, they are still included in the blockchain [47].

DDoS (Distributed Denial of Service). Denial of Service (DoS) is a cyber-attack in which the attacker exploits the network connection to disrupt the normal traffic of a targeted service to make it unavailable for legitimate users. The attack can be launched with different methods such as sending repeated SYN message requests (i.e. "SYN flood"), or packets larger than the maximum byte allowance (i.e. "ping of death") [50]. A single machine can be used to lunch automatically a DoS attack, However, if a DoS attack is launched from multiple machines, usually referred as zombies or bots, it is called a Distributed Denial of Service (DDoS) attack. Hyperledger Fabric network is inherently DDoS tolerant as is distributed and redundant. However, by centralizing the OS, Fabric becomes somewhat prone to DDoS attacks, especially with the consensus method Solo, which involves a single OS [47]. Further, the parties that can host nodes and/or transact on the blockchain network can be restricted to known identities rather than anonymous. Hence, the identity of an endorser is known to everyone in the network; this information makes DDoS attacks possible on selective endorsers to block certain client transactions for updating into ledger [48], degrading the overall network efficiency, or blocking the whole blockchain by stopping chaincode execution and/or the consensus network [51].

Wormhole attacks. In this attack, within a private network, an insider malicious peer creates a virtual private network (or channel) with the outside network and leaks the information of its own private network [48]. In Hyperledger network, an additional security layer (i.e. access control mechanism) is used by creating private channels between members of the network. Only peers inside the channel have access to the ledger of the channel and their identities are known to every other members of the channel. The used access control mechanism entirely depends on trusting every member inside the channel. Therefore, if a single peer member is compromised or become malicious and colludes with the outer adversary, wormhole attack is possible and can lead to leakage of confidential information of all members of the channel [48].

MitM & SSL Stripping attacks. In Hyperledger network, each peer has a client-side interface to receive input data and to enable clients to invoke transactions on the fabric, where transactions generally involve confidential data being passed as input. Insecure interfaces can lead to information leakage, especially by stripping away the encryption offered by HTTPS (called SSL Stripping). SSL Stripping is a serious cyber threat to Hyperledger network since their access control mechanism entirely depends on generating trust among mutually untrustworthy peers [47]. In this attack, once attackers gain access to the network, they can act as a Man-in-the-Middle (MitM) to intercept connections over the network and get data as plaintext.

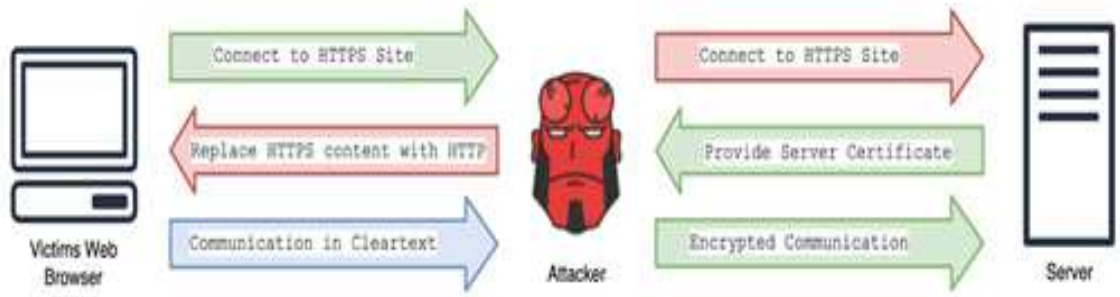

Figure 5-2: SSL Striping attack [47]

Figure 5-2 illustrates the working method of this attack, where the attacker intercepts the connection between the client and the server, behaves like server for the client and client for the server. Once the client sends an HTTPS (encrypted) request to the server asking for the certificate and supplies his certificate, the attacker intercepts this request and replaces the client certificate with his own and replace the HTTPS content in the server response with HTTP content signaling to the client that he will communicate over HTTP only. Thus, the client communicates the login information with the attacker in cleartext [53].

5.2 Threat's mitigation

In order to address the security risks mentioned above and address those attacks, a number of security measures are suggested to be considered by Cyber-Trust's DLT in order to enhance the overall security of the deployed solution.

5.2.1 Addressing the Insider Threat

Insider threat can be avoided in the cyber-trust project if the privacy of the peers (clients, peers and ordering nodes) is preserved. This can be achieved by using a preventive model against internal attacks specific to Hyperledger Fabric, such as the privacy-preserving data aggregation scheme proposed in [57] for the Smart Grids against internal adversaries. It is also important to secure the Member Service of Hyperledger, which is one of the critical components that Hyperledger Fabric provides to support dynamic entity registration and identity management, as well as auditing [56]. **This can be achieved by using modern Trusted Execution Environments (TEEs), such as Intel's Software Guard Extensions (SGX) [55].** This promising technology has been used in [56] as an extra security feature in peer communications. With SGX remote attestation and isolated execution inside the CPU capabilities, each distributed node can be enrolled as a trusted entity and hence greatly reducing the attack surface.

5.2.2 Addressing denial of service attacks

To mitigate this issue to a certain degree in the Cyber trust project, the OS should use Crash Fault Tolerant (CFT) protocols such as Kafka (based on Zookeeper), Raft, etc. These protocols are in principle vulnerable to Byzantine nodes: one single malicious node can effectively prevent the network from reaching consensus [47]. Therefore, a preventive model against internal attacks specific to Hyperledger fabric should be implemented along with these protocols. **Such a recommendation has also been made in Section 3, where Raft has been proposed as the consensus protocol of choice for CTB.**

Providing anonymity to endorsers can also prevent DdoS attacks against selective endorsers. For example, in [49] authors proposed an anonymity mechanism of endorsers using two different techniques: Anonymization of endorsers using pseudo identities and randomizing endorsers. This means that every user in the channel is equally probable to act as endorser for every new transaction. **DdoS attacks resulting from manipulating**

or stopping the Chaincode execution can be avoided by running the Chaincode on trusted execution environment such as SGX. This ensures that the chaincode does not deviate from its specification.

