Aristotle University of Thessaloniki
Faculty of Engineering
School of Electrical & Computer Engineering

In collaboration with
IBM Research Zürich & IBM CIC Benelux

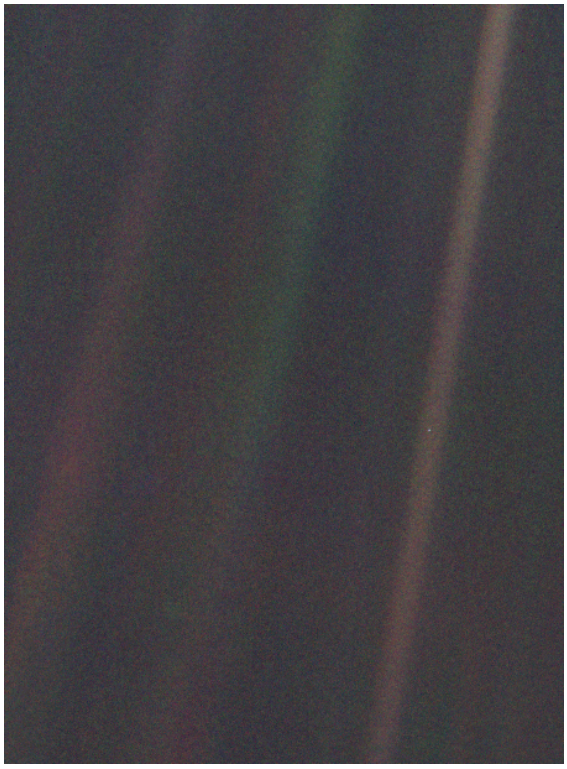
Diploma Thesis

Towards low-powered Industrial IoT devices transacting with consortium blockchains

Amoutzias Athanasios

Supervisor
Dimitrios Mitrakos
Associate Professor

Thessaloniki, March 2019



Look again at that dot. That's here. That's home. That's us. On it everyone you love, everyone you know, everyone you ever heard of, every human being who ever was, lived out their lives. The aggregate of our joy and suffering, thousands of confident religions, ideologies, and economic doctrines, every hunter and forager, every hero and coward, every creator and destroyer of civilization, every king and peasant, every young couple in love, every mother and father, hopeful child, inventor and explorer, every teacher of morals, every corrupt politician, every "superstar," every "supreme leader," every saint and sinner in the history of our species lived there-on a mote of dust suspended in a sunbeam.

Carl Sagan

Acknowledgements

Foremost, I would like to express my sincere gratitude to my thesis advisor Associate Professor Dimitrios Mitrakos. Firstly, for giving me the freedom and guidance to work on a fairly new and unexplored topic, and secondly for his openness and patience. Besides my advisor, I would also like to thank my mentor during my internship at IBM, Luigi Venti, for his support, guidance and ideas. My grateful thanks are also extended to Dr. Gero Dittmann and Dr. Jens Jelitto from IBM Research Zurich for our collaboration on performance analysis on their project. Additionally, I would also like to show my appreciation to Arne Rutjes for his ideas, connections, knowledge and genuine interest on the subject as well as to all my colleagues who contributed with their ideas, open discussions and suggestions. Furthermore, I wish to express my gratitude to the IBM CIC Benelux for giving me the opportunity to do research, collaborate and write this thesis as well as to Ieremias Athanasiadis for introducing me to the group.

Finally, I want to thank my friends and peers for all their crucial support during difficult moments and for inspiring me throughout my studies with their exquisite work and contributions.

Ultimately, I want to express my deepest gratitude to my family, my brother for his crucial help and especially my mother, who unquestionably supported and encouraged me throughout my life.

