BCaR: Blockchain in Car Registration

Tiago Rosado

Instituto Superior Técnico, Lisbon, Portugal,
tiagocrosado@tecnico.ulisboa.pt,
WWW home page: http://tcrosado.github.io

Abstract. Bitcoin and other cryptocurrencies have taken a great importance as revolutionary alternatives to centrally managed currency. Apart from political and financial aspects of this new good, its underlying blockchain technology is truly innovative and presents a set of properties which can not be overlooked. Blockchain technology enables the development of new business models based on decentralization but centralized organizations, as governments, can benefit from this technology to improving safety and security of critical data. In this document we present an overview of existing blockchain infrastructures, from Bitcoin to Hyperledger, and provide some example applications already using blockchain technology. For understanding the benefits and implications of a blockchain technology application in a government's critical information system, we propose a solution, based on Hyperledger and BFT-SMaRt as a consensus algorithm, to implement a car registration system and provide a method to compare it with a currently implemented centralized system for car registration.

Keywords: blockchain, car registration, critical data storage, distributed ledger

1 Introduction

Blockchain technology has reached the mainstream by numerous criptocurrencies who implement this technology and shook the concepts of currency and how the financial system works. Bitcoin, as the most known application of blockchain technology, implements a digital currency not relying on any central authority to be managed [16]. The decentralization provided by blockchain technology can be considered a direct competitor to organizations relying on a centralized business model, such as banks and governments. Considering an extreme situation, blockchain technology could present a decentralized collection of services competing with government's services such as property registration, citizen registration and even a financial system replacement that could render most of the government's work useless.

However, centralized organizations can also look at blockchain technology as an innovation with the ability to improve their efficiency. Currently some of these organizations are investigating and implementing blockchain technology to the benefit of business efficiency and government's transparency. One of such

example is the Estonian government who implemented some of its services using this technology around 2012 [23].

1.1 Objectives

Considering the potential of blockchain technology, the main objective of this work is to implement a car registration infrastructure based on blockchain and compare it to the current system implemented by the Portuguese government.

A car registration system based on blockchain can decentralize this kind of registries and as a consequence improve data availability and resilience to faults. As a set of entities, ranging from leasing companies to government offices, rely on the car registration system, a decentralized system based on blockchain can improve its performance and safety when compared with a centralized solution. Given the decentralization inherent to blockchain it is crucial to understand the role of an entity such as the national registry. As it will be discussed throughout this document, a blockchain application for car registration can still take in consideration the authority of the national registry entity. With this requirement, a blockchain based car registration system can benefit from decentralization but still maintain a centralized authority to partially manage and control the system.

This technology can also represent an effort to improve government efficiency and transparency [23]. As blockchain technology tends to decentralize data storage, it is also necessary to weight the effort needed to implement the system in a full scale and identify possible struggles over this approach, ranging from the initial setup of the system to maintenance effort towards needed hardware, including possible software updates on the system. Therefore we propose to compare an implementation of blockchain technology for car registration using smart contracts with the currently centralized system implemented.

1.2 Organization of the document

In this document we will start by analysing the existing blockchain infrastructures such as Bitcoin [16] and Ethereum [9] and we later introduce the use of smart contracts. As an evolution of initial consensus mechanisms, such as proof-of-work, we introduce Bizantine Fault Tolerant (BFT) based blockchains and compare them considering the properties of fault tolerance and consensus of each one. Later we introduce Tendermint [7] as a BFT based blockchain and identify differences between permissioned and permissionless blockchains. Related to permissioned blockchain we introduce the concept of smart contract execution separation [22] and analyse Hyperledger Fabric (HLF) together with some changes suggested to this infrastructure [22, 17]. Finishing related work we compare each infrastructure in terms of performance, scalability and fault tolerance. In Section 3 we propose a solution based on HLF and some modifications mentioned in the related work section [22, 17]. We then propose an evaluation methodology and present a work schedule to develop the thesis.

2 Related Work

In this section we will go over a set of blockchain infrastructures and applications available. We will start by introducing Bitcoin as the first application based on blockchain technology and how this technology solves some of the problems behind the creation of a digital currency. In relation to blockchain we introduce the problem of consensus, how Bitcoin reaches network consensus and other methods of consensus that should be considered, such as proof-of-stake. Next we present Ethereum as an infrastructure enabling the use of smart contracts and the development of decentralized applications. We define what are smart contracts and how Ethereum handles their execution. Later we introduce the use of BFT based consensus in blockchain technology, comparing them with the proof-of-work approach. As an application of a BFT consensus algorithm to the blockchain technology we introduce Tendermint and explain the advantages of this infrastructure. Also with BFT consensus mechanisms we present Hyperledger Fabric and later define the differences between permissioned and permissionless blockchains as well as the differences between private and public blockchains. Finally we mention some applications currently using blockchain technology, how they are configured and compare the various infrastructures mentioned in this section.

2.1 Bitcoin

Bitcoin is the first decentralized digital currency and works based on blockchain technology, not relying on any central authority to manage currency. Bitcoin was introduced by Satoshi Nakamoto [16] and presents a peer-to-peer network where transactions are executed between users without any intermediary. Bitcoin's transactions are verified by network peers and recorded in the blockchain.

A blockchain is a distributed ledger that works as a collection of transactions grouped in blocks and those blocks are chained to each other. Data within the blocks can not be modified and changes or new data must be appended to the chain. This provides an immutable log of every transaction ever made in the blockchain. A distributed ledger is a database that can be shared in a network by multiple locations, as each network participant has a local copy of the database [23].

The way blockchain works also solves a big problem about digital currency: double spending. Double spending is an attack on which the same set of coins is spent in more than one transaction.

Consensus In order for the blockchain network to agree on the next block to add to the blockchain there is a need to decide who should be the author of the block. Bitcoin uses a proof-of-work to decide this matter [16], but as this is a computational intensive task there are alternatives such as proof-of-stake. This methods are both used in order to ensure the integrity of the blockchain. Given the concurrency between nodes it is possible that two nodes can broadcast

different verisons of the next block at the same time. In this situation, other nodes start working over the first block they reveive but still save the other branch, this represents a fork. A fork is removed once the next proof-of-work is found. As the new block is added to one of the branches, the branch becomes longer and the shorter branch is removed [16].

Proof-of-work A proof-of-work is hard to produce and easy to verify, created by nodes in the blockchain network. This implies the need for a computational intensive problem to be solved. The node responsible for solving the problem is the potential author of the next block to be added to the blockchain [24]. In case of Bitcoin, a node needs to scan for a nonce value for a block so that the hash of the block starts with a defined number of zero bits. The computational effort to solve this problem is exponential to the number of zero bits required [16].

This type of proof has however the potential side effect of slowing the transaction processing within the network as major computational work is wasted in the proof-of-work to create a block instead of transaction processing. The approach used by proof-of-work increases the difficulty of successful attacks, given the need for a great computational power to create a proof-of-work and use it to tamper blockchain data.

Proof-of-stake As the proof-of-work solution is not resource efficient, an alternative is proof-of-stake. Proof-of-stake aims to reach network consensus, therefore determine the creator of the next block, based on a combination of factors, as the account balance or age of the node.

Peercoin [14] presents an example on a proof-of-stake based cryptocurrency. In Peercoin, if a node receives for example 10 coins and does not transaction them for 10 days, it will have 100 coin-days, which is the age of the node. To determine the creator of the next block, Peercoin requires nodes to pay themselves, consuming their coin age in order to generate a block [14]. This process is executed such that the coin age is consumed to achieve certain hash target and the hash target is easier to meet the more coin age is consumed. Therefore if a node with 100 coin-years meets the hash target in 2 days, a node with twice the coin-years meets the hash target in 1 day.

Properties of blockchain Blockchain technology presents a set of important properties to ensure data integrity, availability, authenticity and fault tolerance. In terms of integrity and authenticity, the system enables a granular access control in conjunction with privacy and transparency.

Using blockchain technology, provided new data to be added to the blockchain, new operations are easily replicated through the entire network. Blockchains are designed in a manner that security management is mostly done in the background by the technology itself.