5.2.3 Addressing wormhole attacks

Wormhole attack is a new attack that applies to all major permissioned blockchain, including Hyperledger Fabric, and allows an attacker to steal information from honest peers in the same channel. Thus, a novel approach to secure the communications and preserve privacy within the channels is important. For example, work in [48] proposed to **anonymize the senders and recipients of transactions inside a channel**. Thus, the members of a channel can avail the benefits pertaining to that channel, but no member can view transaction communicated between two members within that channel. Group signature approach [54] is used to anonymize the actual identity of the sender, while the receiver is anonymized using the bilinear pairing-based cryptography [58]. This approach provides perfect unlinkability of public keys of users. While, on the other hand, the Backtracking approach (bitcoin) has been used to check the validity of a transaction. **Those approaches seem to be contradicting to the strong accountability needed by Cyber-Trust's DLT due to the need for integrity checking and storage of forensic evidence metadata, and hence need to further investigated.**

5.2.4 Addressing MitM & SSL stripping attacks

SSL stripping attacks intercept and decrypt confidential information sent over the network. This attack is relatively easy to launch and amongst the most dangerous Man in the Middle (MitM) attacks. An SSL certificate alone cannot protect against this attack. **Therefore, an intrusion detection system (IDS), like the one delivered in the context of the project, can be used to prevent this kind of attacks.** IDS will basically follow the network traffic closely. If an attacker tries to miss the traffic flow and intervene, IDS will prompt them instantly. Another solution to block all MitM attacks is to use a virtual private network (VPN). The use of such cryptographic tunnels creates additional secure layers when users access to the network over connections such as Wi-Fi [59]. A Strong client authentication approach can also prevent this kind of attacks like the multi-factor and multi-channel id authentication approach proposed by google [60].

5.3 Conclusions

As mentioned above, one of the most challenging issues that should be taken into account in the Cyber-Trust's DLT are insider threats. All Cyber-Trust's DLT nodes could be vulnerable to this threat, including the membership service provider (MSP) admins, ordering nodes, validating nodes and clients, which can cause great damage to Cyber-Trust's DLT network and lead to further attacks. Hence, the security of the Cyber-Trust DLT peers should be preserved by using a proactive model against internal adversaries, **with a secure identity management scheme as support**, to ensure that the data sources are credible rather than spurious or illegal. In this context, pseudo identities could be used to anonymize the senders and recipients of transactions inside a channel, and therefore hid the transaction communicated between two members of this channel from all other members. This will help to reduce the insider threat attack surface and creates additional security layer against DdoS and Wormhole attacks.

6. Privacy analysis of Cyber-Trust blockchain

In this section, we focus on privacy risks arising by the usage of the Hyperledger Fabric, towards presenting the appropriate mitigating measures that have been adopted in the Cyber-Trust system.

6.1 Privacy risks in blockchain applications

Despite the obvious advantages of the blockchain technologies, privacy issues are still a major challenge to be addressed, currently being an active research area. Such issues are mostly present in public blockchains, since in such a case citizens and companies are still wary to adopt such technologies, due to the publicity of their data/transactions [38]. However, even in private blockchains – which is the case of Cyber-Trust system – there exist privacy challenges, despite the fact that, obviously, the non-publicity of the ledger alleviates some issues by default. Several general privacy concerns, in relation with the relevant provisions of the General Data Protection Regulation (GDPR), are being stressed in D7.1 [67] and D7.2 [68] (although these deliverables cover the whole Cyber-Trust project, personal data protection aspects in the context of blockchain are also being considered therein).

In general, main privacy challenges in blockchain scenarios (of any form), can be summarized as follows (see also [39]):

i) **Transaction linkability**. This is the case in which data ("transactions" in the context of the blockchain) corresponding to one user can be linked, thus allowing development of a profiling of the user according to her/his data. For instance, this is an obvious threat in a public blockchain, in which the user corresponds to a unique address, which in turn constitutes somehow a pseudonym of the user[21].

   Note that, in general, the purpose of the personal data process occurring within the context of the blockchain should be taken into account in order to evaluate such a privacy threat. In the case of the Cyber-Trust framework, the relevant blockchain serves as vehicle to provide integrity and ownerships of requests between LEAs and ISPs when the first are asking to access forensic evidence – and, to this goal, metadata of these evidence are also placed in the blockchain (see D7.5). Therefore, some "transaction linkability" shall be allowed; for example, a complete backlog of communication is needed, to establish a chain of custody. Therefore, in the Cyber-Trust system, the transaction linkability privacy threat rests with the aim to ensure that only legitimate "users" (stakeholders) have access to these data that can be linked, and such an access should be confined to the absolutely necessary data with respect to the desired purpose (which in turn needs to be fully transparent); no further processing in a manner that is incompatible with these purposes should be allowed[22].

ii) **Private keys management**. Private keys are being used to sign each "transaction" in the blockchain and, thus, they constitute an important primitive for both security and privacy. Compromisation of private keys results in identity thefts, as well as in privacy issues.

iii) **"Malicious" smart contracts.** There is also a privacy challenge concerning the smart contracts that are able to access the data in order to process transactions. There is possibility that a smart contract might be able, either deliberately (due to a malicious programmer) or not, to leak information on what is being processed. In this way, privacy might be breached [40]. Such an example is being

[21] It is well-known in the literature that, even if the pseudonym do not allow by itself a re-identification of a user, it is still being considered as personal data since, under specific circumstances, it may allow identification (e.g. if the user provides additional information to verify that she/he is the owner of the pseudonym – see also D3.1 [65]). Moreover, linkage of information ("transactions"), corresponding to a single user, that is being available in the blockchain could also give rise to identification of a user or deriving conclusions for her/him far beyond the purpose of the processing (see also D7.1).

[22] See also the data protection principles of lawfulness, fairness and transparency of the purpose of the process (Art. 5(1) of the GDPR), as well as the purpose limitation principle (Art. 5(2) of the GDPR) – see also D3.1 [65].

described in a simple scenario of an “odds and evens” game between two players being described in [41], where it is shown that an adversary can carry on an attack which always allows him to win the game.

iv) **Non-erasable data**. Due to the inherent immutability property of blockchains, data that have been put into a blockchain cannot be deleted (and they remain always available to any user/stakeholder having access to the blockchain). This clearly gives rise to privacy issues, which are strongly related to the so-called right to erasure (or right to be forgotten) that is provisioned in the GDPR.

v) **Privacy interoperability across different blockchain-enabled scenarios**. As it is described in [38], a blockchain can be applied in different kind of scenarios, which demand diverse privacy requirements. Therefore, it is essential to have a fragmentation so as to ensure that diverse blockchain implementations that are not interoperable need to be in place.

Note also that loss of data confidentiality (i.e. unauthorized access to individual’s personal data) is also a privacy issue, but we refrain from explicitly stated it since such a risk has been already addressed above in the security context.

It should be stressed that several privacy enhancing techniques, based on advanced cryptographic primitives, are known for alleviating privacy challenges in blockchains; such possible solutions have been analyzed in D7.1 [67]. However, as recently stated in [38], the performance of such privacy enhancing technologies is still an issue that needs to be handled. Moreover, the authors in [38] explicitly state, as privacy challenge, the case that a blockchain is being used in an IoT scenario to provide several services (such a case is actually the Cyber-Trust system): the authors claim that, due to the limited capabilities of the devices, they might have difficulties to accomplish certain blockchain-related operations – and they actually classify this as a distinct privacy challenge by itself.