Abstract

With the invention of Bitcoin and the emergence of blockchain platforms, consortium blockchain networks came to existence. Hyperledger Fabric is a tool to provide businesses with permissioned blockchain networks and accelerate their workflows. Additionally, industry uses Internet of Things devices for automation, controlling, tracking and monitoring, but being in their early stage they introduce a notable attack surface [1] and they suffer from standardization. Research has shown that blockchains can be effective on the security of IoT devices as well as offer a common platform to create standards. This study aims to bridge IoT and permissioned blockchain setups with Hyperledger Fabric and provide a performance analysis of a tool[1] that enables low-powered devices to transact with a blockchain network. Building on existing work on the applicability of IoT and blockchains, it asks: How can a low-powered device transact with a Hyperledger Fabric network, what are the benefits, how does it perform, can it scale down further and open the doors to microcontrollers? In this context, consortium or permissioned blockchains are defined as networks that connect known entities which collaborate and/or have competing interests.

Our approach is based on a review of literature on permissioned blockchains, IoT security, use cases and finally a performance analysis of a tool that bridges IoT and Hyperledger Fabric. Indeed blockchains can strengthen the security and automation of IoT and can be seen as a provider of trust for transacting devices. Performance analysis demonstrated an improvement of 71% on bandwidth usage and 53% on CPU resources, thus establishing the first step to scale down further.

[1] developed by Dr. Gero Dittmann [2] and Dr. Jens Jelitto [3] from IBM Zurich

Contents

List of Figures

List of Tables

Glossary

B | C | D | E | H | I | M | P | S | T | U

B

B2B Business to Business. 1

B2B2C Business to Business to Customer. 1

BLE Bluetooth low energy. 35

C

CA Certificate Authority. 5

D

DDoS Distributed Denial of Service. 10, 16, 17

DLT Distributed Ledger Technology. 1, 12, 23

E

EOA Externally Owned Accounts. 10

EVM Ethereum Virtual Machine. 9, 10

H

HLF Hyperledger Fabric. iv, 1, 2, 22–25, 30, 35

HSM Hardware Security Module. 17, 20, 35, 48

I

IIoT Industrial Internet of Things. 16, 36

IoT Internet of Things. iv, 1, 2, 16, 35

M

M2M machine-to-machine. 16, 18

MCU Microcontroller Unit. 43, 48

MQTT Message Queuing Telemetry Transport. 18, 37, 38

MSP Membership Service Provider. 13, 23–25

P

P2P peer-to-peer. 6, 12, 13, 19, 36

PKI Public Key Infrastructure. 2, 4, 19, 36

PoW Proof of Work. 8, 9, 13, 14

S

SDK Standard Development Kit. 25, 30, 35

T

TEE Trusted Execution Environment. 17, 48

TXP Transaction Proposal. vii, 25, 26, 37

TXPR Transaction Proposal Response. vii, 25, 26, 37

U

UTXO Unspent Transaction Output. vii, 7, 8, 10

Chapter 1

Introduction

1.1 Motivation

In 2008, Satoshi Nakamoto published the Bitcoin whitepaper[10] where he described a decentralized network that enabled electronic currency with some very peculiar characteristics. The main achievement of this network, that came into existence in early 2009, is that it solves the double spending problem by making everyone's best economic interest to support an honest network. Blockchain as cryptographically linking blocks in an append-only data structure existed long before the introduction of Bitcoin [11]. Young Vitalik Buterin, inspired by the work of Satoshi Nakamoto and already contributing to the Bitcoin codebase published the whitepaper of Ethereum [12]. Ethereum is a Turing complete solution contradicting to Bitcoin. With the rise of Turing complete blockchain networks, consortium blockchains came to existence. We are investigating one of the Hyperledger frameworks, the Hyperledger Fabric [13]. Fundamentally it is a Distributed Ledger Technology (DLT), which means a shared ledger in a permissioned network with competing interest players. Consortium blockchains or DLTs are created to accelerate B2B and B2B2C interactions, at their base they are centralized-authoritarian, distributed databases, recorded in a blockchain fashion, with transaction based signed messages.

With the surge of Internet of Things (IoT) devices in consumer products and in the industry the attack surface has increased and security breaches have risen [14]. Thus, under some circumstances, distributed ledgers can be a suitable solution to mitigate specific attacks and accelerate the current workflow. We believe, it is always important to have the right tools for the job and make them easily accessible to everyone.

