Expertise
and insight
for the future

Amrit Gautam

Immutable Storage of EV Charge Records Using Blockchain Technology

Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Bachelor's Thesis

11 March 2019

Author	Amrit Gautam
Title	Immutable Storage of EV Charge Records Using Blockchain Technology
Number of Pages	37 pages
Date	11 March 2019
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Professional Major	Software Engineering
Instructors	Kimmo Sauren, Senior Lecturer

After the first successful implementation of the decentralized and distributed digital currency, Bitcoin, its core disruptive technology called Blockchain has been the subject of interest for many people and business organizations in order to harness its power. This thesis is an attempt to understand blockchain technology and its use cases in the field of electric mobility. The primary goal of this thesis was to develop a prototype based on blockchain technology, to store immutable charge records of electric vehicles for Liikennevirta Oy, aimed to strengthen trust and data integrity of customers.

The project started with preliminary research on different available platforms in order to find a suitable platform to meet the requirement of the business. Design of the network architecture and the smart contract were done consecutively after the selection of the platform. Manual deployment of the prototype was done to Amazon Cloud Service using a single instance of Elastic Cloud Compute for testing purpose.

As a result, a permissioned blockchain network, a smart contract (chaincode) and a REpresentational State Transfer (REST) Application Programming Interface (API) server were developed using Hyperledger Fabric platform and Hyperledger Composer tool. The charge data record is stored in the nodes of the network when the electric vehicle charging process is initiated or stopped. Another potential use case of blockchain technology to build a decentralized roaming platform for electric mobility providers has been realized and discussed in this thesis.

In conclusion, Blockchain technology itself leaves no doubt for maintaining the integrity of stored data. However, it has no control over the truth of the data being asked to be stored as it is influenced by outside factors such as human and sensors. The developed prototype also relies on the authenticity of data supplied by charging stations to provide trust and transparency.

Keywords	distributed system, decentralization, ledger, blockchain technology, smart contract, hyperledger fabric, hyperledger composer, electric vehicle

Contents

List of Abbreviations

API	Application Programming Interface
BFT	Byzantine Fault Tolerant
BFT-SMaRT	Byzantine Fault Tolerant State Machine Replication
BNA	Business Network Archive
BND	Business Network Definition
CA	Certificate Authority
CDR	Charge Data Record
CFT	Crash Fault Tolerant
CRUD	Create, Read, Update and Delete
DApp	Distributed Application
DDoS	Distributed Denial-of-Service
DPoS	Delegated-Proof-of-Stake
EMP	Electric Mobility Provider
EV	Electric Vehicle
HLF	Hyperledger Fabric
HTTP	Hypertext Transform Protocol Secure
MD5	Message Digest Algorithm 5
MSP	Membership Service Provider

OCPP	Open Charge Point Protocol
OSN	Ordering-Service Node
PBFT	Practical Byzantine Fault Tolerance
PKI	Public Key Infrastructure
PoS	Proof-of-Stake
PoW	Proof-of-Work
REST	REpresentational State Transfer
RIPEMD	RACE Integrity Primitives Evaluation Message Digest
SHA	Secure Hashing Algorithm
TLS	Transport Layer Security
UI	User Interface
VPN	Virtual Private Network

1 Introduction

"Trust" is the very fundamental property on which we rely on, by default, when using services of any kind on a daily basis provided by different service providers. Such services include an exchange of anything of value which varies from buying grocery to trading of gold. In any case, consumers need to have blind faith in service providers that they are not being deceived. Service providers are the only and responsible members for maintaining a single source of truth as consumers are left out with just the receipts of the transaction. Despite the presence of separate regulatory bodies responsible to monitor and regulate these service providers on a regular interval, treachery, fraud, and deception persist.

As business organizations are going paperless and moving towards digitalization, thanks to the Internet, information security is becoming more critical. Data breaches add complexities for organizations to maintain the integrity of stored information and build trust in this expanding digitized world. The de facto standard to digitally store information is to use the centralized database which performs well according to the need of the businesses despite its demerits such as a single point of failure and truth, possibility to alter information without the consensus or approval of the consumers, and lack of inbuilt history log [1]. However, Blockchain, as a new and emerging technology promises to increase data integrity, trust and transparency compared to a centralized database by being distributed and decentralized at its core.

Liikennevirta (Virta) Oy is a startup company with the core goal to boost the rate of adoption of Electric Vehicles (EVs) by operating the reliable charging network and give the best charging experience for its users globally. In addition, it builds new technologies to manage and maintain different renewable energy resources with a vision to end climate change. [2.] In order to build trust and transparency with end consumers and different business partners, it is critical for the company working in the field of Energy sector to record consumption of electricity or Charge Data Record (CDR) by EVs, in any of the charging stations in its network, securely in an immutable and distributed way. This case is one potential use case of modern blockchain technology, which is studied in detailed throughout this thesis.

The goal of this thesis was to build a proof of concept system which allows immutable storage of charge records of EVs using blockchain technology. Data is transferred

through the use of REpresentational State Transfer (REST) Application Programming Interface (API) between Virta's system and blockchain system.

The remainder of this thesis is divided into 6 chapters. Chapter 2 provides a brief overview of different components that compose the blockchain such as the distributed system, cryptography, consensus mechanisms, smart contracts and platform selection. Chapter 3 describes the architecture of selected blockchain platform, Hyperledger Fabric (HLF) in detail. Chapter 4 presents the implementation details of the developed prototype blockchain network and smart contract. Results and findings of the project are discussed in Chapter 5 and Chapter 6 respectively. Finally, Chapter 7 concludes this thesis and proposes a new use case of blockchain technology in the field of electric mobility for future development.

2 Background

This chapter aims to provide the background information required to understand the rest of this thesis. It starts with the basic concepts such as distributed system, ledger, cryptography that blockchain technology is made of and concludes with the selection of right platform for the development of the prototype.

2.1 Basic Concepts

2.1.1 Distributed and Decentralized System

Distributed system is the collection of independent nodes that collaborate to achieve the same goal. Independent nodes can be hardware devices ranging from sensors to supercomputers or software processes. Any distributed system is a single coherent system from the perspective of end users. Synchronization and coordination among distributed nodes are the fundamental challenges within a distributed system due to the fact that each independent node has its own notion of time. Management of membership of nodes in a closed group of a distributed system is much complex than an open group where any nodes can join and leave freely. [3, 1-7.]

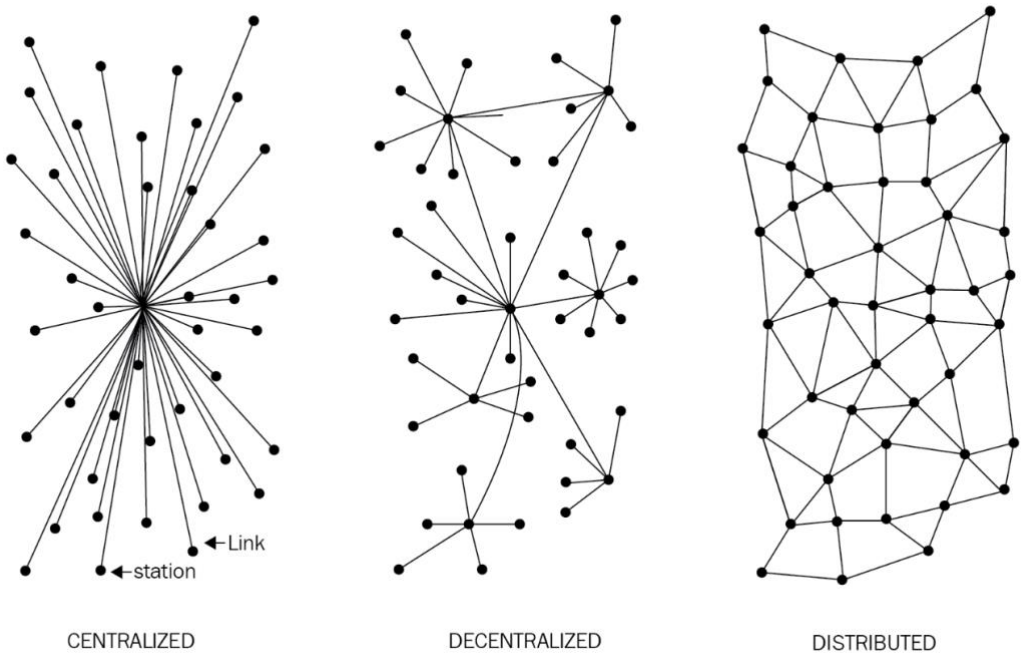

Figure 1. Different types of network architecture. [4]

A decentralized system lacks a single, central governing entity. Instead, governance is distributed among independent entities. Consensus has to be reached among all entities in the network for a change to be effective. Figure 1 shows three different types of network architecture: centralized, decentralized, and distributed, commonly used in the field of computer science.

2.1.2 Peer-to-peer Network

A node in a Peer-to-peer network is independent and possesses the capability to act as a server as well as a client at the same time. A key requirement in this network architecture is that the network should be fully functioning even if any arbitrary node is removed from the network. [5.]

2.1.3 Ledger

A ledger refers to any tangible (e.g. books) or intangible (e.g. electronic files, databases) entity on which information is recorded in an organized way. A distributed ledger is a digital ledger which is spread across the network among all the peers of the network and each peer is responsible to maintain the integrity and consistency of the ledger [6].

2.2 Cryptography

Cryptography is one of the core foundations on which blockchain technology builds upon and plays a vital role to create a trustless environment in a decentralized network. It is the study of method and techniques to establish secure communication so that the intended recipient of the message can only read and process the information despite the presence of adversaries. Use of modern cryptography ensures that the information is secured with the main objectives of confidentiality, data integrity, authenticity, and non-repudiation. Confidentiality refers to keeping the content of information from all but those authorized to have it. Data Integrity confirms the accuracy and consistency of data. Authenticity relates to corroboration of the identity of any parties involved. Non-repudiation refers to the ability to ensure that any party which has originally sent the message cannot deny their authenticity. [7, 1-5.]

Raw information which has to be securely transferred is encrypted using Ciphers, a mathematical algorithm, with specials keys. This produces ciphertext, a piece of information which is complete nonsense and junk until it is decrypted with correct keys by its recipient on the other end.