6.2 Privacy mechanisms of Hyperledger Fabric

The Hyperledger traffic supports two important privacy mechanisms. The first one rests with the so-called **channels** concept, which allows to separate the information between different channels [42]. The second one is the concept of the **private data,** which allows to isolate data between different organizations within the same channel [39]. The channels are associated with the network level, whereas the private data are associated with the chaincode (application) level. Such privacy mechanisms are very important in cases of providing blockchain network and applications into a consortium environment (where consortium is a number of organizations\parties with common business goals); this is actually the case of the Cyber-Trust system.

6.2.1 Channel

Channel is an important concept in Hyperledger Fabric. It is associated with a set of organizations sharing business goals and maintaining a same and consistent data store. As defined in [42], a channel is a state partition of the full system that (a) is autonomously managed by a set of peers (but it is still aware of the bigger system it belongs) and (b) optionally, hides the internal state from the rest of the system. The latter feature actually, when enabled, is crucial in terms of privacy.

A Channel is identified by a unique ID. After a peer joins a channel, a ledger is created and is running on the peer. When a peer joins more than one channel, these ledgers are running independently. This is shown in next Figure 6-1 [43]. By these means, a channel provides a mechanism to limit both applications and data to a specific set of entities.

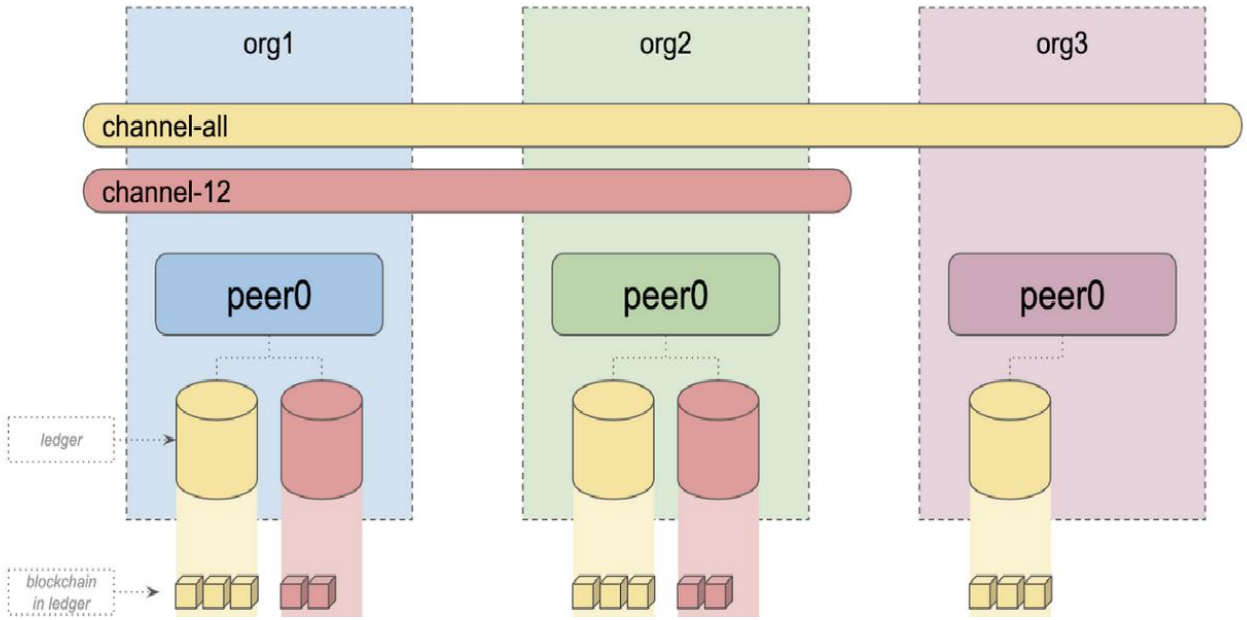

Figure 6-1: A scenario of two channels in a business network [43]

At application level, the chaincode is deployed at channel level. In case that a peer joins more than one channel, the correct ledger (uniquely identified by its ID) is accessed each time. It should be pointed out, as stated in [43] that we can deploy one or more chaincodes at channel level. In other words, there is flexibility in developing chaincode and application specific to a channel – whereas, of course, we still can deploy the same chaincode on different channels.

6.2.2 Private Data

Private Data provides privacy from a different perspective. More precisely, Private Data allow, for a specified set of organizations, that a subset of them keeps the actual data, while the remaining keep only a proof of such data, but not the actual data. This can be performed within a same channel – in other words, isolation of data occurs within a same channel [39].

The main benefit of Private Data occurs in the case that that we need to isolate data, but there is no privacy need at chaincode or application level. In such a case, a strictly channel approach will not be efficient since we would have to construct a separate channel for every subgroup of peers – that is a separate ledger for each subgroup, with the same set of chaincode being instantiated for each of them (and things become more complex if another channel in case that a new subgroup of peers is necessitated, since a new channel needs In many cases, an enterprise may need to keep certain information private in a channel from other companies additional administrations, policies, membership accesses, and many more) [43].

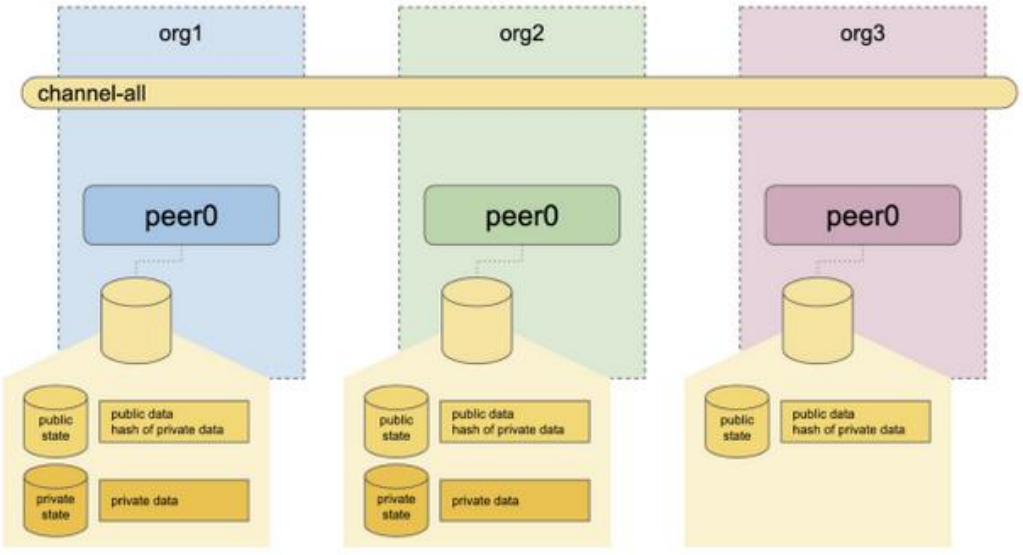

Figure 6-2: One channel with private data collection defined for org1 and org2 [43]

In Private Data, one can deploy multiple collections inside a channel, each of them serving different business objectives for a subgroup of organizations. Clear gains occur in terms of scalability. The above are shown in Figure 6-2.