1.2 Research Topics

We aim to explore the following research topics in this thesis:

1. Consortium Blockchains and applications.
2. Integration of Internet of Things and Blockchains.

3. Identity of things and trusted operations.

1.3 Outline

We start at chapter 2 with an introduction to cryptographical hash functions, asymmetric cryptography and Public Key Infrastructure (PKI). We continue with Bitcoin and how transactions take place and consensus is reached and we expand on Ethereum.

In chapter 3 we discuss about the adoption, mutation of blockchains in the industry and how they differ from public networks. We continue with Internet of Things in industry and finally we conclude by merging both topics with a case study.

In chapter 4 we dive in Hyperledger Fabric, our tool of choice. We elaborate on transactions, smart contracts and key components of this network.

In chapter 5 we present related work done with IoT and blockchains, what other companies do, what academia researches, what IBM does and how we expand on that.

In chapter 6 we offer an analysis of a new tool while providing a brief background on software profiling.

In chapter 7 we conclude and discuss about future work.

Chapter 2

Blockchain Essentials

We cover the fundamentals needed to explain blockchain networks. At the first section, we start by covering fundamental cryptography concepts, next we cover the origin of blockchains; Bitcoin and how it works, then we continue with Ethereum and how it expanded and diverged from Bitcoin.

2.1 General Background

This section covers basic cryptographic concepts that run the web of trust today.

2.1.1 Cryptographic Hash Functions

A cryptographic hash function is a computationally efficient mathematical algorithm, that has certain properties which make it suitable for cryptography. It maps data of arbitrary size to a bit string of fixed size, known as hash. The slightest change to the input makes a large change in the resulting hash. Designed to be a one-way function, it is easy to produce the output given the input but computationally infeasible to produce the input given the output. In theoretical cryptography, the security level of a cryptographic hash function has been defined using the following properties. [1] [15]

1. **Preimage resistance**: It is computationally infeasible to find any input x which hashes to that output y. Given y, it is difficult to find any preimage x' such that $H(x') = y$ when given any y for which the corresponding input x is not known.
2. **Second preimage resistance**: It is computationally infeasible to find any second input which has the same output as any specified input, that is given x to find a 2nd-preimage $x' \neq x$ such that $H(x) = H(x')$.
3. **Collision Resistance**: It is computationally infeasible to find any two distinct inputs x, x' which hash to the same output such that $H(x) = H(x')$

[1] H is a hash function with inputs x, x' and outputs y, y'

2.1.2 Public Key Cryptography

Public Key Infrastructure (PKI) also referred to as asymmetric cryptography, is a system that uses a pair of keys, a *public key e* that is widely known and a corresponding *private key d* which is known only to the owner. The two keys have a mathematical relationship and in secure systems the task of computing d given e is computationally infeasible. In such a system, any person can encrypt a message using the receiver's *public key* and that encrypted message can only be decrypted by the receiver's *private key*. The public key defines an *encryption transformation* E_e, while the private key defines the associated *decryption transformation* D_d. Any entity B wishing to send a message m to A obtains A's public key e, uses the encryption transformation to obtain the *ciphertext* $c = E_e(m)$ and transmits c to A. To decrypt c, A applies the decryption transformation to obtain the original message $m = D_d(c)$

Public key cryptography establishes a secure communication by satisfying the following cryptographic objectives:

1. **Confidentiality**: Keeps the content of information from all but those authorized to have it. Achieved by encrypting, B uses A's public key to encrypt the message and only B can decrypt that message with its private key, but anyone could impersonate A.

2. **Data Integrity**: Addresses the unauthorized alteration of data using digital signatures. Digital signatures is a scheme where both parties verify that the information sent was created by the claimed sender and has not been tampered with. Entity A uses its private key to produce a hash of the message and attaches it as a signature to the message. Entity B receives the signed message and uses the same hash algorithm that A used with its public key. Finally it compares the resulting hash with the message's actual hash, if the two are equal then it knows the author is the one with the possession of the particular private key.

3. **Authentication**: Associated with identification and applied to both entities and the data itself, which implicitly provides data integrity. Achieved by encrypting ones message with its own private key, B uses its private key to encrypt the message and send it to A, it can decrypt the message using B's public key and thus verifying the sender, but anyone intercepting the communication can decrypt it.