A cryptographic protocol or system is composed of different basic blocks of low-level algorithms which are called Cryptographic primitives. Figure 2 shows an overview of different blocks that can be classified as Cryptographic primitives. [6.]

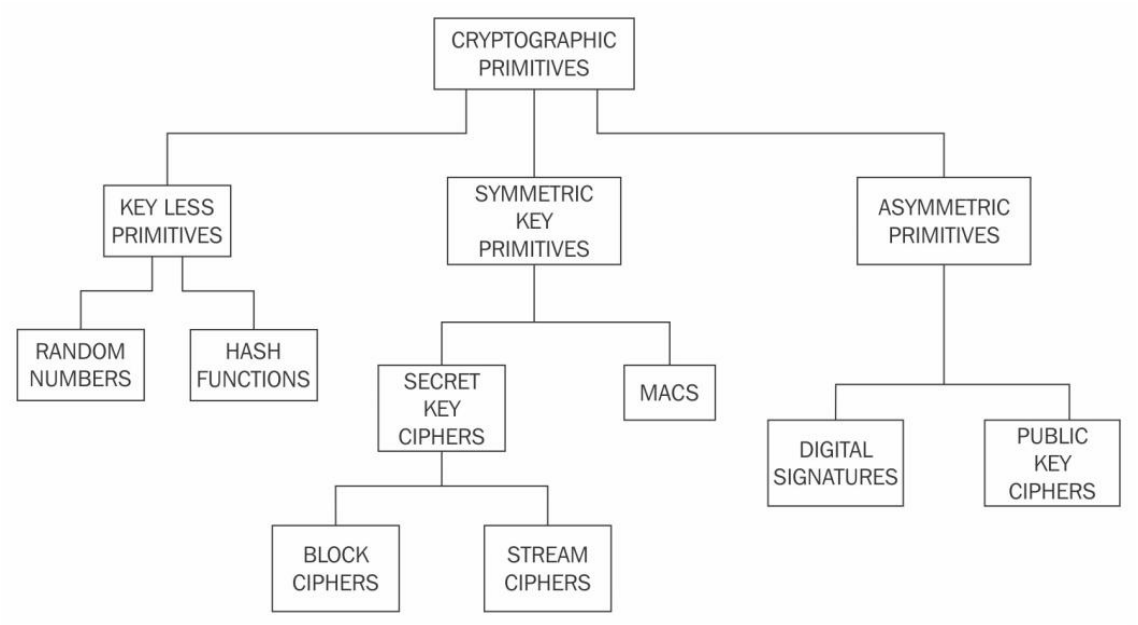

Figure 2. Taxonomy of Cryptographic primitives. [6]

Blockchain technology uses Cryptographic primitives for maintaining the identity of participants, the integrity of world ledger, the authenticity of transactions, and the privacy of transactions. Some of these cryptographic primitives are explained below. [6.]

2.2.1 Symmetric Cryptography and Asymmetric Cryptography

In symmetric key cryptography or primitives, both the sender and the receiver use the same key for the encryption of information and decryption of ciphertext. This requires the secret key to be shared prior separately over a secure channel between the parties involved which can be quite hard to achieve. This is one of the main drawbacks of symmetric key cryptography. [6.]

Asymmetric key cryptography, also known as public key cryptography, uses a pair of keys for each user called a public/private pair during encryption and decryption process. Such key pairs are generated with one-way functions where a private key is supplied as an input to generate a public key as an output. It is computationally infeasible to perform an inverse operation and determine the private key given a public key. Thus, public keys can be distributed openly. The sender encrypts the message with the public key of the receiver which can be decrypted only with the correct private key of the receiver on the other end of the communication network. [6.]

2.2.2 Digital Signatures

One application of asymmetric key cryptography is digital signature. Digital signatures are extensively used in the blockchain to digitally sign the messages and claim its authorship using a private key by its associated party. This provides a strong basis to confirm the authenticity, integrity, and non-repudiation of any messages which is sent over the untrusted distributed network. Messages itself becomes the part of digital signature due to which any attempt to tamper with it makes it obsolete. [8, 6.]

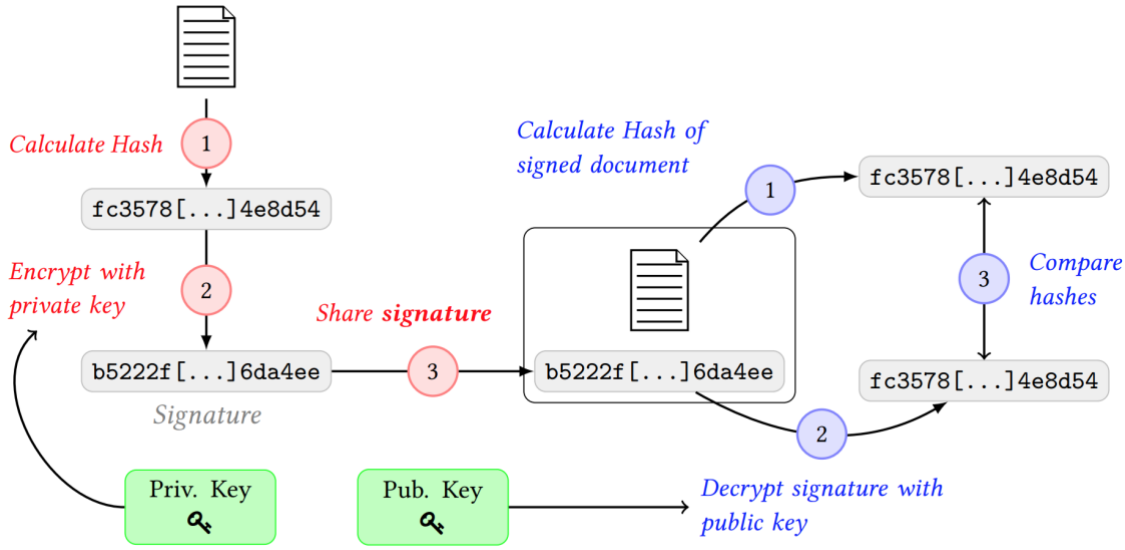

Figure 3. Illustration of the creation of a digital signature (red) and a subsequent verification of the signed document's authenticity (blue). [8, 6]

As illustrated in Figure 3, the authenticity of a signed document can be verified by the receiver upon comparing the hash of the document with hash obtained by decryption of the digital signature using signee's public key. If the two-hash values match, it provides

proof that the document was signed by the owner of the public key who holds the match matching the private key.

2.2.3 Hash Functions

A hash function is such a type of function that takes an input data of arbitrary length, performs an operation on it and outputs data of fixed length which is commonly referred to as hash values, digests or hashes. Hash functions work keyless meaning that no key is involved while generating the digests compared to symmetric and asymmetric cryptography. These functions are one-way functions to create other cryptographic primitives. Following are the important properties that any cryptographic hash functions must fulfill in order to be considered secure: [6.]

- Deterministic
  A hash function should always produce the same hash for the same input each time. [6.]

- Quick Computation
  A hash function must be very quick to produce hash regardless of input size. [6.]

- One-way/Pre-image resistance
  Given an input a, a computed hash b, and a hash function h, such that $b = h(a)$, it must be computationally infeasible for an attacker to calculate the correct input *a* from the given hash value b. a is considered to be the pre-image of b. [6.]

- Second pre-image resistance
  Given an input p, a hash function h, it must be computationally infeasible for an attacker to calculate other input q such that $h(p) = h(q)$ where $p != q$. [6.]

- Collision resistance
  Hash of two different inputs x, y should not be same i.e. $h(x) != h(y)$. [6.]

Hash functions are used to generate the hash of each block in blockchain which gives it a unique identity. A new block is linked to the previous block by a hash of the previous block forming an immutable chain of blocks. Hash functions also play a key role in consensus algorithms of blockchain which is discussed in section 2.4. Thus, the integrity

of the blockchain is assured. Message Digest Algorithm 5 (MD5), Secure Hashing Algorithm (SHA-1, SHA-2, SHA-3), RACE Integrity Primitives Evaluation Message Digest (RIPEMD) are some examples of the hash functions available to use. [6.]

2.3 Introduction to Blockchain Technology

A blockchain can be defined as an immutable, decentralized and distributed ledger on which transactions are recorded in the chain of blocks. These blocks are linked together with a cryptographic hash upon reaching the consensus by the participating peers of the peer-to-peer blockchain network. Once a block is added to the ledger, it is practically impossible to change it until all the subsequent blocks are also altered with the consensus among all the peers for the change. The copy of ledger resides on each participating peer of the network. As illustrated in figure 4, each organization controls a peer node which participates in consensus and maintains the ledger in the blockchain network. [6.]

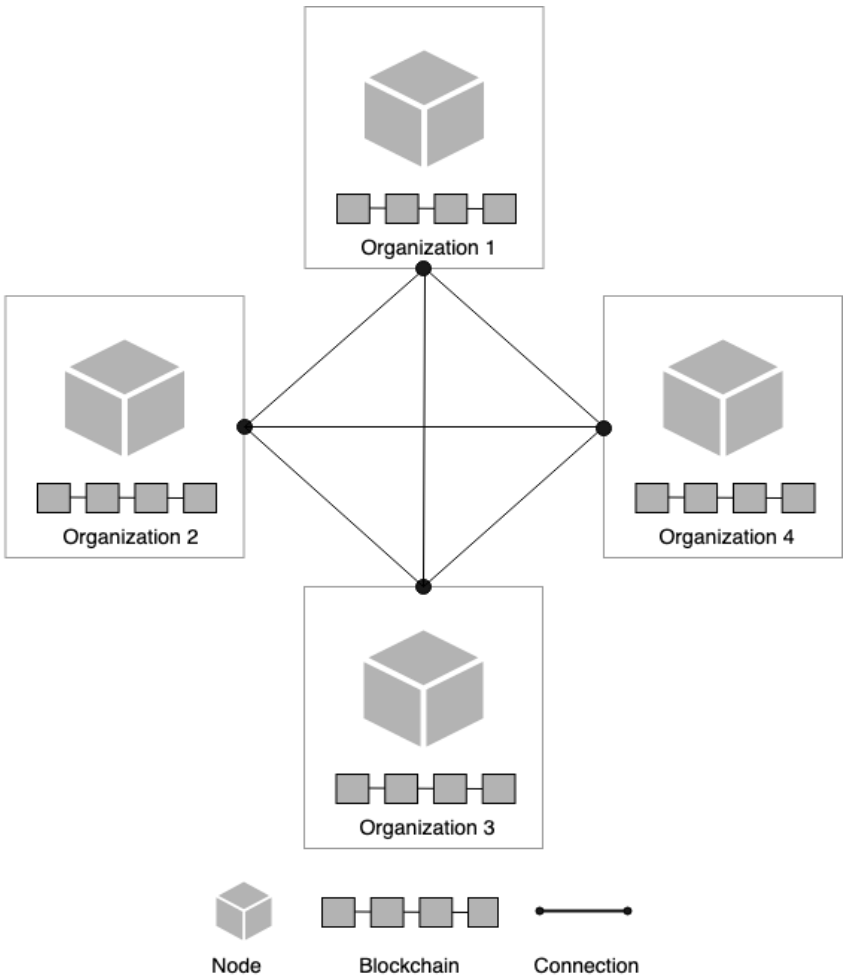

Figure 4. Simple blockchain network

A typical block of the blockchain consists of a block header, multiple transactions: the smallest units of data that can be stored in the blockchain and block metadata. Block header contains a cryptographic hash of the previous block except for genesis block which is the first block of a blockchain and it does not refer to a previous block as shown in figure 5. Genesis block is used as a starting point to build the chain of blocks upon and hardcoded into the system. These cryptographically linked blocks form the unbreakable chain of blocks. [9.]