6.2.2.1 Private Data Collection (PDC)

In the Hyperledger Fabric, The PDC (private data collection) is the term referring to collection of peers organizations that are authorized to store private data on a channel [39]. The data stored includes: i) Private data, which are stored in the peer's private database, ii) the hash value of the private data (see Figure 6-2). For private data, the peers on the channel use the hash value of the private data when sorting and writing the endorsement, as evidence of the existence of the transaction and for state validation and auditing. According to Hyperledger Fabric 2.0 documentation, PDC can be used when we need to keep only a portion of the ledger secret from a set of organizations.

For private data, the information that is broadcasted between peers uses the gossip protocol. However, only the authorized organization can see this. There are no ordering services in this case, and they can't see the private information. Anyhow, as the gossip protocols broadcast the information from one peer to another, Anchor nodes in the channel need to be in place (that is, according to the Hyperledger Fabric glossary, a peer node on a channel that all other peers can discover and communicate with. Each member (organization) on a channel has at least an anchor peer).

There exist cases where a member of a PDC wants to share private data with other organizations – for example, if the member wants to transfer the asset to a third party. In such a scenario, the third party can calculate the hash of the private data and subsequently check that the hash value is consistent with the hash stored on the channel ledger, thus proving the existence of the transaction.

Concluding, some organizations will have full access to the ledger, and others may only see what they are allowed to. In case that transactional data should remain hidden from the ordering services, usage of PDCs is a solution.

An Example. Let's check out an example from Hyperledger Fabric 2.0 documentation to understand the situation better[23]. Let us assume that, in a trading platform, there are five enterprises in a channel: 1) The

[23] This example is also available in https://101blockchains.com/hyperledger-fabric-2-0/

farmer who sells goods, 2) The Distributor who moves those goods, 3) The Shipper who moves goods between two parties, 4) The Wholesaler who buys goods from the Distributor, 5) The Retailer who buys the goods from the wholesalers and shippers.

We assume that the Distributor can charge differently in every case. So, he might want to keep transactions with the Shipper and Farmer private because he may have other deals with the Retailer and the Wholesaler. Moreover, the Distributor does charge less to a Wholesaler than to a Retailer. Thus, he may want to keep that a secret from the Retailer. The Wholesaler, on the other hand, can also have private relations with the Shipper and the Retailer.

A channel-based solution is an option, but creating a separate channel for every private information renders the system much complicated. Rather than doing that, we may have different PDCs for each of the members, as follows:

I. Private-Data-Collection-1: Shipper, Farmer, and Distributor

II. Private-Data-Collection-2: Shipper, Retailer, and Wholesaler

III. Private-Data-Collection-3: Wholesaler and Distributor

All the Distributor peers will have private databases containing private data for Shipper, Farmer, and Distributor relation and Wholesaler, and Distributor relation.

According to Hyperledger Fabric 2.0 documentation, there are some enhancements that actually make it possible for the new private data patterns to work.

First, the option of collection-Level Endorsement Policies is allowed. By default, endorsement policies are specified in the chaincode definition, which is agreed to by channel members and then committed to a channel (that is, one endorsement policy covers all of the state associated with a chaincode). For PDCs though, we can also specify an endorsement policy at the private data collection level, which would override the chaincode level endorsement policy for any keys in the private data collection, thereby further restricting which organizations can write to a private data collection.

Moreover, in Fabric 2.0, one can use implicit organization-specific collections without any upfront definition. If we want to utilize the per-organization private data patterns, we don't even have to define the PDCs when deploying chaincode; this is one of the major Hyperledger Fabric 2.0 features.

6.2.3 Channels vs. PDCs

As stated in [39], these two private mechanisms provide the following services: (1) New channels are needed when the entire transaction and ledger must be kept strictly confidential to the outside members of the channel, (2) when the transaction information and ledger need to be shared among some organizations, so as some of them need to be able to see all the transaction data and other organizations need to know the occurrence of this transaction to verify its authenticity, a private data collection (PDC) should be established. In addition, because private data is propagated through peer-to-peer rather than block, the privacy data collection is used when the transaction data must be confidential for the sorting service peer.

6.3 Mitigation of privacy risks in Cyber-Trust blockchain

In this section, privacy-enhancing measures in the Cyber-Trust DLT, to address the aforementioned privacy challenges, are being discussed. First, it should be explicitly stressed that the Cyber-Trust DLT is a permissioned private ledger (see also the relevant requirement explicitly stated in D3.3 [66]). Hence, by default, unauthorized users/peers cannot write or read the ledger. Only pre-determined legitimate entities (ISP providers/LEAs) are allowed to have such access – under appropriate safeguards as discussed below. Nobody else can have any access to the Cyber-Trust DLT. Therefore, privacy issues occurring explicitly in

public permissioned/permissionless ledgers (due to the fact that anybody can read and/or possibly write) are out of the question in our case.

In addition, information on suspicious network traffic or potential malicious activity (which will be used in the subsequent evidence analysis) is being stored only in the off-chain database, and not in the DLT. Only logging information of any such activity, so as to provide a proof-of-existence service, will be stored in the DLT. Therefore, having access only to the DLT and not to the – securely stored – off-chain database, does not allow identification of network/device communication data (which are being considered as personal data). We subsequently analyze the main privacy challenges described in Section 6.1:

1) Transaction linkability. This risk, as stated above, is related with the possibility of linking/combining information corresponding to an individual, so as to create a profile of her\him (which could possibly be used for other purposes, without informing the user).

In the Cyber-Trust scenario, we define three confidentiality levels (see also D7.5):

i) The public information which is available to any partner of the Cyber-Trust system, related with trusted file storage (i.e. info on patches that need to be installed, their valid/genuine sources etc.) This information does not contain personal data and, thus, no privacy risks occur.

ii) Private information that can't be shared between partners. It corresponds to the data responsible of the ownership management – i.e. information on who owns which device, produced by which company as well as its internet provider. This information determines if the request for modification on a device by another person than the owner of the device can be executed based on the authorization level of the requester. This information is generated by the Cyber-Trust administrative service [ref to D4.1, D4.5] and is not accessible by any other entity.

iii) Private information that can be shared between partners. It concerns metadata referring to forensic evidence and other information useful to LEAs during criminal proceedings. At their creation, metadata is private and only visible to the creator of the data. Moreover, private information stores the communications between two partners; such a storage serves as a complete backlog of the communications (.txt and .pdf files).