As a distributed ledger, tampering blockchain data is not easy given the same data is replicated through the entire network. A possible attack to tamper data

within the blockchain would need to target a great number of network nodes simultaneously in order to be successful. Theoreticaly, in the Bitcoin network an attacker would need to control over 50% of the network to successfully alter the blockchain, however it has already been proved the attacker only needs to control 25% of the network to execute the attack [12].

2.2 Ethereum and Smart Contracts

Bitcoin blockchain already provides a scripting language, as well as the ability to create *colored coins* and *metacoins* enabeling the use of custom protocols on top of Bitcoin with added benefits, like running decentralized applications. However those sub-protocols do not inherit the simplified payment verification. This occurs as Bitcoin network validates correct transactions within the network respecting the rules of Bitcoin transactions but also accepts invalid transactions from the perspective of some meta protocol built on top of Bitcoin as long as those are valid transactions in Bitcoin [9]. A totally secure simplified payment verification would need to perform a backward scan to the beginning of the blockchain to verify if a certain transaction, relying on a meta protocol, is valid [9].

Furthermore, Bitcoin's scripts do not support loops so the underlying programming language is not a Turing-complete scripting language. Bitcoin's scripting language also does not provide a way to limit the withdraw value and transactions only have two states, either they are spent or unspent, not providing multi-stage contracts or any other transaction states. Finally, Bitcoin transactions are blind to the blockchain as transactions can not access block data, such as the nonce or the hash of the previous block. This kind of information could be used as source of randomness [9].

Ethereum [9] was created to improve the initial blockchains and provides a blockchain infrastructure able to execute fully distributed applications and smart contracts. This improvements are partially achieved by providing a Turing-complete scripting language. The Ethereum infrastructure uses a proof-of-work approach as a consensus mechanism, although Ethereum is expected to support proof-of-stake in the future through an implementation of Casper proof-of-stake [8].

Smart contracts Smart Contracts are programs written to form agreements between users in the blockchain. Using smart contracts it is possible to ensure the clauses of a contract are accomplished and that breaching the contract is expensive or even prohibitive [18].

Blockchain establishes a consensus based on minimal trust between network nodes to execute smart contracts. When a node receives a transaction, contract functions are ran to ensure the validity of the transaction and the conditions stated in the contract are met. In case of failure the transaction is discarded by the network nodes.

Extending the capabilities of smart contracts, it is possible to run decentralized applications based on the blockchain technology. Therefore, we can create

applications such a car registry platform completely based on smart contracts. In this example there is a need to create smart contracts to define what is a car ownership entry in the blockchain as well as what are the informations needed to transfer the ownership of a car from one node to another and register it on the blockchain.

Ethereum's states are stored as a set of accounts. An Ethereum account has four fields, a nounce ensuring each transactions is unique and the transaction is not processed twice; current *ether* balance of the account; contract code, if present and storage [9]. *Ether* is a cryptocurrency used inside the Ethereum infrastructure to pay transaction fees. Also Ethereum accounts can be external accounts, controlled by private keys and having no contract code, or contract accounts, controlled by contract code. Contract code is executed in a contract account each time this account receives a message. During contract execution it is also possible for the contract code to read and write on the account's storage, as well as to send messages or even create other contracts [9].

Ethereum's messages are similar to Bitcoin's transactions but have crucial differences to support smart contracts. Messages support responses and can be created by internal accounts such as contract accounts. Ethereum's transactions are defined as signed data packages storing a message to be sent from an external account [9].

Ethereum's transactions (sent from external accounts) contain the destination of the message, a signature of the sender, the amount of *ether*, the data being sent, *startgas* and *gasprice* fields. In order to avoid infinite loops in contract code, each transaction needs to set a limit (*startgas*) on how many computational steps the code execution can take. The *gasprice* field is used to define the amount of *ether* to be paid to the miner for each computational step.

The computational unit used by Ethereum is *gas*; normally a computational step costs 1 *gas* and the network charges 5 *gas* for every byte used in a transaction. A possible attacker would need to pay a *gas* amount proportional to the resources used to perform the attack, discouraging attacks requiring great computational power [9].

2.3 Tendermint and the BFT approach

Initial blockchains such as Bitcoin's and even the improved version implemented in Ethereum still have some disadvantages related to the methods used for node consensus, as distributed consensus algorithms have some scalability constraints.

Given the theme of distributed consensus it is important to compare traditional Byzantine Fault Tolerant (BFT) algorithms [10, 11, 20] to consensus mechanisms implemented in traditional blockchains, namely proof-of-work approaches. Considering a BFT based blockchain is expected to follow the path of standard BFT algorithms, this kind of blockchain would have poor scalability compared with standard blochains using proof-of-work approach. However, BFT algorithms have a performance advantage [21], as shown in Figure 1. On the other hand, proof-of-work is easier to implement in public blockchains as nodes only need to know about one other node to perform the proof-of-work

and this approach enables decentralized identity management. BFT algorithms require each node to know every other node in the network so that consensus is reached, normally requires a trusted central authority, therefore this approaches are more likely to be implemented in permissioned or private blockchains [21].

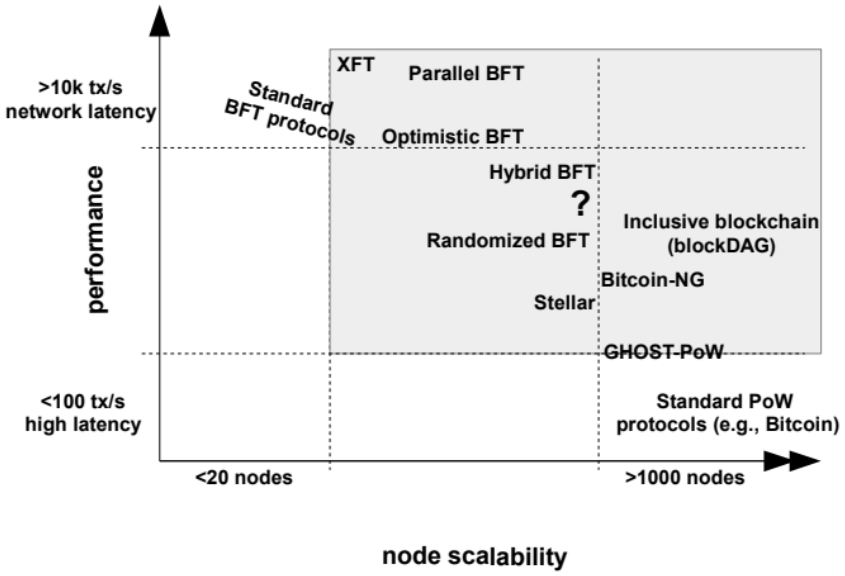


Fig. 1. Comparison of BFT protocols with blockachain protocols regarding scalability and performance. (Adapted from Vukolič [21])

Regarding consensus conflicts, such as the creation of temporary forks in the blockchain, BFT based algorithms manage them with greater efficiency compared with proof-of-work. BFT algorithms, contrary to proof-of-work, achieve this as they grant the property of consensus finality, as in Definition 1. Considering the realization of this property, block addition to the blockchain is immediately confirmed.

Definition 1 *(Consensus Finality) If a correct node p appends block b to its copy of the blockchain before appending block b′, then no correct node q appends block b′ before b to its copy of the blockchain [21].*

Resilience to attacks is also a matter of analysis as one of the key advantages of blockchain is assurance of immutable data, once the data is stored in the blockchain. Proof-of-work approach in fact supports up to 50% of faulty nodes in the network, however regarding Bitcoin's approach to fault mitigation, tolerance drops to 25% [12], contrasting with BFT algorithms that support up to $\frac{n}{3}$ faulty nodes, where n is the number of nodes on a totally asynchronous network.

Network synchrony in Bitcoin network is acheived as new blocks are only accepted if their timestamps are superior than last 11 blocks accepted by network nodes. On the other hand, this approach can be circumvented by timestamp manipulation [21]. BFT algorithms, as complying with Definition 1, support long asynchronous periods and global outages. Latency in BFT algorithms usually matches network latency, contrary to proof-of-work approach where rising block size translates in higher throughput with the downside of higher latency. As latency of the system increases the number of possible blockchain forks increases as well, resulting in more opportunities to perform double spending attacks and successfully executing them, as mentioned by Vukolič [21].