4. **Non-repudiation** Prevents an entity from denying actions by using digital signatures, a procedure that involves a trusted third party to resolve the dispute. i.e A authorizes a transaction to sent an asset to B and later denies it. Since the transaction is signed with A's private key, there is no room questioning.

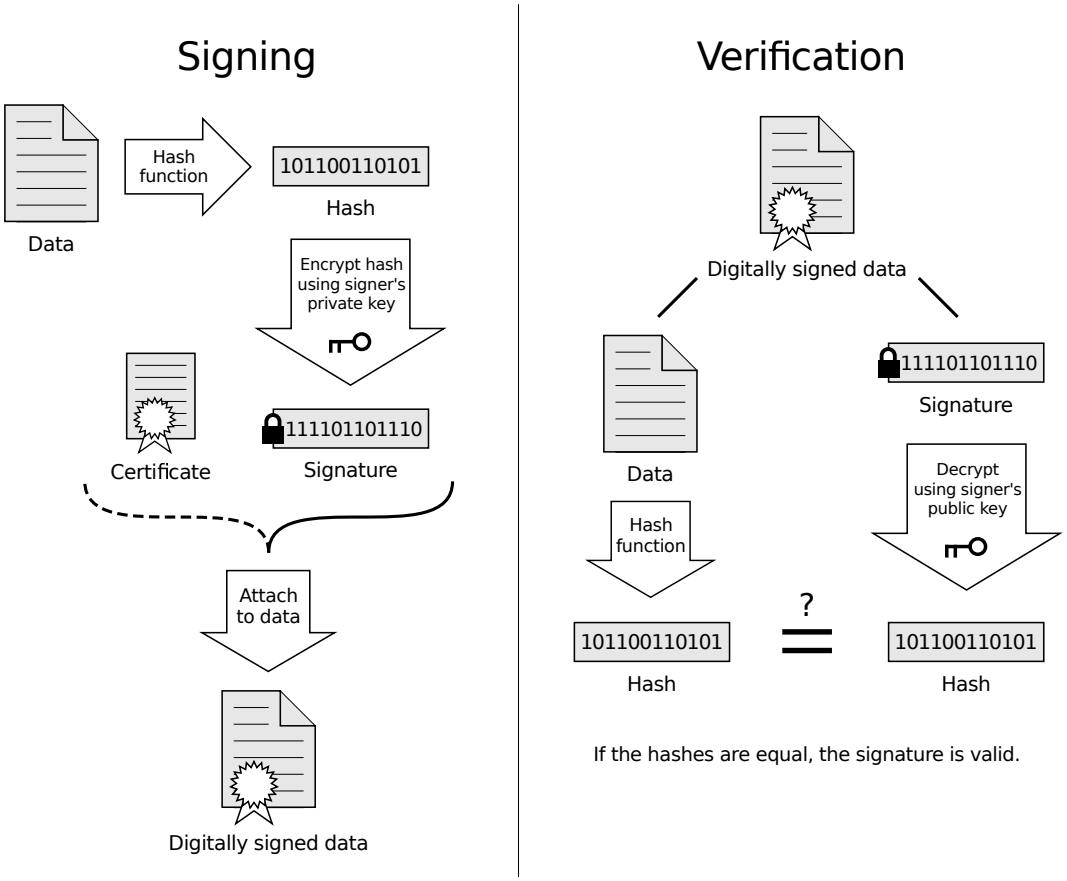


Figure 2.1: How digital signing works [4].

2.1.3 Public Key Infrastructure

In this subsection we cover the basics of Public Key infrastructure in order to associate them later to Fabric's components. PKI comprises [16] :

- **Certificate Authority (CA)**: A trusted node that acts as a maintainer, keeping the public keys for all nodes. The CA also bootstraps newly joined nodes in the network by adding their public key to their list, thus only the new node and the CA need to be configured.
- **Certificates**: A signed document, as a passport, issued by the CA stating that a particular name holds a particular private key. Other parties, knowing the public key of CA, can authenticate a user.
- A repository for retrieving certificates.
- A method of revoking certificates.
- A method of evaluating a chain of certificates from known public keys to the target name.