For both cases ii and iii, the PDC feature of Hyperledger Fabric (see Section 6.2.2.1) is being utilized so as to render data private (case ii) or available only to a specific set of stakeholders upon request (case iii). Regarding the evidence data, we recall that the DLT stores only metadata, which are useless (meaningless) without the relevant off-chain data. Such access to off-chain data, in conjunction with the access to the corresponding on-chain metadata, will be provided only after a successful LEA's request, under the fulfillment of all the relevant legal requirements per jurisdiction. Regarding the communications between partners, such data will accessible only if there is need to check the history of communications and such need is absolutely necessary and fully justified (i.e. for establishing a chain of custody, so as to verify all previous communications between organizations/partners). In such a scenario, access will be restrictively given to the parties that should be allowed to get this information (and this granting of access is also recorded to the DLT, for further strengthening the transparency).

Therefore, from the above it becomes evident that the transaction linkability as a privacy risk does not hold in the Cyber-Trust DLT. The DLT by itself does not contain any personal data (e.g. personal/device identifiers etc.) that could allow the extraction of profiling of an individual[24].

[24] We should recall that the aim of the Cyber-Trust system is to provide security to individuals and, to this end, monitoring of suspicious network activities and verifying the patching of the device are in place. Although these pieces of information could be considered as "profile generation", this process lies exactly in the legitimate purpose of the system and is fully transparent to the user who provides her/his free consent for this. Therefore, this process should not be considered as a privacy issue.

2. **Private keys management.** Public key cryptography is being used in the Cyber-Trust DLT for digital signing the data that are being stored. More precisely, a Membership Service Provider (MSP) is responsible for maintaining the identity of all the participants in the system. It issues credentials in the form of cryptographic certificates which are used for the purpose of authentication and authorization. *Fabric CA (Certification Authority)* acts as a private root CA provider capable of generating keys and certificates. For example, the logger utilizes a digital signature algorithm to sign the metadata of the evidence information that are stored in the DLT; to achieve this, a digital certificate is being used, used by the certificate authority (CA) of HyperLedger Fabric, which is under the sole control, in our case, of the Cyber-Trust system. By these means, only legitimate users (i.e. users with valid certificates, issued by the Cyber-trust system) may interact ("write") to the DLT. Each user uses his own private key to transact.

   Note that in the Hyperledger Fabric, as also stated above (see also previous deliverables – e.g. D7.2 [68]), endorsers have a crucial role. A user (client) proposes a transaction to endorsing peers who in turn simulate the transaction and validate with the smart contract available. This endorsement is called execution. Once the execution is done at the endorsing peers the response would be sent back to the client with endorsement signatures (in case of private data, the endorser disseminates them to the authorized peer via the gossip protocol). Since endorsers are under the full control of the Cyber-Trust system, their signatures cannot be forged.

   A crucial issue though is how to protect the private keys of users outside the Cyber-Trust system (i.e. peers such as at LEAs). First, it should be noted that private keys are not being stored in any database. Moreover, it should be pointed out that as in any PKI-based system, customers are responsible for the storage, backup, and disaster recovery of all their keys. This will be explicitly clarified to any client/organization that receives a digital certificate. Moreover, the Cyber-Trust system will have in place a mechanism for certificate revocation and suspension, in case that something "strange" in the incoming transactions is being detected. Besides, the nature of the data that is being stored in the Cyber-Trust DLT allows an early identification of any such malicious request\transaction. In the unexpected scenario that a malicious transaction has managed to be put into the DLT, due to a stolen private key of a client, additional appropriate information will be subsequently put to the ledger so as to render all malicious transactions as fake.

3. **"Malicious" smart contracts.** The execute-order paradigm of Hyperledger Fabric separates the transaction flow (that might be executed by different nodes of the system) into the next steps. Firstly, in the execution phase, the execution of the transactions is taking place by checking their accuracy by a subset of the network, called endorsers. Then, in the ordering phase, the ordering service nodes (OSN) order all the transactions including the signatures of the endorsing peers. Last, in the validation phase the transactions are validated to prevent race conditions events.

   Smart contracts (also called as chaincode in the Hyperledger context) are computer programs that contain logic to execute transactions and modify the state of the assets stored within the ledger. Chaincode runs in a secured Docker container isolated from the endorsing peer process. Chaincode initializes and manages the ledger state through transactions submitted by applications.

   In the Cyber-Trust system, we used the Go language to create chaincode. Due to their importance, chaincode has been meticulously developed and tested (see D7.3). Tests illustrated that it responds correctly to any case that we could imagine, where there is no any privacy breach.

4. **Non-erasable data**. As also stated above, this issue is still a great challenge in the blockchain research community, as well as a debate topic in a public discussion. First note that it may be possible that the right to erase personal data (i.e. the right to be forgotten – see also D3.1 [65] and D3.3 [66]) may not be applicable in some cases; for example, the Directive Directive (EU) 2016/680 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data by competent authorities for the purposes of the prevention, investigation, detection or prosecution of criminal offences or execution of criminal penalties, states that (see art. 16)

> "*Member States shall provide for the controller to inform the data subject in writing of any refusal of rectification or erasure of personal data (…) and the reasons for the refusal. Member States may adopt legislative measures restricting, wholly or partly, the obligation to provide such information to the extent that such a restriction constitutes a necessary and proportionate measure in a democratic society with due regard for the fundamental rights and legitimate interests of the natural person concerned (…)*".

This has been also highlighted in D7.5, which states that the data subjects' rights with respect to personal data contained in a judicial decision or record or case file may be carried out in accordance with national law, which may differ from state to state.

In any case though, since there exist cases in which the right to personal data erasure will be applicable, the adopted approach in the Cyber-Trust system proceeds up to the greatest extent, given the inherent immutability of the ledger. More precisely, we may recall that evidence data (which are of the highest importance in terms of privacy) are not being stored in the DLT but in an off-chain database. Hence, deletion from this database is feasible. The DLT will contain only the metadata (hashed value etc.) which are practically irreversible and meaningless. In addition, for any such deletion, the authorized entity will create a new transaction, indicating that these old meaningless metadata correspond to evidence that has been deleted. In general, this will be the case for any deletion of data in the blockchain.