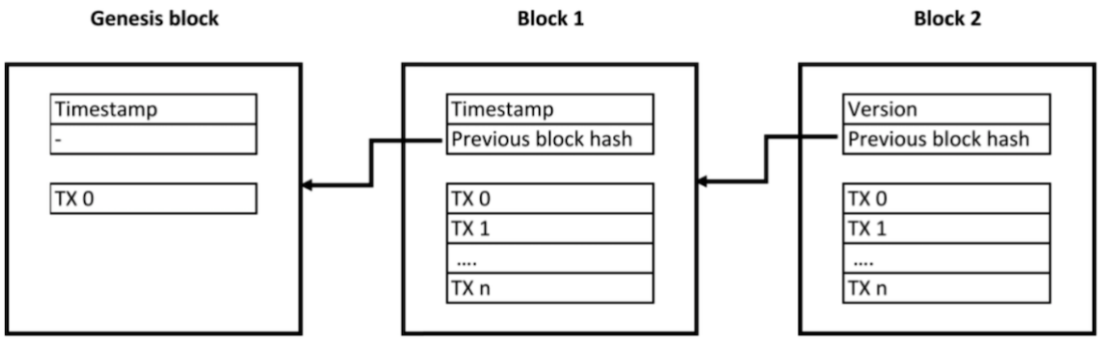

Figure 5. Chain of blocks linked together by a cryptographic hash. [6]

In order to fulfill the needs of different users, business, and industries, various blockchain solutions have emerged over the past years with a varying set of protocols. Despite the differences found in different blockchain solutions, the foundation still remains the same. All these solutions can mainly be categorized into two types: permissionless and private blockchain.

2.3.1 Permissionless (Public) Blockchain

Permissionless blockchain networks are open to everyone where any participants can enter and leave the network at their will. No central authority is involved in the management of the network and membership of the participants. It is truly decentralized. All transactions are also visible to the public. Real identities of the participants are concealed with private-public cryptographic keys. Currently, the process to reach consensus among peers on new information, which has to be added to the ledger, is resource intensive and time-consuming. As a result, Permissionless blockchain is often slower than Permissioned blockchain. Some of the popular Permissionless blockchain solutions are explained briefly below: [9.]

- Bitcoin

  Bitcoin is the first decentralized digital currency that individuals can directly trade with each other without the need for intermediaries. It is not issued and controlled by any central authority such as banks or government. It is the first currency to solve a difficult problem of double-spending without the need for centralized authority. Validation and confirmation of each bitcoin transaction is done by the entire bitcoin network which is then stored in the immutable ledger on each running nodes. All transactions are visible to the public. The first node to

successfully create the new block is rewarded with bitcoins. Such incentives keep Bitcoin's network functioning. [10.] The byproduct of bitcoin is its revolutionary underlying technology called 'Blockchain' which has attracted a large amount of attention in recent days that could be applied in various other sectors to establish trust and transparency digitally.

Bitcoin's architecture is based on a white paper titled 'Bitcoin: Peer-to-Peer Electronic Cash system' posted to a cryptography mailing list on 31st October 2008 by an anonymous person or a group of persons named Satoshi Nakamoto. On 3rd January 2008, Nakamoto mined the first block of the bitcoin and was made available to the public. The real identity of Nakamoto is still unknown. [11.]

- Ethereum

  Ethereum provides a platform for users to build and deploy distributed applications (DApps) on public distributed Ethereum network, leveraging the power of blockchain technology. In contrast to Bitcoin where blockchain is particularly used to decentralize money, Ethereum expands the horizon with the possibility to decentralize anything that can be resembled by code. Ethereum platform has thousands of nodes independently running all over the world which executes deployed application, also known as Smart contract, and maintains the immutable ledger. The developer must pay 'Ether', a type of cryptocurrency, to deploy Smart contract to the Ethereum platform. Such payments help to keep the Ethereum's large network infrastructure running. Ethereum was first proposed in late 2013 by Vitalik Buterin and made publicly available in 2014. [12.]

2.3.2 Permissioned (Private) Blockchain

Permissioned blockchain networks are centralized and regulated by an authority or a consortium of authorities. Only with the right certificates and permission issued by respective authorities, participants can join and interact with the network. Transactions are validated by trusted peers whose identity is well known. Since the scope, users, and participants of Permissioned blockchains are limited, transactions are processed relatively faster than Permissionless blockchain. Multichain, BigchainDB, HLF, Hyperledger Sawtooth are some of the examples of Permissioned blockchain solutions. [9.]

- Multichain

  Multichain is an open source platform for the creation and deployment of private, permissioned blockchains. It is forked from Bitcoin core and hence uses Bitcoin's protocol, transaction, and blockchain architecture. It can run on Linux, Mac and Windows servers. Like Bitcoin, it deals with only the transfer of assets such as cryptocurrencies, so there is no possibility to embed complex business logic. Smart contracts cannot be written for this platform to be executed by nodes of the network. Public key cryptography is used to manage user permissions. Multichain allows users with sufficient permissions to create and work with multiple blockchains at the same time. This important feature also gives the possibility to apply restriction for users on different blockchains. Hash of the genesis block and configurable parameters uniquely identify each blockchain running in this platform. [13.]

- BigchainDB

  BigchainDB is known as 'blockchain database' software which combines the benefits of blockchain and database. It aims to provide the features of blockchain such as decentralization, immutability, owner-controlled assets as well as databases such as high throughput, low latency and high capacity. The first version (0.1) of the software was released in February 2016. However, it was not Byzantine Fault Tolerant (BFT), which is discussed in section 2.4.3, and prone to a single point of failure as a single master node does all the writes in the network. To overcome these issues, the newer version (2.0) was released in 2018. It uses Tindermint's BFT consensus algorithm that keeps network functioning even if one-third of the nodes gets compromised and synchronizes the data between all the nodes. Data is structured as an asset in BigchainDB which can characterize any physical or digital object. The system allows CREATE and TRANSFER transactions only on the assets. MongoDB is used as a database of choice to store transactions. Smart contracts cannot be executed on this platform but can be stored. BigchainDB can be connected with other blockchain platforms which run smart contracts like HLF via oracles or inter-chain communication protocols. [14.]

2.4 Consensus Mechanisms

Blockchain network consists of multiple distributed nodes which plays a key role for maintaining the same state of the ledger. Participating nodes may accept and add a new validated block to their copy of ledger when broadcasted to the network or reject it. There is always a probability of the existence of malicious or faulty nodes which tries to compromise the integrity of the network. Consensus mechanism is a way to achieve consensus on a proposed state, even in the presence of adversaries nodes, and keep the ledger synchronized, following a set of rules by the participating nodes. It is considered as the soul of any blockchain network. Achieving consensus to a single version of the truth by majority of the peers is extremely critical and keeps network functioning. Selection of consensus algorithms depends on the type of blockchain in use such as permissioned or permissionless blockchain. [6.] Consensus mechanisms can be categorized as [8.]:

2.4.1 Computation Power: Proof-Of-Work

In Proof-Of-Work (PoW) consensus mechanism, a validator node needs to submit the proof of work to publish and broadcast a newly created block to the network [8]. Validator node needs to work on a mathematical problem which is to calculate the hash value that is less than a specific value set by the protocol along with the combination of hash values of previous block data. The solution is costly and time-consuming as it involves brute forcing the solution, but the validation of the solution is quick. As soon as the solution to the given problem is solved by any validator nodes, it is published to the network as a proposed block. Participating nodes validate the solution and other rules to reach the consensus and if correct, add it to their ledger. At this point, the validator node gets rewarded with some currency for the work done to find the solution and as an incentive to keep the work going. This whole processing is also termed as ‘Mining’ and validator node as ‘Miners’. Double-spending problem where same digital currency can be spent more than once is solved through consensus mechanism by maintaining the single source of truth. The system implementing this consensus mechanism is resistant to Distributed Denial-of-Service (DDoS) attack because it cannot be known beforehand which node in the network will be able to solve the puzzle first. PoW is still theoretically vulnerable to 51% attack that can result in double spends. [15.] Bitcoin [10], Ethereum [12] use this consensus mechanism.

2.4.2 System Stake: Proof-of-Stake, Delegated-Proof-of-Stake

Using the PoW consensus mechanism, mining of new blocks gets computationally expensive with the growing difficulty target of the mathematical problem. Miners need to compete with each other to be the writer of the next block on which a significant amount of computational resources and electricity is being wasted. It is estimated that - in 2013 – energy (electricity) consumed by bitcoin mining (operational cost of CPUs and cooling system) equaled that of the country Ireland. [8.]

Consensus mechanisms such as Proof-of-Stake (PoS) or Delegated-Proof-of-Stake (DPoS) mitigate such high operating cost of mining by allowing nodes or users to stake the token they own in the system in order to create new blocks, removing competition between miners. The probability of creating a new block and getting a reward increases with the increase in stake. Casper is Ethereum's version of PoS which is currently under active development. DPoS is different from PoS in that it is permissioned by stakeholders. Stakeholders do not take part in creating new blocks in DPoS but votes for delegates/witnesses who are responsible to create new blocks. Witnesses get paid upon creation of a new block each time. It was first proposed and used by BitShares in 2014. [8.]