2.2 Bitcoin

2.2.1 Overview

Bitcoin is a payment network protocol that enables P2P payments in a sense that no central authority issues new money or tracks transactions. A record of all transactions is held in a shared ledger, where each transaction is verifiable, transparent and can be tracked through all its history. Batches of transactions with a timestamp are called blocks and each block is cryptographically connected to its previous block by including the hash of the previous block. In that manner, a chronological data structure is created, containing verifiable transactions, or else money, that can be spent by the owner's keys.

Bitcoin's blockchain is secured through repeated computations or else sealed with energy expenditure. The end product is a network that does not need any intermediaries, transactions are made in a P2P fashion and the blockchain through all this energy expenditure becomes secure and immutable.

Lets assume that Alice wants to send ₿7 (bitcoins) to Bob. Alice has already ₿10, she creates a transaction with one input and 2 outputs, one for Bob at ₿7 and one to her address for the change at ₿2.995 and the rest ₿0.005 are fees to the miners (today's fees are ₿0.00005000)[17] to get her transaction into a block. The output for Bob is signed with his public key and the output to her address as change is signed with her private key. After the creation of the transaction, it propagates through nodes in the network, with a gossip protocol, where each node upon receiving it validates it and gossips it to other connected nodes. Finally, it ends up to a node which will include it in the next block that it will successfully mine [18].

Every node is capable of putting transactions they choose in a block and mine it, but nowadays mining is done collectively and the mining pool chooses the transactions for the next block from the transaction pool [19]. Now that there is a block candidate with ordered transactions in it, it is repeatedly hashed by changing an additional value called the nonce. After many changes, a hash is found that is smaller than the difficulty target. A legit block nowadays needs to start with 18 zeros.

Difficulty adjusts every 2016 blocks and it targets the whole network to produce 2016 blocks every two weeks [20]. An interesting fact is that intervals between finding the next block follow a probability distribution that is memoryless. Block times follow the exponential distribution and the probability of a block **not** being found in x minutes or longer is:

$$p(x) = e^{-(x/10)} \tag{2.1}$$

So that means that finding a block in 10 minutes or less gives as a probability of 0.63, calculating for $x = 10$ in 2.1. The number of blocks that are expected to be found in a certain time (e.g 2016 per two weeks) follow a Poisson process. The probability of finding k blocks in T minutes is:

$$P(k, T) = (\frac{T}{10})^k \times \frac{e^{-}(T/10)}{k!} \tag{2.2}$$

So the probability of finding exactly six blocks in an hour would be 0.16 [21].

2.2.2 Transactions and UTXO

In this subsection, we elaborate on transactions and the Unspent Transaction Output (UTXO) set.

Transactions are the most important part of every blockchain network, they are the fuel that change the state of the blockchain. At a high level, a transaction is the authorization of a value from the owner to the recipient. Then, the new owner becomes capable of spending this new value by creating a new transaction and transferring ownership of the value. The most common transaction form for bitcoin is a single payment as seen in a simplified[2] manner in 2.2. At a lower level, one input contains the digital signature and public key[3], the output to the recipient is signed with recipient's public key.