As a concluding remark on the data erasure in blockchains in general (including the case of Cyber-Trust), we refer to [44] stating that, from a legal point of view, and taking into account the technical limitations, there exist

> "indications that the obligation inherent to Article 17 GDPR (i.e. right to erasure) does not have to be interpreted as requiring the outright destruction of data. In Google Spain, the delisting of information from research results was considered to amount to erasure (…) This may be taken as an indication that what the GDPR requires is that the obligation resting on data controllers is to do all they can to secure a result as close as possible to the destruction of their data within the limits of their own factual possibilities.".

5. **Privacy interoperability across different blockchain-enabled scenarios.** This is the case in which the concepts of channels and private data supported by the Hyperledger Fabric (see Section 6.2) are quite beneficial. By appropriately choosing each time the number of channels, the organizations that will have access to each channel, as well as the PDCs inside each channel (with appropriate owners), all possible scenarios can be fully adequately handled, without posing privacy at risk.

6.4 Conclusions on privacy risks – Next research steps

The above analysis indicates that appropriate safeguards are in place to mitigate privacy risks in the context of the Cyber-Trust blockchain. To this end, we next recall the basic privacy requirements stated in D3.3 [66], in order to reveal that they have been addressed.

- Only authorized entities have access to the metadata stored in the DLT, the visualization tool and the off-chain database where the actual material is kept.
- The actual personal data (regarding network/device activities) are being stored in an off-chain database and not in the DLT; The DLT stores hashed values, which point to actual data stored on the off-chain conventional database.
- The data stored on the DLT and the off-chain database are safeguarded against internal or external security threats.
- The CTB and the off-chain solution accompanying it should follow the common principles with regards to the storage of electronic evidence, as described in D3.2.

- A private and permissioned solution for the DLT is being used.
- The status of the participants in the DLT will be clarified well in advance.

Research in this field is still ongoing. Some other possible approaches that could be appropriately adopted in the (near) future are the following:

a) Usage of appropriate Hardware Security Modules to manage private keys may further enhance the protection of the private keys of the organizations/entities that interact with the Cyber-Trust DLT[25].

b) Zero-knowledge proofs (ZKPs) constitute an important cryptographic primitive for enhancing privacy, including the blockchain applications (see e.g. D7.1 [67]). In the context of the Hyperledger Fabric, two privacy aspects will be achieved using ZKPs: Anonymous client authentication with Identity Mixer, and privacy-preserving exchange of assets with Zero-Knowledge Asset Transfer (ZKAT) [45].

   i. Identity Mixer leverages ZKP to offer anonymous authentication for clients in their transactions. ZKP protocols may have important role in case that the actual identity of the client – and possibly any attributes associated with – should remain secret from the rest of network entities, such as its peers. For example, this would be important in cases that these entities wish to verify that the creator of a transaction is a member of a particular organization (known as a "membership proof"), or that it is in possession of a specific set of attributes (known as "selective disclosure of attributes") but not the exact identity. In both cases, the protocols guarantee that nothing is revealed about the client's identity.

   ii. The Hyperledger Fabric community puts efforts on the use of ZKP to accommodate privacy-preserving asset management with audit support (also known as Zero-Knowledge Asset Transfer, or ZKAT). By this feature, clients (transactors) will allow to issue assets and request transfer of their assets without revealing anything to the ledger for the assets being exchanged beyond the fact that the transfer complies with the asset management rules. ZKAT is built on top of anonymous authentication mechanisms offered by Identity Mixer.

   Although it is not yet obvious that, in the Cyber-trust system, such privacy features are desirable (since transparency in the context of the desired security services is essential), further considerations of ZKPs should be investigated.

c) A promising technology to efficiently implement the "right to erasure" in blockchains is the so-called notion of redactable blockchains (see, e.g., [46]). Such blockchains are able to remove data or to re-write data. The main way to achieve this is the usage of the so-called chameleon hash functions, that is hash functions for which a collision can be found given a secret trapdoor information. Hence, this may be the right path to implement the right to erasure in practice (this is also mentioned, as a research alternative that needs to be further taken into account, in [44]).

d) Implementing post-quantum digital signatures in a blockchain is also a very important future research task. This issue – and possible relevant technologies – has been widely discussed in D7.1 [67]. At the moment, the Hyperledger Fabric as a technology does not support post-quantum signatures.

[25] Similarly to the approach described in https://cloud.ibm.com/docs/blockchain?topic=blockchain-ibp-security#ibp-security-ibp-keys

7. Conclusions

The security threats of Cyber-Trust's DLT solution are analyzed by the present deliverable are classified into a number of areas: (a) consensus security; (b) smart contract security; (c) software and network security; and (d) privacy aspects. A summarization of the key findings is provided below.

Consensus security. Since the implementation of the CTB consensus protocol is among the most important components of a DLT solution, the combination of Fabric with a *CFT protocol (and in particular Raft)* is considered to be ideal for Cyber-Trust's permissioned blockchain environment optimally balancing between security and the required performance. Future extensions might consider BFT protocols, like BFT-SMaRt, once they are adopted by the Hyperledger project.

Smart contract security. A number of measures *have already been implemented* to harden CTB solution as mentioned in Section 4.2; however, a recent trend in analyzing smart contracts' security has been identified that is based on *formalization techniques*, like symbolic execution, formal modelling, model checking, etc., which could also be used to properly analyze other properties such as scalability, performance, and privacy. This approach seems to be quite promising and could be examined as future research work.

Software and network security. Representative attacks were considered against the CTB solution, namely insider threats, (distributed) denial of service attacks, wormhole attacks, man-in-the-middle attacks, and SSL stripping attacks. In general, proactive solutions having been chosen, such as *the use of BFT/CFT protocols in CTB* (see above), have been identified as a possible mitigation to insider threats and misbehaving nodes. An alternative/complementary option would be to *employ a trusted execution environment, such as Intel's SGX*, to address insider threats and also DDoS attacks resulting from the manipulation or stopping of the chaincode execution. This is an interesting direction that could be explored in a future research project. Techniques to mitigate wormhole attacks vary from those relying on the anonymization of the senders and recipients in the transactions inside channels, to those employing group signatures. Due to the need for accountability in CTB, such solutions need to be further assessed possibly along with privacy solutions (like ZKAT). Finally, an IDS solution, combined with strict policies in certificate management, are found to be sufficient for detecting MitM and SSL stripping attacks against the project's DLT solution.

Privacy aspects. Appropriate safeguards implementing the basic privacy requirements, stated in D3.3 [66], are already in place to mitigate privacy risks in the context of CTB. As research in this field is rapidly evolving, other promising approaches have been identified that could be adopted in a (near) future research effort. These span the use of *hardware security modules* (e.g. those having also been identified above to mitigate other threats), the *employment of ZKPs in the context of the Hyperledger Fabric* to achieve anonymous client authentication with identity mixer, and privacy-preserving exchange of assets with ZKAT, and the exploration of a new promising technology, referred to as *redactable blockchains*, to efficiently implement the "right to erasure" in CTB, and finally the adoption of post-quantum digital signatures (to be further investigated in a forthcoming deliverable).