Tendermint Presented as a blockchain implementation based upon BFT algorithms to reach consensus, Tendermint provides a transparent, accountable and higher performance infrastructure, when compared with non-BFT solutions. This infrastructure offers optimization for consortia and inter-organizational logic with enough flexibility to support the development of a wide variety of applications, useful in areas ranging from e-voting to finance [7]. A big advantage of Tendermint when compared to other platforms is the availability of evolution and governance mechanisms, enabling infrastructure management and control by organizations. Compared with other blockchain infrastructures, Tendermint was developed in an academic environment and relies on signing, hashing and authentication algorithms designed in such environment, instead of NIST standards, hopping the strenghtening of its infrasctuture [7]. Tendermint uses a state-machine replication algorithm implementing Byzantine Fault Tolerant Atomic Broadcast ensuring accountability. If safety of the system is violated it is possible to identify the responsible.

Each block integrated in Tendermint's blockchain has a certain height or index such that there are no two blocks with the same index in the blockchain, preventing any blockchain forks as it happens on Bitcoin network. Validators are nodes used in Tendermint with the duty of keeping a complete copy of every state replicated in the blockchain and with the ability of proposing new blocks to the blockchain and voting on block proposals. At each height validators take turns proposing the next block to be added to the blockchain and verify that only one block is being proposed in each round.

Before transactions are grouped in blocks and later added to the blockchain, Mempool works as a temporary storage for unprocessed transactions. Therefore, as a new transaction arrives to network nodes, they add the new transaction to a list of unprocessed transactions which is called a Mempool. As transactions are validated, batched in blocks and added to the blockchain they are removed from the Mempool [7]. Blockchains normally rely on taxing transactions as a mitigation of denial of services attacks. For example, Bitcoin network nodes select transaction to create blocks based on transaction fees and when the Mempool is full one of the rules to free up space from the Mempool is by removing transactions with lower transaction fees [13]. Tendermint, on the other hand, relies on a filtering system to ignore transactions with the intent of disrupting system's

service and hopes legitimate clients are capable of resubmitting transactions if those are ignored by the system [7]. As a transaction is created and the network nodes receive it, the transaction is verified by application's logic, added to the Mempool and sent to other network nodes using the ordered multicast algorithm. Each node is responsible for ensuring that transactions are processed in the same order as it receives them.

As Tendermint implements a BFT algorithm, the system needs at least 4 validators (to tolerate 1 fault the system must have $3f + 1$ nodes given $f < \frac{n}{3}$), as well as 1 leader or proposer per round, to reach consensus. In each round a proposer within the available validators is selected through a round robin, proposing a new block to add to the blockchain. Once a validator receives a block proposal it verifies the block and signs a pre-vote which is broadcasted to the network, noting that pre-vote can be *nil* and in that case the validator is voting to skip the round and select another proposer [7]. Through this voting mechanism only one step is needed to ensure all voters know the other voters' intents.

Definition 2 *(Polka) A set of more than two-thirds of pre-votes for a single block at a given round [7].*

In the secound step of the voting mechanism the system ensures enough validators know the pre-voting results. Once a validator receives a *polka* (see Definition 2), he signs and broadcasts a pre-commit vote over the proposed block. After the pre-commit vote, if a validator receives a *polka* of pre-commit votes it commits the proposed block to his blockchain and moves to the next round increasing the height for which the next block will be proposed.

During the voting process, the proposed block is locked with a Pre-vote lock before the pre-vote phase begins. Assuring any locked block is unlocked in case of any system failure or in case consensus is not reached, Tendermint also implements Unlock-on-Polka resulting in block unlock if a *polka* for a higher round or height is received.

Separating the phase of voting or electing the next block of the blockchain from the actual addition of the block to the blockchain also presents an advantage. During voting process it is required a *polka* in the pre-voting step, as well as in the pre-commit step, so the system needs at least $3f + 1$ validators to agree in committing a given block to the blockchain, being f the number of faults tolerated by the system. However, after the voting process , the actual addition of the agreed block to the blockchain only needs the majority of replicas to do it, specifically $2f + 1$ validators [7].

Tendermint has a dedicated module called Governmint to manage the governance of the infrastructure, able to be used by a group of entities. Each group responsible is capable of internal voting for proposals whichever can lead to new actions being executed in Tendermint [7]. In a crisis recovery scenario, not only Tendermint ensures the identification of safety violating nodes but a client only needs to have access to one correct validator to understand who are the wrong validators and what is the correct blockchain.

2.4 Hyperledger Fabric

Hyperledger Fabric is a blockchain infrastructure sponsored by Linux Foundation and a consortium of companies such as IBM and Cisco. Hyperledger Fabric implements PBFT algorithm for consensus, but the consensus mechanism is modular [22] allowing consensus protocol changes with a low effort. Employing Hyperledge Fabric protocol, as a client creates a transaction he sends it to endorsing peers responsible for simulating the transaction, verifying permissions and signing transactions, if those are valid within the network. Signed transactions are then sent to their creators who gather them into a transaction proposal. Transaction proposals are broadcast to an ordering service in charge of ordering upcoming transaction proposals and creating signed chain blocks containing those transactions. As peers collect blocks, they verify endorsement policies and state changes in the blockchain. Finally, once a peer adds new blocks to its blockchain, he reports to clients the validity of their transactions. A characteristic aspect of Hyperledger Fabric protocol is the fact that invalid transactions are also stored in the chain, providing a way to identify malicious clients, although they are not executed by network peers.

Sousa et. al. [17] benefit from the flexibility of the Hyperledger Fabric architecture to propose a BFT Ordering service and explain how BFT-SMaRt can be applied as a consensus algorithm in Hyperledger Fabric. As the consensus leader receives requests, he proposes the request's group to all replicas. Every replica receiving the proposal verifies if it comes from the elected leader and if the batch of requests is valid considering the current state of the blockchain. Batches are then registered and a WRITE message is sent to the other replicas. On the next step, if a replica receives $\frac{n+f+1}{2}$ WRITE messages [17] containing the hash of the batch, an ACCEPT message is sent to every replica. Concluding this process, as a replica receives $\frac{n+f+1}{2}$ number of ACCEPT messages for the same hash, the batch of requests is taken as the decision of consensus.

An alternative system named WHEAT, based on BFT-SMaRt, has also been proposed [17]. Through WHEAT, as the WRITE phase is concluded, consensus results are sent to clients with the downside of a possible action rollback if the consensus leader changes, but provides a significant reduction on latency (almost 50%) [17]. Also through WHEAT, clients needs to wait for $\frac{n+f+1}{2}$ responses compared to $f + 1$ responses required for BFT-SMaRt.

2.5 Permissionless versus Permissioned Blockchains

In this section we will define what are public and private blockchains and also what are the differences between permissioned and permissionless blockchains. Considering the blockchain infrastructures analyzed we will also classify their configurations as public, private, permissionless and permissioned blockchains.

Public blockchains rely on public nodes, meaning that any node can contribute to the network by running a program designed to keep a local blockchain copy, contribute to network consensus, contribute to process transactions and

create blocks. As public blockchains enable any node to contribute to the network, information stored in this type of blockchain setup needs to be public. Public blockchains relying on public nodes without restricting their access and actions in the blockchain can also be considered permissionless blockchains.

As a necessity to restrict blockchain network participants to identifiable and explicitly authorized nodes with specific permissions emerged permissioned blockchains [22]. Thus it is possible to have public permissioned blockchains. In this scenario any node can still join the network however certain actions and information may need special authorizations to be executed or accessed. Even further it is possible to maintain various blockchains using this setting depending on the level of permissions of the network nodes.

On the other hand, private blockchains restrict even further the blockchain by requiring all the nodes to be authorized before joining the network. Therefore private blockchains provide mechanisms, for the organizations managing the blockchain, to restrict the nodes accessing and contributing to the blockchain. As a result, private blockchains increase the confidentiality of the data they store. However,the fact of relying on a central authority and a restrict number of nodes may provide less resilience and sturdiness.

Considering public blockchains, the requirement of making public any data stored in the blockchain has definitely the advantage of increasing data transparency but sacrifices privacy [23]. Trading off transparency with privacy needs to be considered as in case of Bitcoin it has been proved the ability to disclose ownership information. However Zero Cash uses signing algorithms through zero-knowledge proof and makes it harder to disclose such information [23].