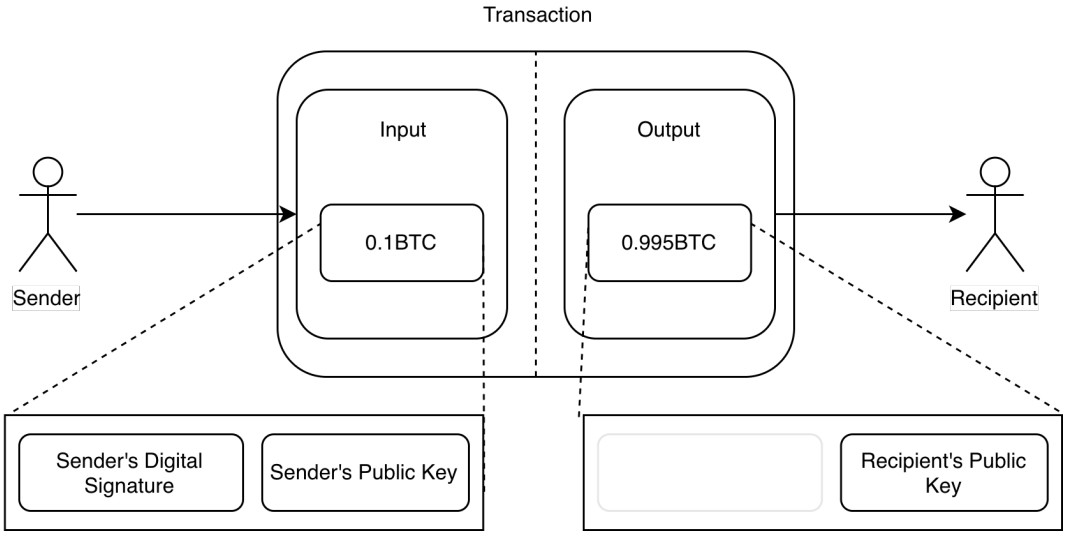


Figure 2.2: A simplified single transaction with one input and one output.

The fundamental building block of Bitcoin transactions is the UTXO set. This set, consists of UTXOs that are being used as an input to the next transaction. Each UTXO could be described as an instance of a coin, having any arbitrary value, indivisible, that is going to be spent altogether, destroyed and another new UTXO will take its place. Each UTXO that is consumed as an input to a transaction will create a new one. The relationship of inputs(M) and outputs(N) is $M - N$ where $M, N \geq 1$. Wallets track within the UTXO set, which UTXOs the user's private keys can control, and represents it as a balance. A transaction references previous outputs as new inputs and assigns all inputs to new outputs as seen in 2.3[4].

To conclude with UTXOs, we should emphasize their crucial characteristics. Each UTXO is discrete, indivisible, of arbitrary value, denominated in satoshis[5] The only way to be spent is to be consumed in their entirety by a transaction.

[2]The difference in the input and output value is the transaction fee.
[3]The sender's public key is used to verify the digital signature.
[4]Fees have been excluded for simplicity.
[5]Satoshi is the smallest unit of bitcoin, ₿10^{-9}.

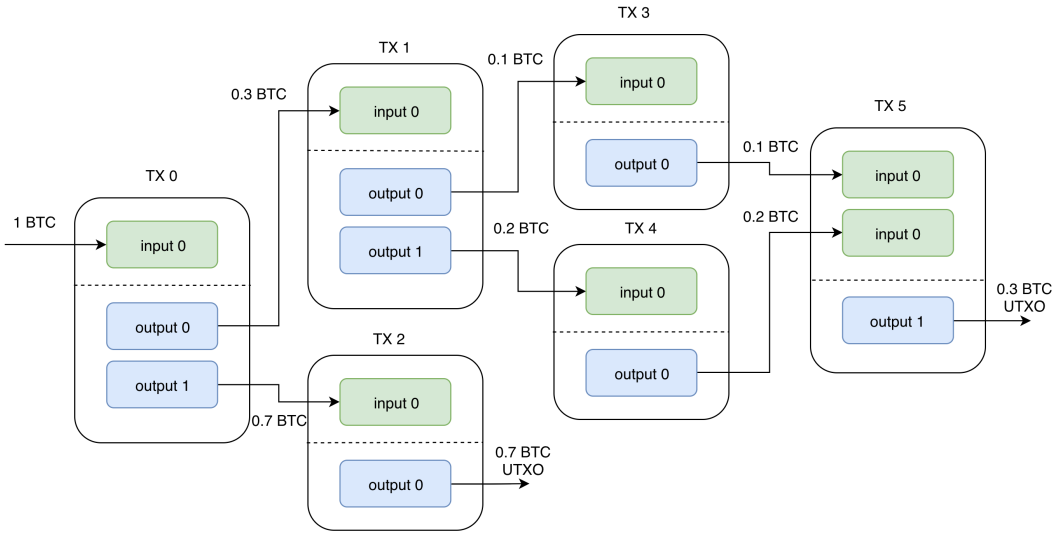


Figure 2.3: UTXOs become input to the next transaction.

2.2.3 The Blockchain

In this subsection, we elaborate on the abstract idea of blockchain.