8. References

[1] Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., … & Muralidharan, S. (2018, April). Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference* (p. 30). ACM.

[2] Kiayias, Aggelos, et al. "Ouroboros: A provably secure proof-of-stake blockchain protocol." *Annual International Cryptology Conference*. Springer, Cham, 2017.

[3] Daian, Phil, Rafael Pass, and Elaine Shi. "Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake." *International Conference on Financial Cryptography and Data Security*. Springer, Cham, 2019.

[4] Hadzilacos, Vassos, and Sam Toueg. "Fault-tolerant broadcasts and related problems." *Distributed systems (2nd Ed.)*. 1993. 97-145.

[5] Kreps, Jay, Neha Narkhede, and Jun Rao. "Kafka: A distributed messaging system for log processing." *Proceedings of the NetDB*. Vol. 11. 2011.

[6] D. Ongaro. The Raft consensus website, Nov. 2014. http://raft.github.io/.

[7] Ongaro, Diego, and John Ousterhout. "In search of an understandable consensus algorithm." *2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14)*. 2014.

[8] Oki, Brian M., and Barbara H. Liskov. "Viewstamped replication: A new primary copy method to support highly-available distributed systems." *Proceedings of the seventh annual ACM Symposium on Principles of distributed computing*. 1988.

[9] Christopher Ferris. (2019): "Does Hyperledger Fabric perform at scale?". Available at: https://www.ibm.com/blogs/blockchain/2019/04/does-hyperledger-fabric-perform-at-scale/

[10] Bessani, Alysson, João Sousa, and Eduardo EP Alchieri. "State machine replication for the masses with BFT-SmaRt." *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 2014.

[11] Sousa, Joao, Alysson Bessani, and Marko Vukolic. "A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform." *2018 48th annual IEEE/IFIP international conference on dependable systems and networks (DSN)*. IEEE, 2018.

[12] Hyperledger Fabric documentation: Available at: https://hyperledger-fabric.readthedocs.io/en/release-2.0/orderer/ordering_service.html

[13] Altman, Eitan. *Constrained Markov decision processes*. Vol. 7. CRC Press, 1999.

[14] Yamashita, Kazuhiro, et al. "Potential risks of hyperledger fabric smart contracts." *2019 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*. IEEE, 2019.

[15] Küsters, Ralf, Daniel Rausch, and Mike Simon. "Accountability in a Permissioned Blockchain: Formal Analysis of Hyperledger Fabric (Full Version)."

[16] Vukolić, Marko. "Rethinking permissioned blockchains." *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*. 2017.

[17] Brewer, Eric A. "Towards robust distributed systems." *PODC*. Vol. 7. 2000.

[18] Gaur, Nitin, et al. *Hands-On Blockchain with Hyperledger: Building decentralized applications with Hyperledger Fabric and Composer*. Packt Publishing Ltd, 2018.

[19] Garay, Juan, Aggelos Kiayias, and Nikos Leonardos. "The bitcoin backbone protocol: Analysis and applications." *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, Berlin, Heidelberg, 2015.

[20] Pass, Rafael, Lior Seeman, and Abhi Shelat. "Analysis of the blockchain protocol in asynchronous networks." *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, Cham, 2017.

[21] Cristian, Flaviu, et al. “Atomic broadcast: From simple message diffusion to Byzantine agreement.” *Information and Computation* 118.1 (1995): 158-179.

[22] Alkhalifah, Ayman, et al. “A taxonomy of blockchain threats and vulnerabilities.” (2019).

[23] Natarajan, Harish, Solvej Karla Krause, and Helen Luskin Gradstein. “Distributed Ledger Technology (DLT) and blockchain. FinTech note; no. 1. Washington, DC: World Bank Group.” (2019).

[24] Wan, Zhiyuan, et al. “Bug characteristics in blockchain systems: a large-scale empirical study.” *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. IEEE, 2017.

[25] Deirmentzoglou, Evangelos, Georgios Papakyriakopoulos, and Constantinos Patsakis. “A survey on long-range attacks for proof of stake protocols.” *IEEE Access* 7 (2019): 28712-28725.

[26] Eyal, Ittay. “The miner’s dilemma.” *2015 IEEE Symposium on Security and Privacy*. IEEE, 2015.

[27] Bahack, Lear. “Theoretical Bitcoin Attacks with less than Half of the Computational Power (draft).” *arXiv preprint arXiv:1312.7013* (2013).

[28] Rosenfeld, Meni. “Analysis of bitcoin pooled mining reward systems.” *arXiv preprint arXiv:1112.4980* (2011).

[29] Eyal, Ittay, and Emin Gün Sirer. “Majority is not enough: Bitcoin mining is vulnerable.” *International conference on financial cryptography and data security*. Springer, Berlin, Heidelberg, 2014.

[30] Saad, Muhammad, et al. “Partitioning attacks on bitcoin: colliding space, time, and logic.” *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2019.

[31] Apostolaki, Maria, Aviv Zohar, and Laurent Vanbever. “Hijacking bitcoin: Routing attacks on cryptocurrencies.” *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017.

[32] Hallman, Roger, et al. “IoDDoS-the internet of distributed denial of sevice attacks.” *2nd international conference on internet of things, big data and security. SCITEPRESS*. 2017.

[33] Conti, Mauro, et al. “A survey on security and privacy issues of bitcoin.” *IEEE Communications Surveys & Tutorials* 20.4 (2018): 3416-3452.

[34] Douceur, John R. “The sybil attack.” *International workshop on peer-to-peer systems*. Springer, Berlin, Heidelberg, 2002.

[35] Conti, Mauro, et al. “A survey on security and privacy issues of bitcoin.” *IEEE Communications Surveys & Tutorials* 20.4 (2018): 3416-3452.

[36] Miller, Andrew, et al. “The honey badger of BFT protocols.” *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 2016.

[37] Turki, Haithem, Fidel Salgado, and Juan Manuel Camacho. “HoneyLedgerBFT: Enabling Byzantine fault tolerance for the Hyperledger platform.”

[38] Bernabe, J. B., Canovas, S. J. L., Hernandez-Ramos, J. L., Moreno, R. T., Skarmeta, A. (2019): Privacy-preserving solutions for Blockchain: a systematic review and challenges. IEEE Access, vol. 7, no. 1 (pp. 164908-164940). IEEE

[39] Ma, C., Kong, X., Lan, Q., Zhou, Z. (2019): The privacy protection mechanism of Hyperledger Fabric and its application in supply chain finance. Cybersecurity, vol. 2, no. 1, Springer.