Private blockchains partialy solve this problem and have the potential for holding data outside the blockchain, using the blockchain only to store metadata of the data stored off chain, circumventing one of the problems of public blockchain related to the amount of data stored on the blockchain. In case of Bitcoin, which is a public permissionless blockchain , each transaction can store up to 40 bytes of arbitrary data [24]. As private blockchains have identified owners such as companies, they have the ability to store information about real assets such as land registry [15]. Furthermore, private blockchains are controlled and maintained by organizations, who are able to implement Know Your Customer (KYC) policy and remove the anonymity associated with blockchain in case of legal issue resolutions [24].

Regarding scalability private blockchain networks do have advantages comparing to public blockchains. Considering the use of Ethereum infrastructure [9], as benchmarked by Xu et al. [24], running a public blockchain results in 3 to 20 transactions per second with an average mining time of 17 seconds. Using a private blockchain with the same infrastructure the mining time was around 41 seconds but resulted in a rate of 366 transactions per second.

Given the fact that permissioned blockchains have evolved from permissionless blockchains, Vukolič [22] reflects over the limitations of developing permissioned blockchains based on permissionless blockchains. Vucolić mentions that throughput of the system comes inversely proportional to latency as nodes have

to execute the smart contracts before validating transactions and creating blocks. Thus, Vucolić mentions the possibility of a denial of service attack by executing a long and exaustive smart contract [22]. However, Ethereum [9] already implements a solution to prevent exhaustive execution of smart contracts, by charging transaction size and computational time spent on the network processing transactions and smart contracts.

Separating the execution of smart contracts from the addition of new blocks to the blockchain is also presented in [22]. This separation makes a lot of sense regarding confidentiality and improves the infrastructure performance by breaking the relation between latency and throughput. Vucolić states that smart contracts only require to be executed by endorsement peers [22]. Following this line of thought, endorsement peers return to clients transaction verification results in conjunction with endorsement peers signatures and blockchain state version on which those transactions were verified. As clients receive those results, they send a transaction proposal, containing all the information returned earlier, to consensus nodes, responsible for the ordering service. As consensus nodes receive and process those transactions they perform endorsement validation (by verifying endorsement peers' signatures with no need to re-execute smart contracts) and output a sequence of blocks containing the new blockchain updates [22].

Once more it should be sufficient and safer to have a restricted group of nodes executing and verifying smart contracts and simply have the remaining nodes getting state updates, given consensus and blockchain syncrony doesn't require that all nodes execute smart contracts. In the context of car registration with multiple entities it makes sense to execute smart contracts over a restricted group of nodes controlled by the national registry institution and the consensus mechanism for block addition open to every node controlled by entities relying on this system.

Applying the method suggested by Vucolić [22] has its advantages as it eliminates the need for sequential execution. Thus consensus nodes improve their performance by avoiding the re-execution of smart contracts and also contribute to improved confidentiality, as only endorsement nodes need to be aware of the smart contracts logic. The solution proposed by Vucolić [22] is based on Hyperledger Fabric.

As presented on Table 2, Bitcoin is a public permissionless blockchain as well as Ethereum. However, Ethereum also presents the option of creating a private permissioned blockchain. Tendermint and Hyperledger Fabric are private permissioned blockchains only allowing authorized nodes to join the network and having permissions defined for each node on the network.

2.6 Applications

Although cryptocurrencies, as Bitcoin, are the most popular application of blockchain technology, there is a large set of applications based on blockchain backed by governments, banks and private companies. The use of blockchain and smart contracts may improve the government relationship with citizens and also help

government initiatives to take part of the digital world. As stated, this technology also improves the privacy of citizens and transparency of government work [23].

Applying blockchain technology to business may lower operations costs and coordination efforts, as the system works out possible conflicts [23]. This technology also has the potential for fraud reduction. Distributed ledgers can be used to better secure data itself contained in the blockchain and easily share citizen's data between government entities as well as secure critical infrastructures [23], such as the country's electrical grid.

Blockchain technology can also contribute as a method to certify the origin of products. A blockchain is being developed with this objective applied to diamonds by Everledger [19]. The project should result in a system where every diamond in the world would be registered and at any given point it would be possible to check the origin of a diamond. This presents an effort to reduce the usage of diamonds originated from war zones, as currently this kind of diamonds are mixed with legal diamonds and then sold in the industry, losing its trace.

NASDAQ was one of the first entities implementing blockchain based technology to register securities. Linq, refered by Underwood [19], reduces operation costs of securities registers and ensures real time monitoring and transaction integrity. Linq relies on the Chain infrastructure [5]. Chain Core provides a Turing-complete programming language to create smart contracts and allows any network participant to issue assets through issuance programs [1]. Chain also provides no blockchain forks as long as block signing nodes reach a quorum. Chain delegates to a single node the block creation process but any network node can validate blocks and submit transactions [1].

The Estonian government implemented a blockchain solution in 2012 applied to registries of national health system (e-Health Records [2]), judicial and citizen registry system. A concrete application of the blockchain technology by the Estonian government is e-Residence [3]. Through e-Residence application, the Republic of Estonia issues digital identities (together with a digital ID smart card) for people and organizations around the world who want to develop a location independent business. Using this application, e-Residents can start a company, access business banking, sign and securely send documents and declare taxes online [3]. The Estonian government uses blockchain technology in order for any data registered in their systems and subsequent changes to have a timestamp, identity and authenticity associated, ensuring data integrity [19].

Considering the objective of this thesis, some companies have researched and implemented property registration infrastructures through the blockchain technology. ChromaWay is implementing property purchase through smart contracts and *colored coins* in the Bitcoin network, as proposed by Mizrahi [15]. In conjunction with the Sweden government, ChromaWay finished the second stage of this project in March 2017, modeling a property transaction and including every interested party from buyers and sellers to the banks involved in the necessary loans.

Comparing each application mentioned regarding technical decisions on the blockchain configuration we can understand diferent options avaliable to im-

plement a car registration system (see Table 1). Detailed comparison of each blockchain infrastructure analysed so far will be discussed in the next section. As a side note, Chain Core and KSI blockchain [4] infrastructures will not be discussed as their documentation is limited when compared with the other infrastructures considered.

	Name	Blockchain Type	Smart Contracts	consensus finality	Fault Tolerance
Everledger	Hyperledger / Ethereum	permissioned / permissionless	Yes	Yes (Hyperledger)	$f < \frac{n}{3}$ (HyperLedger) / 50% (Ethereum)
Linq	Chain Core	permissioned	yes	Yes	-
Estonian e-services	KSI blockchain	permissioned	No	-	-
Chromaway	Bitcoin	permissionless	Yes (Colored Coins)	No	25%

Table 1. Blockchain applications comparison

2.7 Discussion

Analyzing the implementation of the similar problem related to property registration [15] through blockchain technology, based on the Bitcoin network, it is clear that the Bitcoin network is not the best infrastructure to implement such system for car registration. Firstly, the Bitcoin network is a public network with no permissioned nodes. This presents a liability as nodes are not controlled of the entity responsible for car registration maintenance. A better solution should consider a permissioned and private blockchain, enabling the responsible entity for authorizing new nodes to join the network. Though this approach it is also ensured that network nodes are identifiable. As we have seen in Section 2.2, one of the key points of using Ethereum over Bitcoin infrastructure was the lack of simple payment verification over sub-protocols built on top of Bitcoin network. Moreover the capacity of storing contract states and the use of a Turing complete language for contract definition makes it easier to implement a more sturdy system based on Ethereum [9].

In relation to consensus protocols, it is also crucial to identify which one seams more adequate for a car registering system, assuming a permissioned and private blockchain. The car registering system is most likely to be implemented in a hardware infrastructure own by the government. Therefore, the use of proof-of-stake is not adequate to the system as one of the key factors of this kind of consensus protocol is relying on nodes owning a great stake in the system. In this system however, stored data refers to ownership of cars and network nodes do not possess goods, as cars are owned by citizens who do not participate in this blockchain network.