2.4.3 Inter-Network Relationships: Practical Byzantine Fault Tolerance

BFT is the ability of a distributed computer network to correctly reach consensus on a single truth despite the presence of malicious node or failure of nodes. BFT assumes some of the nodes involved might be unreliable or corrupt. Practical Byzantine Fault Tolerance (PBFT) is a consensus algorithm that is able to tolerate Byzantine faults using state machine replication. It can function effectively if faulty nodes do not exceed one-third of the total nodes available. It was presented by Miguel Castro and Barbara Liskov in a paper released in 1999. [16.] It is used by HLF up to release v0.6-preview and later replaced by Kafka orderer which is fault tolerant [8].

2.5 Smart Contracts

A smart contract is an agreement made between parties represented in a computer program in the form of business logic which is executed automatically when all the conditions are satisfied. The term 'Smart contract' was first coined by Nick Szabo, a

computer scientist, and cryptographer, in 1994. [6.] Smart contracts are self-enforcing, which means code composing it should be treated as a law and does not need to be legally enforceable. They should not rely on traditional methods of enforcement like government or any corporate law. The need for the trusted third parties becomes obsolete. [17.] The following are the four main objectives of the smart contract design:

- Observability
  Principals (parties who signed the contract) are able to observe each other's loyalty to the contract [17].

- Verifiability
  Principals can prove to the arbitrators whether a contract has been accepted or rejected [17].

- Privity
  The knowledge and control over the contents and performance of a contract should be distributed among parties as much as necessary [17].

- Enforceability
  It refers to the ability to make the contract self-enforcing with improved verifiability, self-enforcing protocols, built-in incentives [17].

2.6 Platform Selection

After careful analysis, HLF is chosen as a platform of choice for deploying and operating the blockchain application. The main reasons for the selection are:

- HLF is a private, permissioned blockchain platform. Interaction with the network is not possible without the valid identity of the participant issued by designated Certificate Authority (CA) [18]. This is a very critical feature as we do not want the charge records to be accessed by anyone using the internet.
- Cryptocurrencies are not used as a part of transactions in the HLF due to which mining of cryptocurrencies is not required reducing the consumption of significant computing resource and time compared to Ethereum, Multichain as discussed in 2.3.1 [18]. Because of this, HLF can handle high transactions rates that is required when thousands of EV Charging stations interact with the network.

- HLF is designed with the unique component based extensible modular architecture for consensus mechanisms, membership provider, data storage [18]. For example, Solo Ordering service can be used as a consensus algorithm which is easy to maintain and work during development and can be replaced with Kafka Ordering service for production use.
- Data is synchronized and stored in all the participating nodes of the network [18].
- General purpose programming languages can be used to write Smart contracts like Go, Node, and Java. It does not require learning new platform-specific programming languages for development like Solidity for Ethereum. [18.]

3 Hyperledger Fabric

Hyperledger Fabric is an open-source platform for developing private, permissioned blockchains. It is one of the projects within Hyperledger projects, maintained by Linux Foundation. In comparison to the public or permissionless blockchain solution where any participants can join the network without specific identity and restriction, HLF platform provides mechanisms to impose a restriction on participant's rights and access to the network. It requires the identity of participants to be known. This platform neither uses any cryptocurrency nor provides economic incentives for the participants responsible for running the network or validating transactions. [19.]

HLF supports modular and pluggable architecture pattern due to which it becomes possible to combine different blocks as per the business requirement during the development of blockchain solutions. Some of such pluggable components include the membership service provider, consensus engine. Creation of digital assets and management of its state to be stored in HLF blockchain can only be done by invoking transactions defined in chaincode that can be also referred as a Smart Contract. All the transactions history is recorded in an append-only replicated ledger securely. Chaincode includes all the business logic which is installed onto the HLF peers but runs in a separate isolated docker process from peers. Domain-specific languages are not required to write chaincode as Solidity for Ethereum, but it can be written with general purpose languages like Go, JavaScript or Java. [19.]

All in all, the HLF platform is tailored for business organizations who can leverage the power of innovative blockchain technology into their services without compromising confidentiality and security.

3.1 Components

HLF is composed of different modular components that can be customized according to the need of the enterprise. These software components are discussed below [18]:

3.1.1 Membership Service

The Membership service is responsible for managing the identity, authentication, and authorization of all the nodes and users interacting with the permissioned blockchain

network. Membership Service Provider (MSP) in HLF makes use of CA in order to support identity management and authorization operations. By default, HLF has stand-alone CA which is called as Fabric-CA, but it can be replaced with any other commercial certification authorities. Fabric-CA handles standard Public Key Infrastructure (PKI) methods for authentication based on digital certificates, X.509, issued to members of the different organizations and clients. Each member organization receives one root certificate. Fabric-CA can also issue temporary certificates to the clients which can be used only for one-time transactions. It runs inside a Docker container and uses SQLite as a default database to store the issued certificates. [19.]

3.1.2 Ordering Service

The Ordering service consists of nodes that are responsible for receiving the executed transactions from the peers, order and combine them into blocks and broadcast them to all the peers on the same channel. These nodes are also referred as Ordering-Service Nodes (OSNs). Blocks received from the Ordering service is validated by each peer before it is committed to the ledger. Information related to genesis block is provided to the Ordering service during bootup process. Currently, it can be implemented in three different ways: A Solo orderer runs on a single node that can be used only for development and testing purpose. Apache Kafka orderer is production ready Ordering service offering scalable, high-throughput, low-latency publish-subscribe messaging platform for the connected peers. It provides strong data consistency in case of failure of nodes. Thus, Kafka orderers are Crash Fault Tolerant (CFT) but not BFT. An experimental orderer based on Byzantine Fault Tolerant State Machine Replication (BFT-SMaRT) has also been made available for testing. [18.]

3.1.3 Peer

Peer is a network entity owned and maintained by the members of the blockchain network. Peers maintain the state of the distributed ledger by each holding the copy of ledger locally. They also deploy, instantiate and interact with chaincodes. Peers communicate via gossip protocol to broadcast ledger and channel information. Endorsing and Committing peers are two different types of peers currently implemented in HLF, which is explained in more detailed in section 3.2 [20.]

3.1.4 Ledger

A ledger in HLF keeps the sequential, immutable records of state transitions done through committed transactions. Each record is stored as key/value pair in the ledger. The ledger is comprised of two parts: Chain and State database. Chain is an immutable transaction log structured as blocks that are cryptographically linked together. A sequence of transactions is stored in each block. LevelDB is used as a Chain database. State database represents the current state of the data or assets stored in the ledger. It is mutable and supports create, read, update and delete (CRUD) operations. LevelDB is also used as a default state database embedded in the peer process but can be replaced with CouchDB instead. Each peer in the network maintains the copy of the ledger for each channel of which they are a member. [19.]

3.2 Architecture

The traditional blockchain platforms follow the order-execute architecture for transactions such as Bitcoin, Multichain and Ethereum. During the process of creation of the new block, transactions are ordered first and then executed sequentially by all the peers of the blockchain network. Such sequential execution of transactions on all peers limits the effective throughput of the blockchain network and may become a performance bottleneck. In addition, this architecture does not support developing smart contracts using general purpose languages like Go, Java because deterministic execution of code is not ensured. For this reason, most of these blockchain platforms have their own domain-specific language for writing smart contracts like Solidity for Ethereum, Ivy for Chain. [18.]

To solve the problems posed by order-execute architecture, as illustrated in figure 6, HLF introduces a new innovative execute-order-validation blockchain architecture [18].

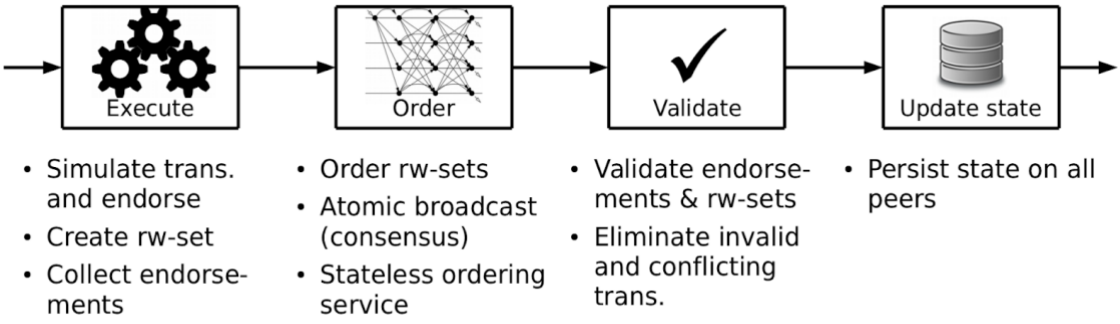

Figure 6. Execute-Order-Validation architecture of Hyperledger Fabric. [18, 5]

Nodes of the HLF network are grouped into three roles, based on the functionality they perform, for the implementation of the execute-order-validation architecture. They are explained as follows:

- Clients
  The client nodes are responsible to submit transaction proposal invoked by users of the blockchain to endorsing peers. Then, they broadcast the received endorsed transaction from endorsing peers to the Ordering service if the endorsement policy is fulfilled. [18.]

- Peers
  All the peers in the HLF network are responsible for maintaining the state and the ledger. Peer nodes are further divided into two roles. [18.]

    - Endorsing peers
      The endorsing peers receive the transaction proposal request for an endorsement from clients. They simulate the transaction and either endorse or reject it based on the current state of the ledger. [18.]

    - Committing peers
      The committing peers receive new ordered blocks from the Ordering service for committing to the ledger which they maintain. Transactions contained in the block is validated by these peers. [18.]

- Orderers

Orderer nodes are responsible to order the received, signed or endorsed transactions from the clients chronologically and create a new block. A newly created block is then distributed to all peers of the network. Orderer nodes are not aware of the state of the ledger so they do not engage themselves in the execution and the validation of transactions. [18.] Figure 7 illustrates the coordination of multiple orderers between all the peers in the blockchain network involving different organizations.