[40] European Union Agency for Cybersecurity (2016): Distributed Ledger Technology & Cybersecurity – Improving information security in the financial sector. Dec. 2016.

[41] Atzei, N., Bartoletti, M. and Cimoli,C. (2016): A survey of attacks on Ethereum smart contracts. Cryptology ePrint Archive, Report 2016/1007.

[42] Androulaki, E., Cachin, C., De Caro, A., Kokoris-Kogias, E. (2018): Channels: Horizontal scaling and confidentiality on permissioned blockchains. ESORICS 2018, LNCS 11098, pp. 111-131, Springer.

[43] Tam, KC (2019): Data Privacy among Organizations: Channel and Private Data in Hyperledger Fabric. Available in https://medium.com/@kctheservant/data-privacy-among-organizations-channel-and-private-data-in-hyperledger-fabric-ee268cd44916

[44] European Parliamentary Research Service (2019): Blockchain and the General Data Protection Regulation: Can the distributed ledgers be squared with European Data Protection Law?

[45] E. Androulaki (2018): Private and confidential transactions with Hyperledger Fabric. Available in https://developer.ibm.com/technologies/blockchain/tutorials/cl-blockchain-private-confidential-transactions-hyperledger-fabric-zero-knowledge-proof/

[46] G. Ateniese, B. Magri, D. Venturi, E. Andrade (2017): Redactable Blockchain – or – Rewriting History in Bitcoin and Friends. IEEE European Symposium on Security and Privacy, pp. 111-126.

[47] Dabholkar, Ahaan et Saraswat, Vishal (2019). Ripping the Fabric: Attacks and Mitigations on Hyperledger Fabric. In: International Conference on Applications and Techniques in Information Security. Springer, Singapore, p. 300-311.

[48] Andola, N., Gogoi, M., Venkatesan, S., & Verma, S. (2019). Vulnerabilities on hyperledger fabric. Pervasive and Mobile Computing, 59, 101050.

[49] Davenport, A., Shetty, S., & Liang, X. (2018). Attack surface analysis of permissioned blockchain platforms for smart cities. In 2018 IEEE International Smart Cities Conference (ISC2) (pp. 1-6). IEEE.

[50] Kolias, C., Kambourakis, G., Stavrou, A., & Voas, J. (2017). DdoS in the IoT: Mirai and other botnets. Computer, 50(7), 80-84.

[51] Brandenburger, M., Cachin, C., Kapitza, R., & Sorniotti, A. (2018). Blockchain and trusted computing: Problems, pitfalls, and a solution for hyperledger fabric. arXiv preprint arXiv:1805.08541.

[52] Baliga, A., Solanki, N., Verekar, S., Pednekar, A., Kamat, P., & Chatterjee, S. (2018, June). Performance characterization of hyperledger fabric. In 2018 Crypto Valley Conference on Blockchain Technology (CVCBT) (pp. 65-74). IEEE.

[53] El-Hajj, W. (2012). The most recent SSL security attacks: origins, implementation, evaluation, and suggested countermeasures. Security and Communication Networks, 5(1), 113-124.

[54] S. Han, J. Wang, W. Liu, An efficient identity-based group signature scheme over elliptic curves, in: Universal Multiservice Networks, Springer, Berlin, Heidelberg, 2004, pp. 417–429.

[55] Costan V, Devadas S. Intel SGX Explained. IACR Cryptology ePrint Archive. 2016 Feb;2016(086):1-18

[56] Liang X, Shetty S, Tosh D, Foytik P, Zhang L. Towards a Trusted and Privacy Preserving Membership Service in Distributed Ledger Using Intel Software Guard Extensions. In International Conference on Information and Communications Security 2017 Dec 6 (pp. 304-310). Springer, Cham.

[57] HE, Debiao, KUMAR, Neeraj, et LEE, Jong-Hyouk. Privacy-preserving data aggregation scheme against internal attackers in smart grids. Wireless Networks, 2016, vol. 22, no 2, p. 491-502.

[58] https://repositorium.sdum.uminho.pt/bitstream/1822/3813/1/report.pdf

[59] Tekdoğan, R., & Efe, A. (2018). Prevention Techniques for SSL Hacking Threats to E-Government Services. Ankara: International Journal of Information Security Sciences.

[60] Xiong, Chuyu. "Multi-factor and multi-channel id authentication and transaction control." U.S. Patent Application 13/229,219, filed March 15, 2012.

[61] L. Chen, L. Xu, Z. Gao, Y. Lu, and W. Shi (2018). "Tyranny of the majority: on the (im)possibility of correctness of smart contracts". IEEE Security & Privacy, vol. 16, no. 4, pp. 30–37, July/August.

[62] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen (2017). "A survey on the security of blockchain systems," Future Generation Computer Systems, pp. –. Doi: 10.1016/j.future.2017.08.020.

[63] N. Atzei, M. Bartoletti, T. Cimoli, A survey of attacks on 52thereum smart contracts (sok), in: International Conference on Principles of Security and Trust, 2017, pp. 164–186.

[64] H. Hasanova, U. Baek, M. Shin, K, Cho, and M. Kim (2019). "A survey on blockchain cybersecurity vulnerabilities and possible countermeasures" *International Journal of Network Management*, pp. 1-36. [Online] Available at: DOI: 10.1002/nem.2060

[65] O. Gkotsopoulou and P. Quinn (eds.), "Regulatory framework analysis," *Cyber-Trust Consortium*, Deliverable D3.1, 2018.

[66] O. Gkotsopoulou and P. Quinn (eds.), "Legal and ethical recommendations," *Cyber-Trust Consortium*, Deliverable D3.3, 2018.

[67] P. Gerard, *et al.* (eds.), "Distributed ledger state-of-the-art report," *Cyber-Trust Consortium*, Deliverable D7.1, 2019.

[68] C. Pavué, *et al.* (eds.), "Cyber-Trust distributed ledger architecture," *Cyber-Trust Consortium*, Deliverable D7.2, 2019.

[69] C. Pavué (eds.), "Cyber-Trust information and evidence storage," *Cyber-Trust Consortium*, Deliverable D7.5, 2020.

[70] A. Singh et al., "Blockchain smart contracts formalization: Approaches and challenges to address vulnerabilities," Computers & Security, 88 (2020) 101654, pp. 1-16.

[71] Chen T, Li X, Luo X, Zhang X. Under-optimized smart contracts devour your money. In: Software Analysis, Evolution and Reengineering (SANER), 2017 IEEE 24th International Conference on. IEEE; Feb 2017, pp. 442-446.

[72] B. Beckert, M. Herda, M. Kirsten, J. Schiffl, "Formal specification and verification of Hyperledger Fabric chaincode," 2018