An ever-growing distributed list of records, append only, replicated and shared among the participants of the network (aka distributed ledger). Each block in the chain contains a list of transactions and a hash of the previous block. Blockchain is a log whose records are batched into time-stamped blocks where each block is identified by its hash. By including the hash of the previous block[6], it establishes a link between the blocks, thus creating a block-chain.

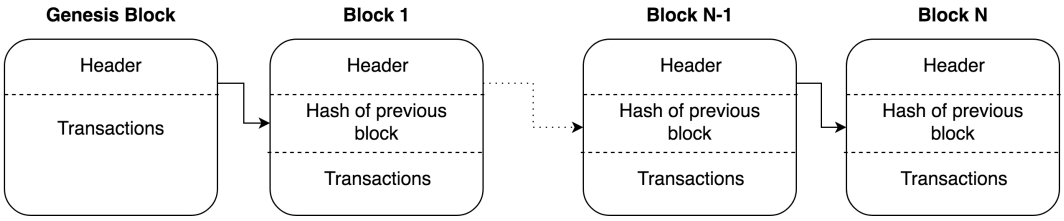


Figure 2.4: Blockchain simplified

A blockchain can be defined as an immutable shared ledger that holds transactions, maintained within a decentralized and distributed network of mutually non-trusting peers. Every peer maintains their copy of the ledger and is responsible of validating each transaction on it.

2.2.4 Mining and immutability

In this subsection, we discuss how PoW works and makes the blockchain immutable. We should state that mining, PoW and consensus are highly interrelated.

[6]The Genesis block, is the first block on the chain and it does not include a reference to a previous block.

Abstractly, every node, receives a transaction, verifies it and propagates it to the rest of the network. This transaction, will eventually be included to a candidate block and finally this block will be part of the blockchain. The bitcoin's system of trust is based on computations through mining which secures the system and enables the emergence of network-wide consensus without central authority.

Bitcoin uses a consensus algorithm that involves mining and PoW. With PoW, the goal is to find a hash of a block that is lower than the difficulty target; the procedure of hashing the block repeatedly to find a suitable hash is called *mining*. As stated before, mining is done collectively and the hashrate across the entire bitcoin network is 46,000 Peta[7] hashes per second [22]. The network repeatedly hashes the header of the block, using the SHA256 algorithm. The output of each hash is random and finding a hash below the target is probabilistic. It is similar to rolling a dice that will be under 6, easy, it becomes harder as the goal number becomes lower, e.g numbers only under 2.

With the passing of time, more blocks are added on the blockchain and it becomes infeasible to change its history. If someone wanted to change a block in the past, they would have to recompute the PoW of each next block in order to get matching hashes.

Finally, the blockchain that is accepted is the longest, heaviest, most expensive or the chain with the most work on it[8]. To conclude, recomputing a new chain is considered impossible due to the energy expenditure and mining hardware[9] that is needed. The deeper in history the alteration the more the computations needed to make it the new acceptable version.

2.3 Ethereum

In this section, we elaborate on how Ethereum expanded on Bitcoin by offering a platform for developers to build their decentralized solutions. We use the term "decentralized" to describe the distribution of the authority and power on a system.

In this section, we offer a brief summary of Ethereum, how it expanded on Bitcoin's idea and its purpose.

2.3.1 EVM and Turing Completeness

Ethereum has a virtual machine for the execution of smart contracts and transactions, called Ethereum Virtual Machine (EVM). It can be perceived as a global single threaded CPU, where every smart contract computation and transaction is executed on each node in the network. Every node in the network uses the EVM to ensure redundantly correct execution and consensus to agree on the answer. The EVM, has

[7] 46000×10^{15} hashes per second

[8] Work is defined as the sum of the difficulties.

[9] Bitcoin is mined with ASICs, the most efficient solution.

the capacity to be a quasi-Turing-complete state machine; "quasi" because all execution is regulated through gas expenditure, which dictates how much computing power can be spent on a transaction. Hence, the halting problem[10] is solved. The behavior of the EVM depends on the programming language used for the smart contract. Ethereum has its own programming languages, Solidity which offers Turing completeness and Vyper which forbids certain operations to enhance security.