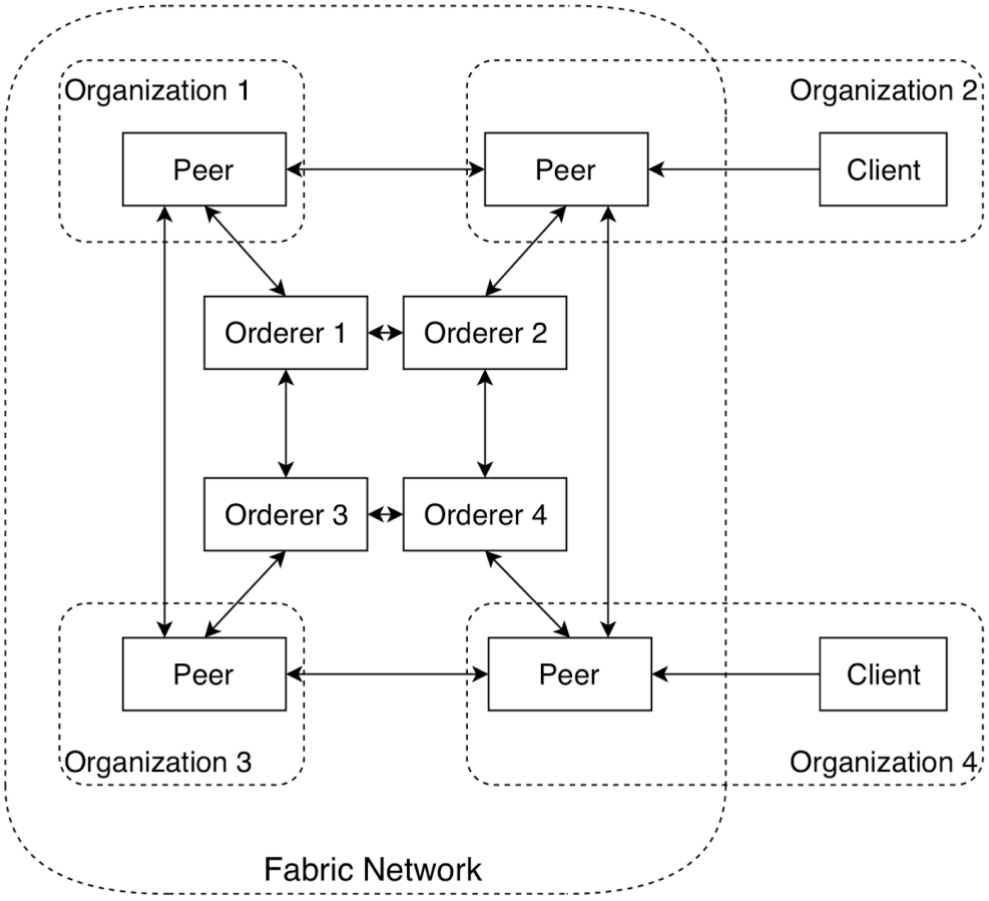

Figure 7. Sample network layout of nodes in Hyperledger Fabric. [21, 9]

Not all the peers of the network are required to execute all transaction proposals as it is only done by the peers who are titled as endorsing peers. Executing transaction before the ordering phase solves the non-deterministic problem arisen by using general purpose languages for developing smart contracts. Division of operations of execution, ordering and validation among peers of the network boots the performance and independent scalability of the system. [18.]

3.3 Transaction Flow

All transactions that interact with HLF blockchain system are processed based on the execute-order-validate architecture. The transaction flow that occurs between nodes which are assigned different roles in HLF is illustrated in figure 8.

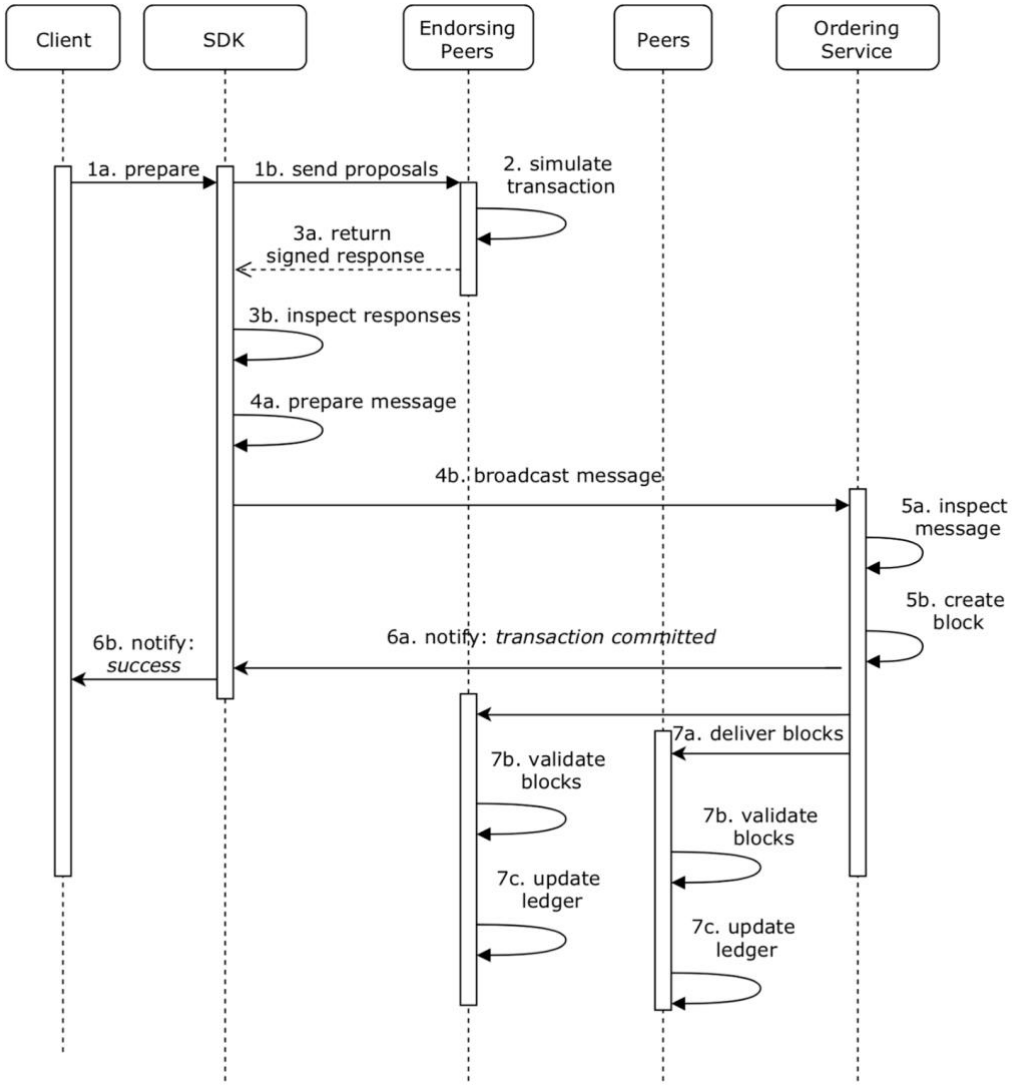

Figure 8. Normal transaction flow in Hyperledger fabric. [21, 11]

Step 1

Transaction proposal is signed and sent to endorsing peers for execution by the client. A signed proposal contains the identity provided by the MSP of the submitting client, transaction payload, chaincode identity, a nonce (a counter or a random value) to be used only once by each client, and transaction identifier derived from the client identifier and the nonce.

Step 2

Endorsing peers execute the received transaction proposal on the specified chaincode against the current state of the ledger. Execution or simulation of the transaction does not update the ledger.

Step 3

The result of execution is sent back in a signed proposal response by endorsing peers. The client collects all the endorsement to verify the endorsement policy invoked by the transaction.

Step 4

The client creates the transaction and submits it to the orderers.

Step 5

Orderers group the submitted transactions and creates a new signed block. The newly created block is then distributed across all the committing peers.

Step 6

The client is notified about the success of the transaction proposal request which is now committed to the blockchain.

Step 7

All peers in the network perform validation of each transaction within a received block from the Order and commit changes to the ledger.

4 Implementation

As a proof of concept and part of this thesis, a blockchain network, a chaincode and an API server were developed to store immutable EV charge records for Liikennevirta Oy. HLF version 1.1 is used as a blockchain framework and the Hyperledger Composer version 0.19 is used as a tool to develop the prototype. This chapter describes the implementation details of the developed network such as design, architecture and data models, the chaincode and the API server that interacts with the network.

4.1 Network Architecture

The developed prototype of the network consists of one-member organization (Virta), three peer nodes with their own copy of the ledger, one SOLO orderer and an API server. The network runs in an isolation inside Virta's private network. Thus, it is not accessible from outside world. The architecture of the blockchain network is illustrated below in Figure 9.