A second consensus protocol mentioned is the proof-of-work. This method does make sense to be used by the car registering blockchain however has its downsides. As the problems presented by proof-of-work require a great effort to solve [16] they are directly translated into huge processing power spent. Proof-of-work demands high electricity costs, as a matter of fact Bitcoin network, which

uses this consensus mechanism, consumed more electricity in 2017 than Ireland in this same year [6]. Thus the use of proof-of-work is not an option given the costs of maintaining such system.

At last we consider the use of a BFT based consensus mechanism such as the one presented by Tendermint or the consensus mechanism presented by Hyperledger. BFT based consensus accomplishes consensus finality as presented by Vucolic [21] and mentioned in Section 2.3. Compared to proof-of-work also demands less processing power and presents a fault tolerance system based on the number of existing nodes on the network. This consensus algorithms also present an higher throughput network compared with proof-of-work (see Table 2) and achieve block latencies or average mining times approaching the usual network latencies, in the order of secound or even milliseconds [7, 17].

Considering the existing infrastructures for implementing this system, it should be able to deliver an high performance, so the best way is also favoring the use of a private ledger, improving the transaction speed to around hundreds of transactions per secound using Ethereum or even thousands of transactions per second using Tendermint infrastructure [24, 7].

Tendermint is an infrastructure based on Bysantine Fault Tolerant Systems so it supports faults acording with $f = \frac{n-1}{3}$ where n is the number of nodes and f the number of faults tolerated, roughly 33% of faulty nodes when n aproaches infinity. Compared with the 25% of Bitcoin network [12] and theoretically 50% of nodes failing on Ethereum.

Performance wise, Bitcoin is limited by 7 transactions per second as the consensus mechanism requires a great node effort to process transactions and as consequence requires around 9 minutes to create a block and add it to the blockchain. On the other hand, Ethereum not only provides the choice of using either a public or a private blockchain but also provide major performance improvements as this infrastructure can process around 20 transactions per second in a public blockchain with a latency of 17 seconds or around 366 transactions per second in a private blockchain and latency of 41 seconds as presented by Xu et al. [24] and shown in Table 2. Tendermint provides around 10^4 transactions per second when 32 nodes are connected to the network and more than 10^4 transactions per block. Furthermore provides tolerance to 10 faults according to $f < \frac{n}{3}$ with a latency of around 1.1 seconds [7].

Hyperledger Fabric (HLF) is also a great infrastructure, based on BFT algorithms, to analyse as it is backed by big industry players such as IBM. Considering the use of HLF with BFT-SMaRt [17] and even the work of Vucolić [22] by separating the smart contract execution from block addition seams to be adequate to our solution.

Mentioned as a weak factor for most blockchain infrastructures, consensus protocols are usually implemented as hard coded [22]. Except HLF, both infrastructures mentioned, as Bitcoin network [16], Ethereum [9] and even Tendermint [7], suffer from this weakness. Changing consensus protocols of those infrastructures depends upon major code transformations [22]. Although we tend to suggest a consensus mechanism based upon BFT protocols, the ability to easily

change the underlying consensus mechanism of the infrastructure is a strong advantage if we want a system allowing for scalability, durability and maintainability.

	Bitcoin	Ethereum	Tendermint	Hyperledger Fabric
Blockchain Type	Permissionless	Permissionless or Permissioned	Permissioned	Permissioned
Consensus Protocol	proof-of-work	proof-of-work	BFT Algorithm	modular (PBFT, BFT-SMaRt, WHEAT)
Smart Contracts	Yes (Colored Coins)	Yes	Yes	Yes
Simple payment verification	No	Yes	Yes	Yes
Consensus Finality	No	No	Yes	Yes
Fault Tolerance	25%	50%	$f < \frac{n}{3}$	
Throughput (transactions/second)	7	20(public)/ 366(private)	around 10^4	around 22000
Average Mining Time	9m	17s(public)/ 41s(private)	1.1s	0.6s (BFT-SMaRt) /0.32s (WHEAT)

Table 2. Blockchain infrastructure comparison

3 Proposed Solution

A possible solution for a car registration system based on blockchain technology should follow a set of well defined requirements in order for the system to be useful and for the government to take advantages of it compared with the currently implemented system. In this section we will go over some of the requirements an information system handling car registration data should comply to. Based on the requirements defined, we propose which blockchain configuration seams better suited to this problem. Finally we present a possible solution to implement the car registration system based on blockchain.

3.1 Requirements

Currently the car registration system is totally controlled and maintained by government entities. As blockchain technology relies on distributed nodes, the control detained by government entities must be adapted. Therefore, a car registration system based on blockchain technology can distribute most of the maintenance effort and even some of the system's control by network nodes. However the government should detain most of the control over the car registration. The government should also be able to have the role of a supervising authority.

Although the blockchain technology promotes a completely distributed application of the car registration system, the government should own some, or even most, of the network nodes. This requirement is proposed so that the system can still work even if all of the external entities contributing with network nodes stop providing consensus nodes. The government should also have enough control to override ownership when executing a judicial order. Considering non-functional requirements, the system should be designed for high availability and fault tolerance as well as to guarantee safety of the data. Furthermore the system should also be resilient to attacks. All of this non-functional requirements may be achieved using the blockchain technology.

As part of a car registration system's requirements, at least the following use case scenarios (for UML use case diagrams see Appendix A) should be considered:

A. Main use cases:
   1. A car seller wishes to transfer its car ownership to a car buyer. (see Appendix A.1)
   2. A leasing company provides a contract to a client with the clauses of the client using the car for a defined period of time on which the leasing company is responsible for paying maintenance and insurance expenses in exchange for a defined monthly payment by the client. As the contract reaches its ending, the client can buy the car from the leasing company for a previously defined amount. (see Appendix A.2)
   3. A citizen buys a car using a loan and the car is registered with a retention of title to the bank providing the loan. (see Appendix A.3)

B. Secondary use cases:
   1. A car manufacturer submits a request to create a car registry entry for a newly produced car. (see Appendix A.4)
   2. A car owner submits a request to give the car ownership as a guarantee to a creditor in case the car owner does not fulfill the contract. (see Appendix A.5)
   3. An authority, such as the police, wants to verify the ownership of a car. (see Appendix A.6)
   4. Given a judicial order and in order to liquidate a car owner debt, the car ownership is transferred to the entity responsible for collecting the debt payments. This use case should be executed by a judge and a national registry's employee. (see Appendix A.7)
   5. In event of a crash or other misfortune a car is considered totaled and its state on the car registration system should be updated to officially be considered out of circulation. This order might possibly be executed by a national registry's employee. (see Appendix A.8)

3.2 Blockchain

Considering the requirements of a car registry information system and the various blockchain configurations discussed in Sections 2.5 and 2.7 , the system should be a permissioned ledger controlled by a responsible government unit, such as the national registry institution. The blockchain should also be a private ledger and only authorized nodes should be allowed to join the blockchain network.

As we intend to provide most of the blockchain control to an entity, as the national registry institution, the car registration system should provide various levels of permissions to the different nodes in the network. In this scenario we can have nodes with the only permission of accessing the system records, therefore this nodes do not participate in the consensus mechanisms and can not submit car registration transactions. Other nodes such as the nodes controlled by the government may have higher permissions to the system and may validate and execute smart contracts and transactions.

Ensuring the safety and confidentiality of data, the blockchain network should be a private blockchain. As a private blockchain, the government can also detain the control over who accesses the blockchain. Only authorized nodes detained by the government or by external entities related to the car registration system, such as leasing companies, can join the network. Furthermore this configuration ensures that the car registration data is only available to trusted entities and does not expose citizens information to the public domain.

3.3 The Solution

As part of the solution, an hybrid approach, as suggested by Mizrahi [15], where a car registry transaction (see Definition 3) is signed by the correct owner and by the registry entity is the most plausible approach.