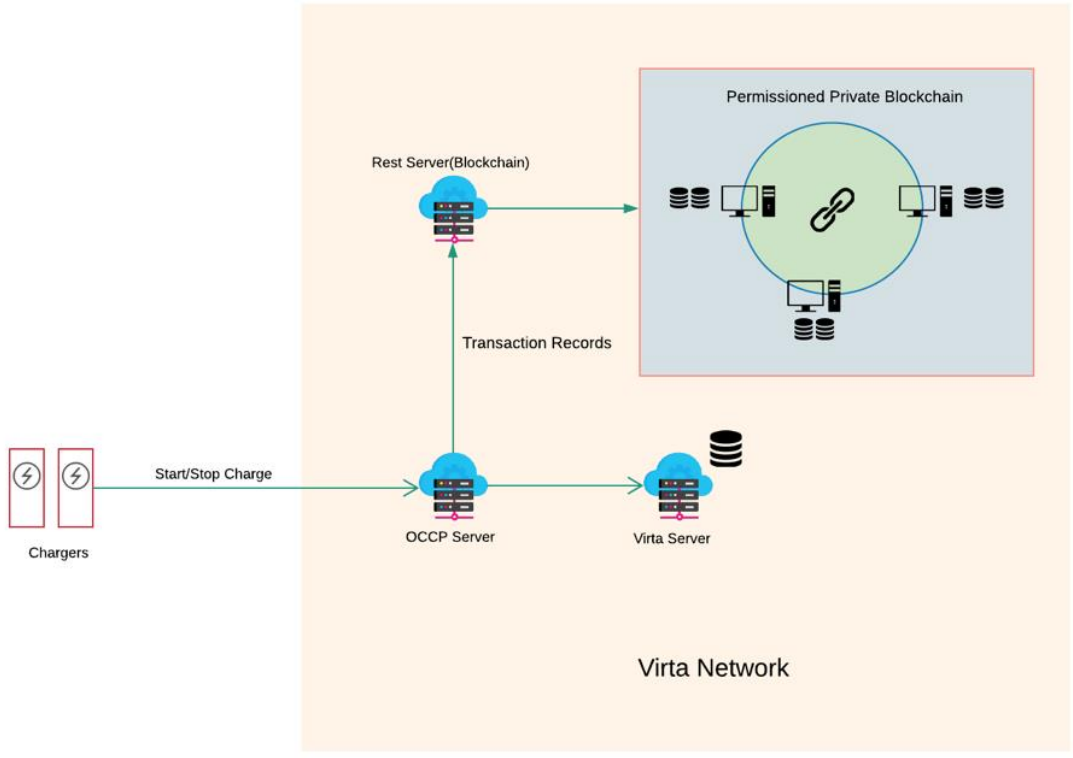

Figure 9. Network Architecture of the blockchain prototype

Among different responsibilities, the Open Charge Point Protocol (OCPP) server acts as a bridge to forward start/stop charging event messages received from EV charging stations to the blockchain API server as shown in the figure 9. Blockchain API server submits transactions to the blockchain network for validation and storage. Transaction records are distributed, stored and synced to all of the three different nodes by the orderer. At the time of writing this thesis, direct communication between EV charging stations and the blockchain network was not possible as EV charging stations cannot be connected and controlled simultaneously by multiple backend platforms.

All network entities are configured to run on the single machine using the Docker containers. *docker-compose.yaml* file contains configuration details to create the required containers. X.509 Certificates and signing keys needed for authentication, communication and transaction between various network components are generated, using configurations defined on *crypto-config.yaml* file, by the Crypto Generator tool, *cryptogen* [19]. The Configuration Transaction Generator, *configtxgen,* tool generates an orderer genesis block, a channel configuration transaction and anchor peer transactions using configurations defined on *configtx.yaml* file [19]. Network artifacts generated upon running these tools should never be lost or changed. Currently, artifacts are generated manually by running these tools using scripts before starting the network.

4.2 Application Development with Hyperledger Composer

Hyperledger Composer is an open development toolset, framework and modeling language that provides a simplified and faster approach to develop chaincodes and applications for blockchain platforms such as Hyperledger Iroha, Hyperledger Sawtooth, HLF [22]. Its usage in the development of the chaincode and API server, for the prototype, is described in the following sections.

4.2.1 Chaincode (Smart Contract) Development

Hyperledger Composer provides higher business-level abstractions allowing to quickly model the business network using assets, participants, and transactions related to them, reducing a need to use lower-level APIs provided by HLF while the development of chaincode. It introduces the concept of Business Network Definition (BND) which consists of model definition, business logic, queries, and permissions related to the application being developed. All these different components, placed in separate files with

the specific file extension, are packaged in a Business Network Archive or a banana (BNA) file that can be deployed to HLF runtime as shown in the figure 10. [22.]

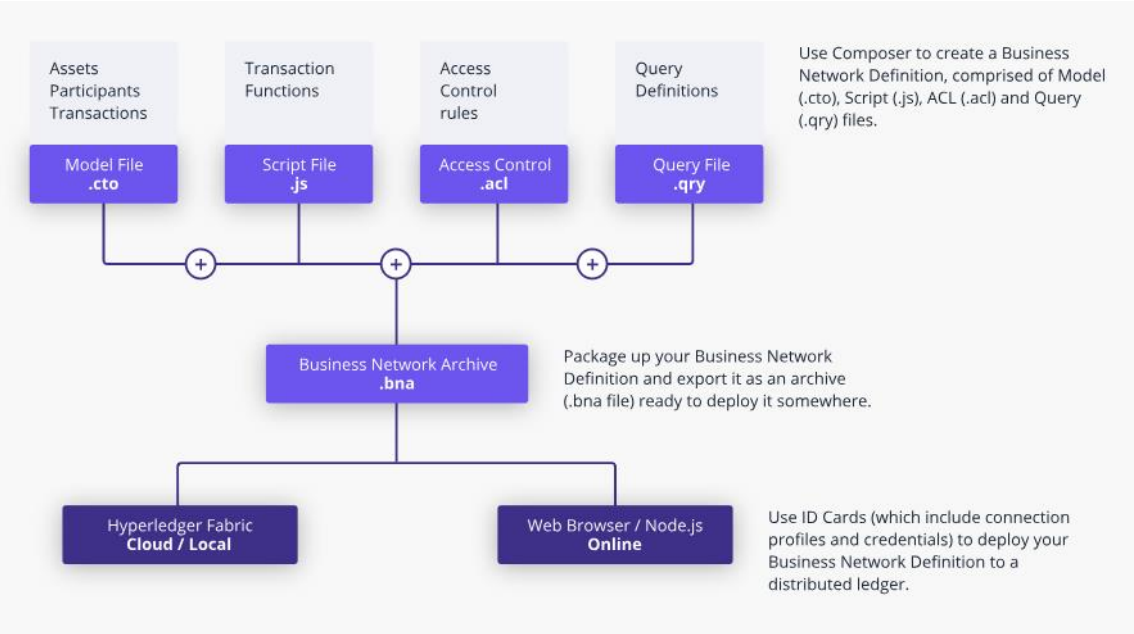

Figure 10. Business Network Definition structure. [22]

The core components of BND are described as follows:

- Model

  A Model file consists of the definition of resources present in the business network such as assets, transactions, participants and events. It has a file extension of *.cto*. A Composer Modeling language is used to define the structure of the Model. Elements present in a Model file are explained below. [22.]

  - Assets

    Assets represent any object of value in the real world. State of an asset may change over time. [22.]

```
asset Charge identified by chargeId {
  o String chargeId
  o String identifier
  o Integer connectorId optional
  o IdToken idToken optional
  o Integer meterStart optional
  o Integer reservationId optional
  o DateTime timeStamp
  o Integer meterStop optional
  o Integer transactId optional
  o Reason reason optional
}
```

Listing 1. Example of an Asset

Listing 1 indicates a 'Charge' asset used in the prototype to represent an EV charge record.

- Participants

  A participant might represent an individual or an organization of a business network. They can create and exchange assets by submitting transactions. An identity document is issued to each participant which is used in order to interact with the network. [22.]

```
participant Person identified by phonenumber {
  o String phonenumber
}
```

Listing 2. Example of a Participant

Listing 2 shows the definition of Participant using Composer Modeling Language.

- Transactions

  Participants interact with assets using transactions. Transaction is responsible to change the state of an asset. [22.]

```
transaction StartCharge {
  o String identifier
  o Integer connectorId
  o IdToken idTag
  o Integer meterStart
  o Integer reservationId optional
  o DateTime timeStamp
}
```

Listing 3. Example of a Transaction

Listing 3 indicates the definition of *StartCharge* transaction that is invoked to create new assets in the prototype.

- Business Logic

  Transaction processor function implements Business logic related to the Transactions defined in Model file. In this function, both the APIs of Hyperledger Composer and HLF can be used. Hyperledger Composer access control rules

are bypassed if HLF APIs are used. It is written in separate JavaScript file other than Model file. [22.]

```
/**
 * Start Charge Transaction
 * @param {org.virta.global.StartCharge} chargeData
 * @transaction
 */

function startCharge(chargeData) {
  return getAssetRegistry('org.virta.global.Charge')
    .then(function(chargeRegistry) {
      var factory = getFactory();
      var NS =  'org.virta.global';
      var chargeId = generateChargeId(chargeData.identifier,
     chargeData.timeStamp)
      var charge = factory.newResource(NS,'Charge', chargeId); // Hard
     coded   identified by placeholder

      charge.identifier = chargeData.identifier;
      charge.connectorId = chargeData.connectorId;

      var token = factory.newConcept(NS, 'IdToken');
      token.idToken = chargeData.idTag.idToken;

      charge.idToken = token;
      charge.meterStart = chargeData.meterStart;
      charge.reservationId = chargeData.reservationId;
      charge.timeStamp = chargeData.timeStamp;

      return chargeRegistry.add(charge);
    })
    .catch(function (error) {
      throw new Error('Error in startCharge transaction: ', error);
    });
}
```

Listing 4. Example of a Transaction processor function

Listing 4 indicates that the function is a transaction (*@transaciton*) that runs when the transaction *org.virta.global.StartCharge* defined in the model file is invoked.

- Queries

  Composer Query language allows to write queries using different criteria to filter the results from the ledger. They are defined in a query file that must be called *queries.qry*. API endpoints are also generated for the queries by Composer REST Server which is shown in figure 11. [22.]

```
query AllChargingHistoryfromSpecificDate {
  description: "Get all charging history from specified date"
  statement:
    SELECT org.virta.global.Charge
      WHERE (_$fromDate <= timeStamp)
}
```

Listing 5. Example of a query

Listing 5 indicates a query *AllChargingHistoryfromSpecificDate* to get all the charging history from the specific date.

- Access Control List

  Hyperledger Composer's Access Control List (ACL) allows to setup rules to determine which users/roles are permitted to perform CRUD operations on elements of a domain model. The access control file *.acl* contains the definition of the rules for a business network. Rules are always evaluated from top to bottom. Subsequent rules are not evaluated after the first match of the rule which makes scanning of the decision table faster. If no ACL rule is fired, transaction gets denied. [22.]

```
rule OwnerRule {
    description: "ALLOW CURD FOR ASSET OWNER"
    participant(p): "org.virta.global.User"
    operation: ALL
    resource(r): "org.virta.global.Charge"
    condition: (r.owner.getIdentifier() == p.getIdentifier())
    action: ALLOW
}
```

Listing 6. Example of an ACL rule

Listing 6 indicates a ACL rule that allows any instance of *org.virta.global.User* to perform ALL CRUD operations on all the objects of *org.virta.global.Charge* only if the participant owns the asset.

4.2.2 Hyperledger Composer Rest Server as API Server

Hyperledger Composer Rest Server tool reduces the work to develop REST APIs for client application to interact with the blockchain network. It generates REST APIs from a deployed blockchain business network, which serves as an API server in the prototype. It run as a standalone Node.js process. [22.] Client (OCPP server) interacts with network using those APIs.

> The Hyperledger Composer LoopBack connector exposes a deployed business network to LoopBack so it can generate a REST API for the assets, participants, and transactions in that business network. [23]

It is possible to secure REST server using Hypertext Transfer Protocol Secure (HTTPS) and Transport Layer Security (TLS) [22]. Since the API server will be running inside

private trusted network, such security has not been configured but must be turned on before production use.

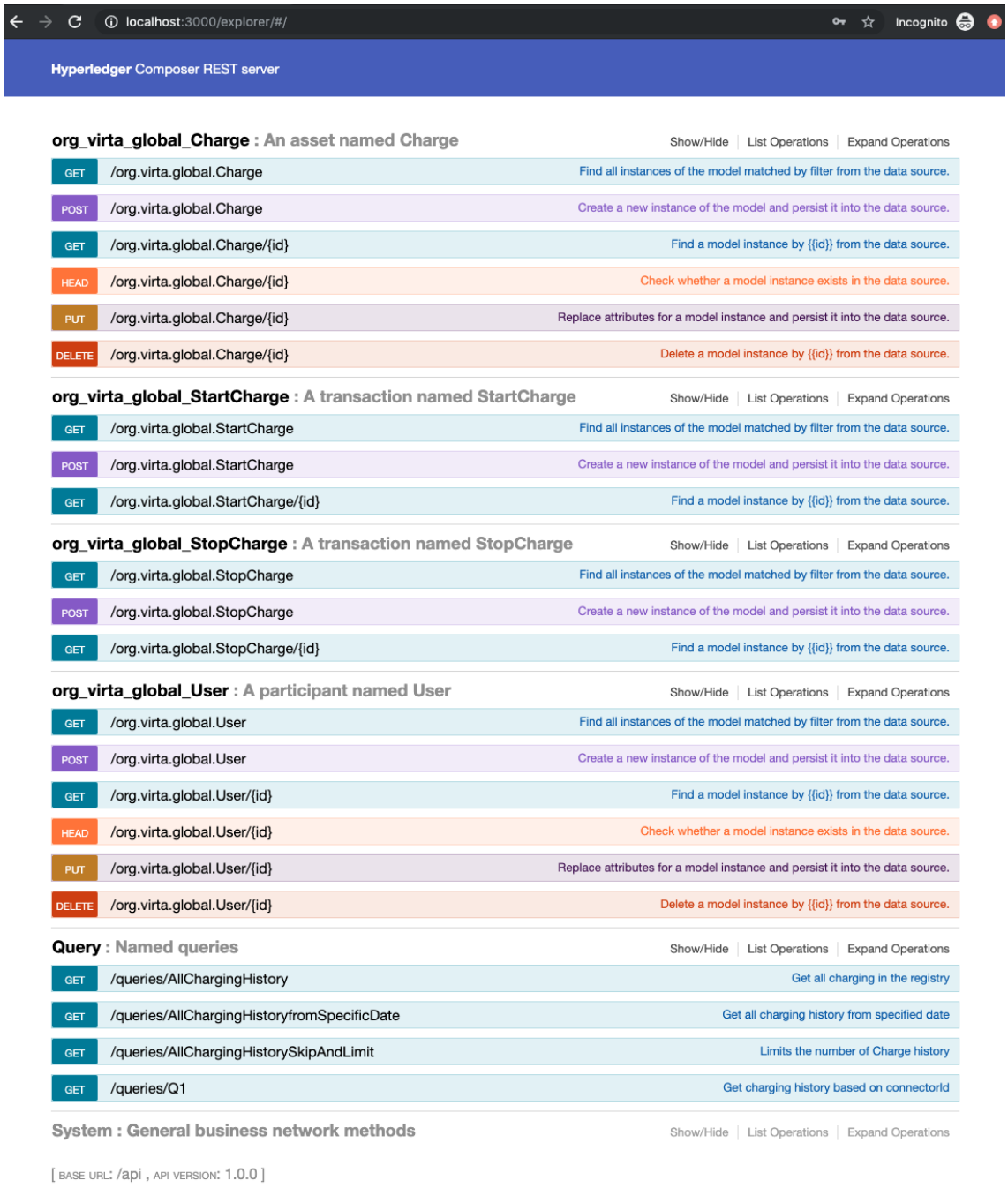

Figure 11. APIs generated using Hyperledger Composer REST server

Figure 11. shows the documentation of REST APIs generated using the Hyperledger Composer REST server. All the assets, transactions, participants have their respective CRUD endpoints defined except for queries as they only need GET endpoint defined.

4.2.3 Hyperledger Composer Playground

As the name suggests, Hyperledger Composer Playground provides a playground with interactive web-based User Interface (UI) to model and test business networks using Composer Modeling Language as shown in figure 12. With the help of this tool, business logic can be tested without deploying applications to the running HLF blockchain network. It allows to edit business network definition, run transactions, import and export BNA files. [22.] It has been really easy and helpful to test the business network definition during its development.

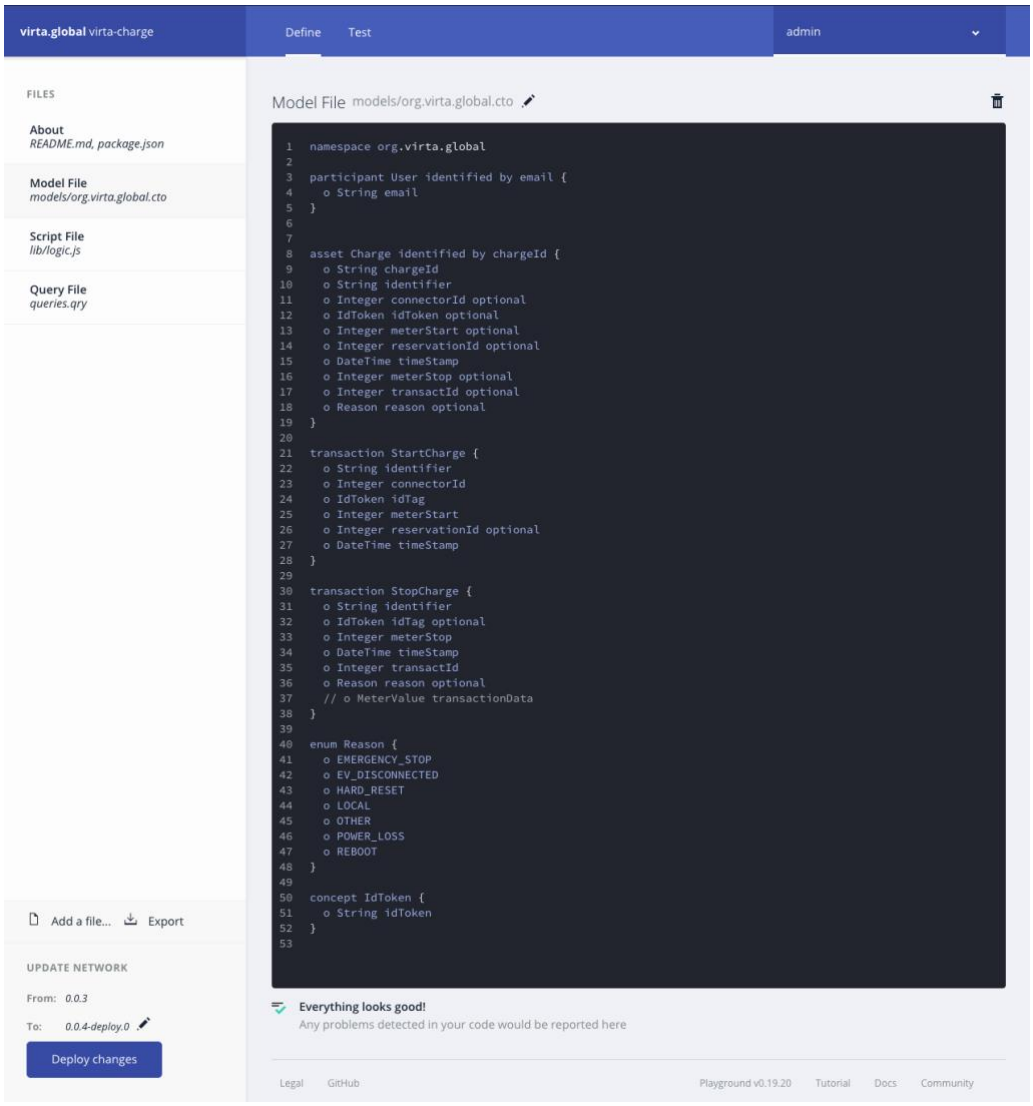

Figure 12. Web based UI of Hyperledger Composer Playground

Hyperledger Composer playground can be installed locally or accessed from the cloud hosted by IBM at *https://composer-playground.mybluemix.net*. It uses browser's local storage to simulate the network when used solely on browser. [22.]

5 Results

The main goal of this thesis was to develop a blockchain based solution to store immutable charge records. Different available blockchain solutions were studied and analyzed in order to find a suitable candidate. The prototype of the solution that achieved the goal of the thesis includes a permissioned blockchain network, a chaincode, and an API server. These are developed using Hyperledger Fabric and Composer technologies.

The blockchain network comprises of a single organization, three peer nodes maintaining their own sets of the ledger, a solo orderer and a Certificate Authority. All these components are configured to run on a single host in a separate containerized environment using Docker containerization technology. Chaincode (Smart Contract) is developed and deployed to the network in the form of .bna files using Hyperledger Composer. For the purpose of testing, the blockchain network and API server are hosted in a single instance of on Amazon Web Services. They are accessible only inside of Liikennevirta's Virtual Private Network (VPN). Client, OCPP server, interacts with the blockchain network using the REST API exposed by the API server to push the charge records received from the Charging stations. However, other applications that are running inside Virta's VPN can also leverage the API endpoints as required.

As the developed prototype is ready only for testing purposes, it should not be used in production. This is because of the current configuration of the blockchain network where all the three peers along with their state databases are running in a single instance which risks the loss of data in case of an instance failure. Similarly, Solo Ordering service is used, which is fine for development and testing, but it should be replaced with Kafka messaging service in production to avoid data loss and ensure fault tolerance [18]. Further work and improvements have to be carried out with the help of Operation team to set up the distributed infrastructure on the Cloud for production use with all the peers, Kafka messaging service, as well as other modular components of the blockchain network, running in separate instances and communicating together. Due to this reason, the performance of the overall system is left to be explored in the real production environment.