Definition 3 *Car registry transaction is any operation related to the creation or modification of a car registry record. This includes the operation to change the car ownership or the operation to declare that a car was destroyed.*

As the system should be able to execute the use cases mentioned in Section 3.1 , we suggest an approach based on Hyperledger Fabric infrasctructure using BFT-SMaRt [17]. We choose Hyperledger to support our application as it supports a private ledger configuration as well as the capacity of attributing permissions to the various nodes participating in the network. As mentioned in Section 2, Ethereum and Tendermint also support this configurations. However, Ethereum currently relies on proof-of-work to reach consensus and as discussed on Section 2.7 a BFT consensus algorithm provides better energy efficiency and performance. Therefore we can consider Tendermint and Hyperledger Fabric as infrastructures to implement this application. As Hyperledger Fabric has a modular consensus mechanism it presents a strong feature, considering the durability and scalability of the system, when compared with Tendermint. On the other hand, Hyperledger Fabric is supported by technology companies, such as IBM, and Linux Foundation providing an infrastructure that will probably have a better support than Tendermint.

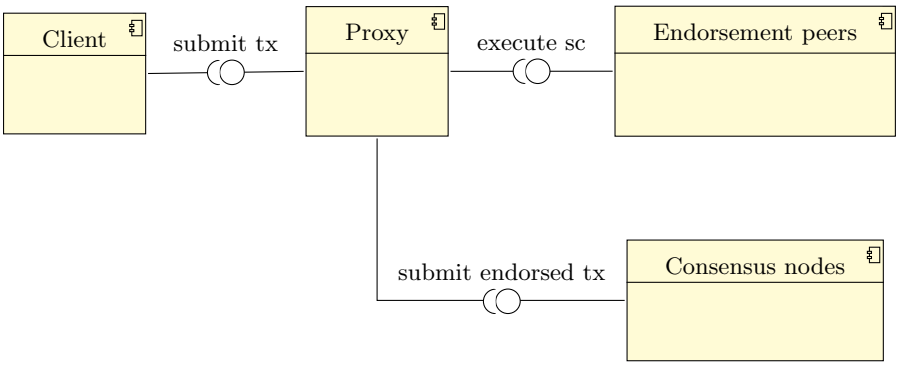


Fig. 2. Overview of the solution

In addition we also propose the separation of smart contract execution from the consensus protocol as suggested by Vukolič [22]. Considering this separation our solution should have endorsement nodes responsible for executing and validating smart contracts, as presented in Figure 2. Consensus nodes do not need to execute the smart contracts again and validate transactions based on the signatures of endorsement nodes. In [22], the client is responsible for submitting a transaction and the underlying smart contract to endorsement peers, collect their signatures an later submit the transaction and endorsement signatures to consensus nodes. To simplify the client coordination efforts we propose a proxy

to handle all of this steps, requiring the client to simply submit the transaction to this proxy.

Using this approach, endorsement nodes can be fully detained by a government entity. Therefore the government will detain most of the control over the system, as endorsement nodes are crucial for the system to work. As the smart contracts will only be executed by endorsement nodes it will turn the creation of a block and reaching a consensus into a simpler process as consensus nodes will only need to verify endorsement signatures to validate transactions.

Consensus nodes will have the only duty of ordering transactions, validating them by collecting a set of endorsement nodes' signatures and reaching consensus. As the car registration system is accessed by multiple entities, ranging from government's financial authorities to leasing companies, this entities can also contribute with authorized consensus nodes to the network. This way the system benefits from distributing its nodes by various locations, as intended by the blockchain technology properties, enhancing the system's overall fault tolerance and performance. The capacity of distributing at least part of the blockchain network through the organizations who access car registration data also presents an added benefit to those organizations, as each one has a copy of the blockchain, providing a great performance improvement on data access compared with a centralized database.

Vukolič [22] presents a scheme where the client initially sends its transaction to endorsement peers and receives the transaction's validation in the form of endorsement peers' signatures. Later the client is responsible to gather those signatures and the version of the blockchain on which the transaction was verified to create a transaction proposal. After this step the client sends the transaction proposal to the ordering service. In order to simplify the client's communication with the blockchain network we propose the use of a proxy responsible for handling the client communications with network nodes. Using a proxy the client sends the transaction through the proxy and later receives the result of transaction request, as shown in Figure 3. This solution might resemble part of Tendermint's approach [7] however takes advantage of the industry backed infrastructure HLF and its flexibility regarding consensus protocol selection and invalid transaction logging. On the other hand, enables the already mention separation of smart contract execution from consensus nodes. Through this approach, the execution of ownership overriding by a government entity is assured as it is possible to tweak endorsement peers to execute this kind of actions. A possible execution of this requirement may delegate to special clients (e.g.: identified by a key pair) the responsibility to create transactions imposed by court or other government entities.

The proxy is then responsible for sending the transaction to endorsement peers and receives a signed transaction or an invalid transaction result. In case of the latter situation the proxy notifies the original client that his transaction is invalid. Otherwise, the proxy collects signatures of the transaction and crafts the transaction proposal as an envelope including the original transaction, trans-

action's signatures and blockchain version on which the transaction was verified. A transaction proposal is then broadcast to consensus nodes (see Figure 3).

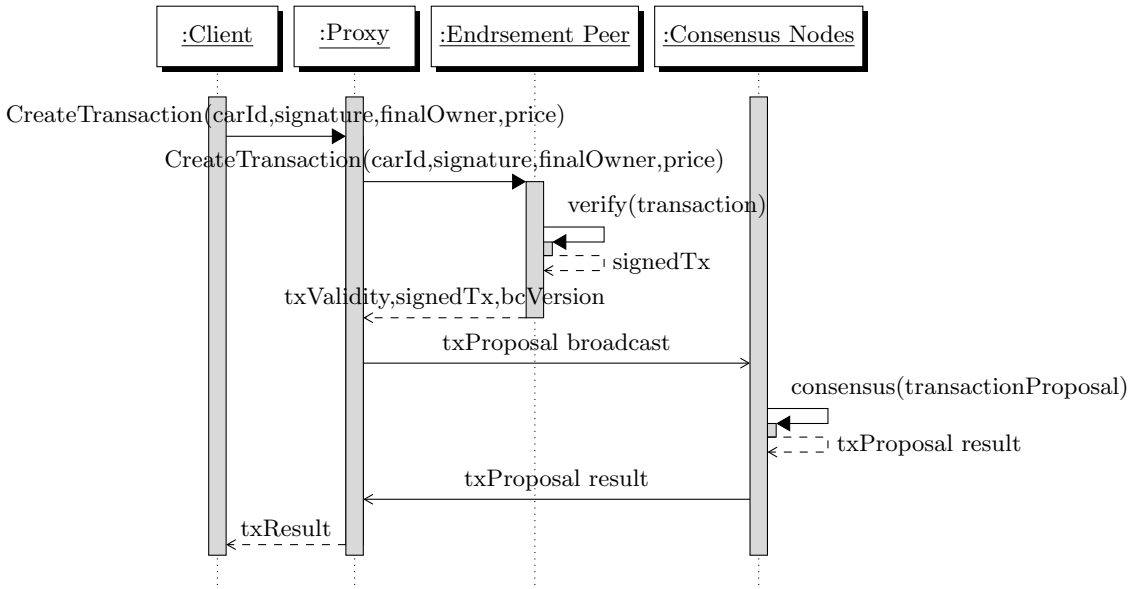


Fig. 3. Suggested transaction process based on Hyperledger Fabric infrastructure.

Once the consensus nodes receive a transaction proposal broadcast, the consensus mechanism is triggered (presented in Appendix 4). The consensus algorithm used is BFT-SMaRt [17], where an elected leader gathers a set of transaction proposals verifying the endorsement policies of the transactions and their validity acording to the current version of the blockchain and creating a transaction batch or a block. This block is then proposed by broadcast to every other consensus node. As consensus nodes receive the transaction batch proposal they verify endorsement policies and the blockchain version to validate the block's transactions. The origin of the proposal is also verified and if it does not come from the elected leader the proposal is invalid. As blocks are verified, they are registered by every node of the network and this implies that every node sends a write message with the hash of the block to every other node in the network. For a matter of simplicity, Appendix 4 only presents the broadcast of the leader and the consensus node 1. If replicas receive $\frac{n+f+1}{2}$ Write messages [17], they broadcast an Accept message and add the new block to the blockchain, updating the blockchain version. Finally each replica receiving $\frac{n+f+1}{2}$ Accept messages delivers to the proxy the transaction result. The proxy is then required to wait for $f+1$ transaction results from replicas [17] before returning the final transaction result to the client.

The alternative WHEAT algorithm, suggested in [17], was also considered and can improve the overall latency of the system. However, WHEAT requires the proxy to wait for $\frac{n+f+1}{2}$ responses to confirm the transaction result. This algorithm also presents the drawback, as mentioned in Section 2.4, of a possible rollback if a leader is changed during a consensus round.

As part of the proposed solution we will need to define which data is needed to execute a car registration transaction and which data should be stored in a car registration entry in the blockchain. It is clear some data, as the car license plate or the engine size of the car, will need to be store in the blockchain as well as information about the owner. However, the analysis of the current car registration system will provide most of the information needed to track the car ownership record. The underlying mechanisms of a car registration entry and the use cases mentioned in Section 3.1 will be defined by creating smart contract for the Hyperledger Fabric infrastructure.

4 Evaluation Methodology

As part of evaluation methodology the proposed solution should be compared to the current car registration system. A detailed analysis of the currently centralized system will be executed to understand its fault tolerance and performance. The data model of the current car registration system will be studied, not only to extract information about how a car registration entry should be defined, but also to understand if additional car information should be stored in the system. The architecture of the current system should also be analysed to evaluate the throughput, latency and fault tolerance of the system. Once this analysis is concluded, a set of tests over the solution here presented should be conducted. We will start by evaluating the performance of the proposed solution with an hardware infrastructure resembling the infrastructure currently used by the centralized system, to understand immediate benefits or downfalls of implementing this new system under the same infrastructure. In this evaluation, the same metrics extracted from the current system should be collected, such as throughput, latency and fault tolerance.

Our solution should be compared with the actual system for car registration considering the availability, fault tolerance and safety of both systems. Also the effort to coordinate attacks to both systems should be analysed, as well as the rate of success of such attacks.

A detailed test considering a variety of nodes in the network should also be performed in a second stage of testing. At this stage it could be interesting to understand latency and throughput performance considering a varying number of nodes responsible for endorsement peers and also a varying number of nodes responsible for the consensus mechanism. Ideally starting from a fault tolerance of 3 fault to atleast 10 faults and extending the fault tolerance even further if possible, mostly on the consensus nodes. A performance test considering a number of faulty nodes should also be performed to test the fault tolerance

of the system. In a matter of safety assurance it should also be considered a tampering attempt towards the proposed solution.

It could be interesting to simulate the performance of the proposed system under a truly distributed architecture, considering at least the consensus nodes as distributed by the different entities currently accessing car registration data. As the number of external entities could vary, at least a set of 3 clustered nodes should be considered. Always having in mind that the national registry entities should own the totality of endorsement nodes and should also own a significant part of consensus nodes. This performance tests can be conducted by relying on a set of nodes installed and configured in a cloud environment, such as Amazon Web Services.

5 Work Schedule

Work/Month	Jan	Feb	Mar	Apr	May	Jun	Jul
ASIS System Analysis	■	■					
Data Model Creation		■	■				
Initial prototype Development		■	■				
Smart contract programming		■	■	■			
Testing					■	■	
Result Analysis					■	■	
Write dissertation			■	■	■	■	■

6 Conclusions

Thought this document we iterated over different blockchain infrastructures and their consensus mechanisms comparing each one. Finally we proposed a solution for car registration based on the existing blockchain infrastructures available.

Starting with the Bitcoin network we presented the different properties of blockchain technology and how this technology works. Current limitations to implement business applications, such as the one we present, using the Bitcoin network were also refered. We then looked at proof-of-work as used by Bitcoin and Ethereum for consensus mechanism and its principal alternative, proof-of-stake. As an evolution of the Bitcoin network we looked at Ethereum infrastructure and its advantages. Ethereum enables of simple payment verification and truly introduces the ability to use smart contracts in transactions, with the added benefit of using a Turing complete language for doing so.

Considering consensus mechanisms, we mentioned the use of a BFT algorithm to decide block additions to the blockchain and introduced Tendermint [7] as an infrastructure using a BFT-based algorithm to solve this problem. We introduced Hyperledger Fabric as another infrastructure based on PBFT algorithms and its modular properties, as well as some alternatives [22, 17] to improve this infrastructure's performance and reliability. Then we defined the

difference between permissionless and permissioned blockchains. Finishing related work section we compared each infrastructure mentioned, analyzing their throughput, latency, blockchain type and consensus mechanisms.

We then proposed a solution based on Hyperledger Fabric with smart contract execution separation, as sugested by Vukolič [22], and a consensus algorithm based on BFT algorithms, BFT-SMaRt [17]. In our solution we also take in consideration the different use cases a car registration system should have, as a simple car ownership transaction or a car registry suppression as a consequence of an accident. Finally we defined a set of guidelines on how the evaluation methodology of the system will be conducted.

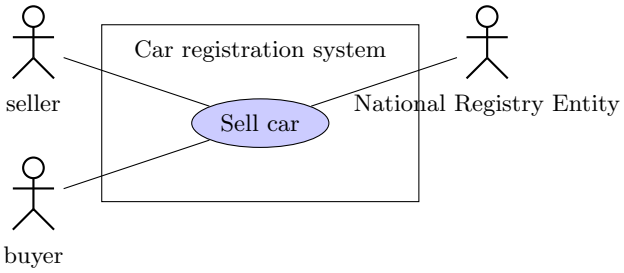
References

1. Chain protocol whitepaper, https://chain.com/docs/1.1/protocol/papers/whitepaper
2. e-health records, https://e-estonia.com/solutions/healthcare/e-health-record
3. e-residency, https://e-resident.gov.ee/
4. Guardtime's ksi technology stack, https://guardtime.com/technology
5. Nasdaq linq enables first-ever private securities issuance documented with blockchain technology (Dec 2015), http://ir.nasdaq.com/releasedetail.cfm?ReleaseID=948326
6. Bitcoin mining now consuming more electricity than 159 countries including ireland most countries in africa (Nov 2017), https://powercompare.co.uk/bitcoin/
7. Buchman, E.: Tendermint: Byzantine Fault Tolerance in the Age of Blockchains (Master's thesis) (2016), http://atrium.lib.uoguelph.ca/xmlui/handle/10214/9769
8. Buterin, V., Griffith, V.: Casper the Friendly Finality Gadget. ArXiv e-prints (Oct 2017)
9. Buterin, V.: Ethereum: A Next-Generation Cryptocurrency and Decentralized Application Platform. Bitcoin Magazine (2014), http://www.the-blockchain.com/docs/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf
10. Castro, M., Liskov, B.: Practical Byzantine fault tolerance. In: Proceedings of the 3rd USENIX Symposium on Operating Systems Design and Implementation. pp. 173–186 (Feb 1999)
11. Correia, M., Veronese, G.S., Neves, N.F., Veríssimo, P.: Byzantine consensus in asynchronous message-passing systems: a survey. International Journal of Critical Computer-Based Systems 2(2), 141–161 (2011)
12. Eyal, I., Sirer, E.G.: Majority is not enough: Bitcoin mining is vulnerable. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 8437, 436–454 (2014)
13. Harding, D.A., Todd, P.: Opt-in full replace-by-fee signaling (Dec 2015), https://github.com/bitcoin/bips/blob/master/bip-0125.mediawiki
14. King, S., Nadal, S.: PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake. Ppcoin.Org (2012), http://ppcoin.org/static/ppcoin-paper.pdf
15. Mizrahi, A.: A blockchain-based Property Ownership Recording System. ChromaWay (2015)
16. Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System. Www.Bitcoin.Org p. 9 (2008), https://bitcoin.org/bitcoin.pdf