6 Discussion

In this thesis, a permissioned blockchain technology, Hyperledger Fabric, was studied. A prototype has been developed where it is being used to store immutable EVs charge records for a single organization, Liikennevirta Oy. Despite successful implementation, there are several concerns that have already been noticed before the prototype is put into the production.

The first and critical concern is about extensive resources required to keep the network running. Production ready HLF blockchain network requires about thirteen instances in the minimal (three PEER nodes, one REST SERVER, two OSN, three Kafka Zookeepers, four Kafka message brokers) running to ensure fault tolerance [19]. Thus, it is not recommended to use blockchain by only a single organization with the sole purpose of keeping records in the blockchain as the cost of development and maintenance of such a system will be expensive. Alternatively, an append-only constraint can be applied to traditional databases tables to store the records which prevent from accidental or purposely deleting of rows of the table, resulting in immutable records as almost when using blockchain technology.

Secondly, HLF technology has been just about two years old as first initial production-ready version 1.0 was released in July 2017 [19]. Rapid changes and improvements in the architecture have been going since then. The maturity of this technology is considered to be low. The process to add a new organization to the network and implementing automation is still complicated. It is recommended to wait for some years until it gets mature enough for production use.

Thirdly, high expertise in Networking is required to set up a distributed production-ready blockchain network but developing the application for the network is simplified due to the possibility to use general purpose languages like GO, Node and Java.

7 Conclusion and Future Works

Blockchain acts as an infrastructure that allows storage of immutable data in a distributed and trustless environment that involves different organization. An effort to use this infrastructure by Liikennevirta, using the prototype developed in this Thesis, to store charging records of EVs helps to gain trust and transparency of customer, regulators as well as auditors. Permissioned blockchain technology is still in its early stage of development and much research and development focusing on performance, automation is required before it can be fully implemented at an enterprise level.

For future works, a new use case of blockchain technology in the field of electric mobility has been noticed during the development of the prototype. With the growth in sales of electric vehicles, the number of Electric Mobility Providers (EMPs) are also increasing whose core responsibility includes management of customers and charging stations in their network. Each EMPs operate on their own way. Customers of one EMP may not be able to charge at charging stations of another charging network controlled by another EMP since there is no possibility for EMPs to share the information of customers and cost of charging. Due to this, trusted third parties like Hubject [24], Gireve [25] are used to handle roaming of customers between charging network controlled by different EMPs so that it will be possible to charge EVs at any of the charging stations. They gather and store all the required information of customers, available charging stations and charge pricing of their partner EMPs at regular intervals. Information of any unknown customers is requested from the third parties when it is not found in the database of EMPs before charging process is allowed.

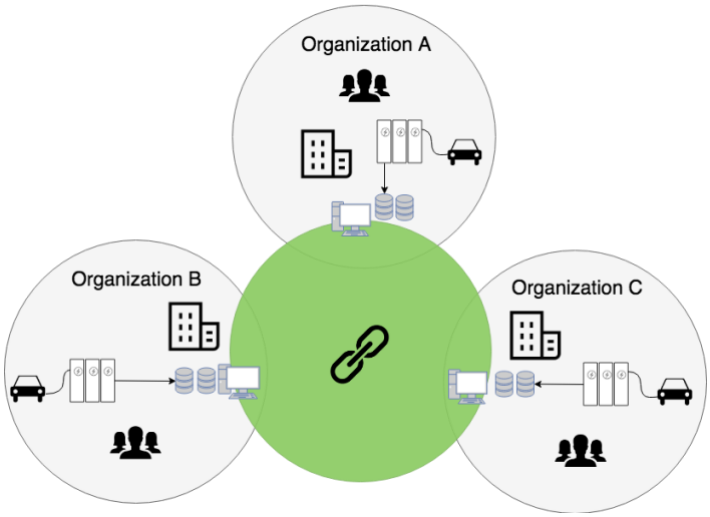

Figure 13. EMPs using blockchain technology to share the information

Permissioned blockchain technology can be used to create decentralized roaming platform by forming an alliance of EMPs. An EMP can be become the member of the roaming blockchain network and run nodes to access the shared ledger with the approval from the alliance as shown in figure 13. Masked information of customers, stations and charging cost could be shared in the blockchain network by all the members of the alliance in an immutable way. It is possible to use smart contracts to carryout financial transactions and signing the roaming contracts between EMPs. Trusted third parties used for the same purpose could now be replaced. There is a possibility to create a 'single global network' of chargers with the direct collaboration between EMPs using blockchain technology without relying on third parties to do so.

References

1 Vince Tabora. Databases and Blockchain, The Difference Is In Their Purpose And Design [online]. Hackernoon.com; 4 August 2018.
URL: https://hackernoon.com/databases-and-blockchains-the-difference-is-in-their-purpose-and-design-56ba6335778b.
Accessed 10 January 2019.

2 Virta. Virta - The Electric Vehicle Charging Company [Online]. Virta.
URL: https://www.virta.global/company.
Accessed 10 January 2019.

3 Maarten van Steen and Andrew S Tanenbaum. Distributed Systems. 3rd edition. distributed-systems.net; 2017.

4 Paul Baran. On Distributed Communication Networks [online]. Santa Monica, California: THE RAND Corporation; September 1962.
URL: https://www.rand.org/content/dam/rand/pubs/papers/2005/P2626.pdf.
Accessed 11 January 2019.

5 Rüdiger Schollmeier. A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications [online]. Linkoping, Sweden. IEEE Xplore; 2001.
URL: https://ieeexplore.ieee.org/document/990434.
Accessed 12 January 2019.

6 Imran Bashir. Mastering Blockchain [online]. 2nd edition. Packt Publishing; 2018.
URL: https://www.packtpub.com/big-data-and-business-intelligence/mastering-blockchain-second-edition.
Accessed 12 January 2019.

7 Alfred et al. Handbook of Applied Cryptography, Chapter 1: Overview of Cryptography [online]. 5th edition. CRC Press; 2001.
URL: http://cacr.uwaterloo.ca/hac/.
Accessed 15 January 2019.

8 Julian Debus. Consensus Methods in Blockchain Systems [online]. Frankfurt School, Blockchain Center; May 2017.
URL: http://explore-ip.com/2017_Consensus-Methods-in-Blockchain-Systems.pdf.
Accessed 30 January 2019.

9 Koshik Raj. Foundations of Blockchain [online]. Packt Publishing; 2019.
URL: https://www.packtpub.com/big-data-and-business-intelligence/foundations-blockchain.
Accessed 10 February 2019.

10 Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System [online]. Bitcoin.org.
URL: https://bitcoin.org/bitcoin.pdf.
Accessed 15 February 2019.

11 Jamie Redman. Satoshi Nakamoto's Brilliant White Paper Turns 9-Years Old [online]. Bitcoin.com; October 2017.

URL: https://news.bitcoin.com/satoshi-nakamotos-brilliant-white-paper-turns-9-years-old/.
Accessed 15 February 2019.

12 Ethereum Homestead. Ethereum Homestead Documentation [online]. Ethdocs.org; June 2018.
URL: http://ethdocs.org/en/latest/index.html.
Accessed 20 February 2019.

13 Dr Gideon Greenspan. Multichain Private Blockchain - White Paper [online]. Coin Sciences Ltd; July 2015.
URL: https://www.multichain.com/download/MultiChain-White-Paper.pdf.
Accessed 25 February 2019.

14 BigchainDB. BigchainDB 2.0 The Blockchain Database [online]. Berlin, Germany: BigchainDB GmbH; May 2018.
URL: https://www.bigchaindb.com/whitepaper/bigchaindb-whitepaper.pdf.
Accessed 26 February 2019.

15 Andrew Tar. Proof-of-Work, Explained [online]. Cointelegraph; 17 January 2018.
URL: https://cointelegraph.com/explained/proof-of-work-explained.
Accessed 27 February 2019.

16 Miguel Castro and Barbara Liskov. Practical Byzantine Fault Tolerance [online]. 545 Technology Square, Cambridge, MA: Massachusetts Institute of Technology; February 1999.
URL: http://pmg.csail.mit.edu/papers/osdi99.pdf.
Accessed 19 May 2019.

17 Nick Szabo. [Online].Smart Contracts: Building Blocks for Digital Markets [online]. Amsterdam: Phonetic Sciences; January 2006.
URL:
http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html.
Accessed 2 March 2019.

18 Elli et al. Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains [online]. IBM; 17 April 2018.
URL: https://arxiv.org/pdf/1801.10228.pdf.
Accessed 1 February 2019.

19 Hyperledger Fabric. Hyperledger Fabric Documentation [online]. Linux Foundation; July 2018.
URL: https://hlf.readthedocs.io/en/release-1.1.
Accessed 15 January 2019.

20 Petr et al. Blockchain Development with Hyperledger [online]. Packt Publishing; March 2019.
URL: https://www.packtpub.com/big-data-and-business-intelligence/blockchain-development-hyperledger.
Accessed 15 May 2019.

21 Stefanos Georgiou. A Trustworthy Process-Tracing System for B2B-Applications based on Blockchain Technology [online]. Technical University of Munich; 14 May 2018.

URL:
https://www.net.in.tum.de/fileadmin/bibtex/publications/theses/2018_trustworthy_process_tracking.pdf.
Accessed 15 May 2019.

22 Hyperledger Composer. Hyperledger Composer Documentation [online]. Linux Foundation; October 2018.
URL: https://hyperledger.github.io/composer/latest/introduction/introduction.html.
Accessed 1 January 2019.

23 Simon Stone. Hyperledger Composer Architecture [online]. IBM; 1 June 2017.
URL: https://www.slideshare.net/SimonStone8/hyperledger-composer-architecture.
Accessed 20 April 2019

24 Hubject. Get to know Hubject [online]. Hubject.
URL: https://www.hubject.com/en/about-us/.
Accessed on 19 May 2019.

25 Gireve. Our Expertise [online]. Gireve.
URL: https://www.gireve.com/en/our-expertise/.
Accessed on 19 May 2019.