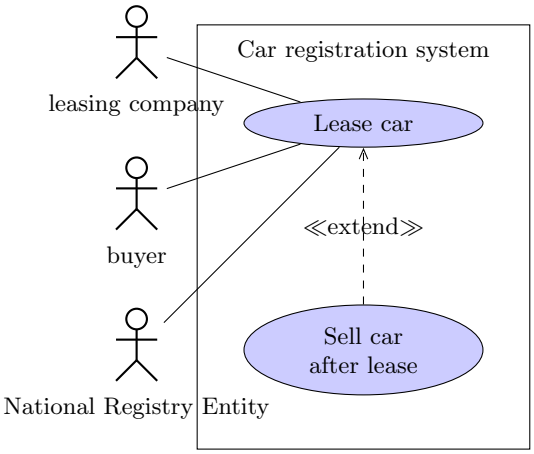
17. Sousa, J., Bessani, A., Vukolić, M.: A Byzantine Fault-Tolerant Ordering Service for the Hyperledger Fabric Blockchain Platform (Section 4) (2017), http://arxiv.org/abs/1709.06921
18. Szabo, N.: The idea of smart contracts. Nick Szabos Papers and Concise Tutorials (1997)
19. Underwood, S.: Blockchain beyond bitcoin. Communications of the ACM 59(11), 15–17 (2016), http://dl.acm.org/citation.cfm?doid=3013530.2994581
20. Veronese, G.S., Correia, M., Bessani, A.N., Lung, L.C., Verissimo, P.: Efficient Byzantine fault tolerance. IEEE Transactions on Computers 62(1), 16–30 (2013)
21. Vukolić, M.: The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 9591, 112–125 (2016)
22. Vukolić, M.: Rethinking Permissioned Blockchains. Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts - BCC '17 pp. 3–7 (2017), http://dl.acm.org/citation.cfm?doid=3055518.3055526
23. Walport, M.: Distributed ledger technology: Beyond block chain. Government Office for Science pp. 1–88 (2015)
24. Xu, X., Pautasso, C., Zhu, L., Gramoli, V., Ponomarev, A., Tran, A.B., Chen, S.: The blockchain as a software connector. Proceedings - 2016 13th Working IEEE/IFIP Conference on Software Architecture, WICSA 2016 pp. 182–191 (2016)

A Use cases

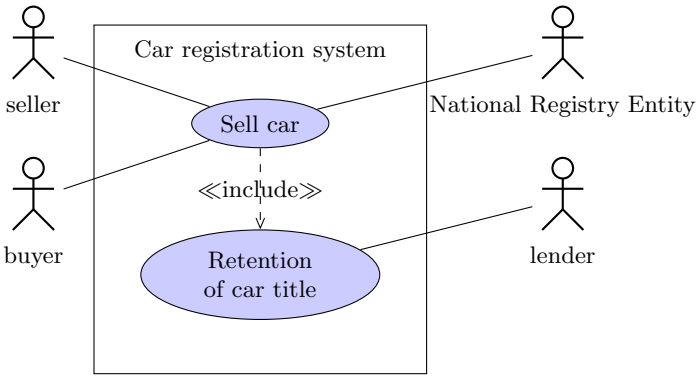
A.1 Car sale



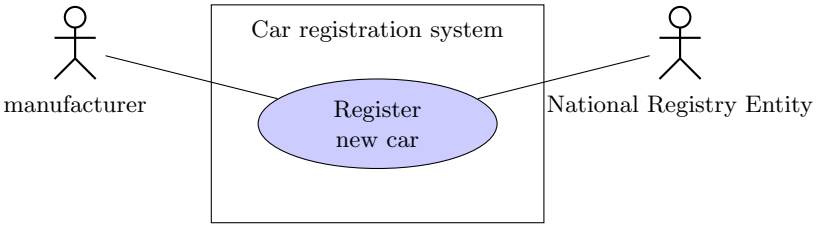
A.2 Car leasing



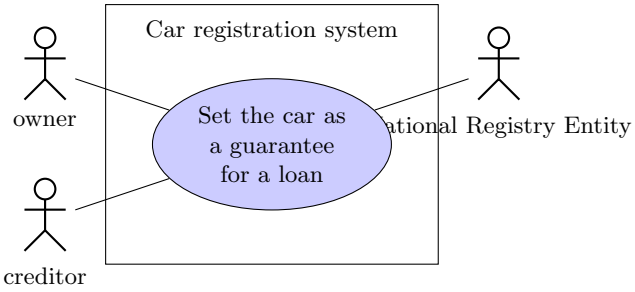
A.3 Retention of title



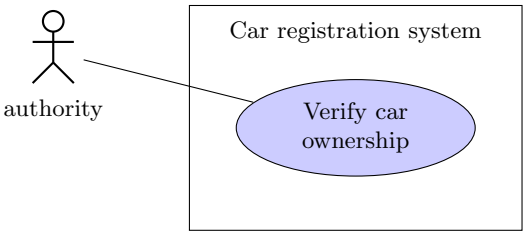
A.4 New car registry



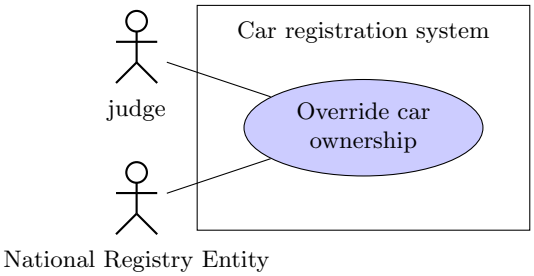
A.5 Mortgage



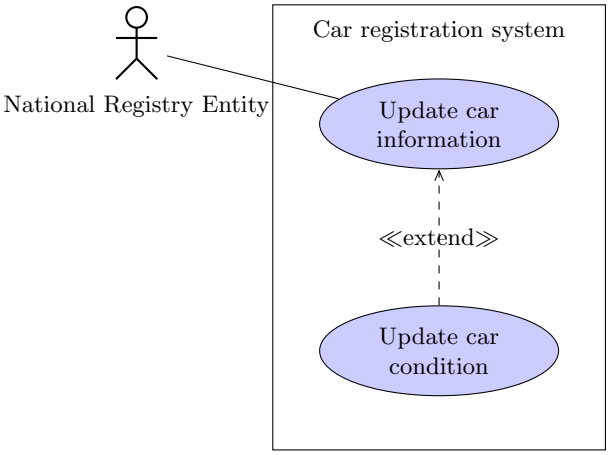
A.6 Car ownership verification



A.7 Ownership override by judicial order



A.8 Update car information



B BFT-SMaRt implementation for Hyperledger Fabric

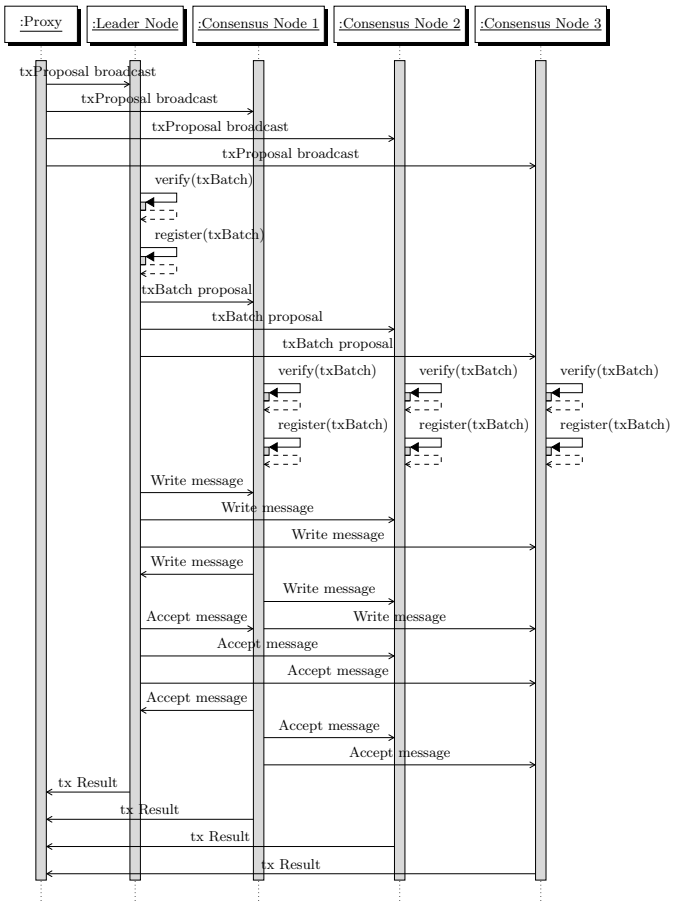


Fig. 4. Block proposal and addition mechanism based on BFT-SMaRt for HyperLedger Fabric [17].