2.3.2 Smart Contracts and Gas

Smart contracts are programs that live on the blockchain and are immutable, after deployment there is no turning back, no modifying or deleting[11]. The smart contracts provide an API callable by anyone and the costs for each call are dictated by the smart contract computations. These costs enacted to:

- mitigate DDoS attacks on the network and
- make smart contracts efficient on computations and storage, thus incentivizing developers to optimize their solutions.

The cost is enforced by gas, and gas is bought with wei, the minimum unit of Ethereum $1wei = 10^{-9}ETH$. Its price is driven by the demand of the network and by the market [23]. Gas and the price of ETH are decoupled, since units of gas align with computation units having a natural cost, while the price of ETH fluctuates as a result of market forces. Each operation offered by Ethereum's programming language has a specific gas price as seen here [24]. Fundamentally, gas is the execution fee that senders of transactions need to pay for every operation made on Ethereum.

2.3.3 Accounts and Transactions

Ethereum has two types of accounts, contract and Externally Owned Accounts (EOA). Contracts have addresses, derived from the creator's address, but they do not have a private key - they are controlled by the logic of their smart contract code. On the other hand, EOA do have a corresponding private key and as a result access to funds and contracts. Every action on the Ethereum blockchain, like Bitcoin, is carried out by transactions. Transactions are the inputs to the EVM to evaluate contracts, update balances and fundamentally update the state of the Ethereum.

Unlike Bitcoin, each transaction is an independent entity and accounts hold a balance, instead of tracking UTXOs. Substantially, a global state, as seen in figure 2.5, stores a list of accounts with balances, code, internal storage and the nonce[12]. A transaction is considered valid if the account initiating the transaction has enough balance to pay for it. Finally, if the receiving account is a smart contract, it causes

[10] The halting problem refers, given input, whether a program will finish running or continue to run forever.

[11] Deleting is available but has to be programmed before deployment, transaction history will remain intact after deleting

[12] The nonce is a sequence number used to prevent message replay

its code to run and as a result internal storage might be changed and more messages might be initiated.

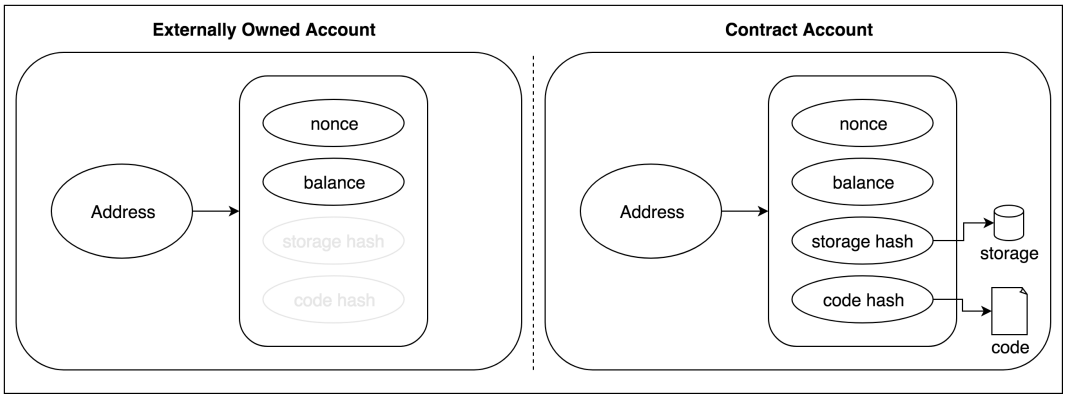


Figure 2.5: Ethereum World State [5].

Chapter 3

DLT and IoT in Business

This chapter aims to provide a brief background on Distributed Ledger Technology and Internet of Things in industry and confidently present a few use cases to bridge both worlds.Distributed Ledger Technology (DLT), private, permissioned or consortium blockchains are used interchangeably in this chapter.

3.1 Distributed Ledger Technologies

As discussed in chapter 2, a blockchain is essentially a trustless, P2P, append-only list of records verified and shared among participants in the network. With the rise of Ethereum, blockchains gained Turing completeness and an easier way to implement business logic and automate procedures, called smart contracts. As seen from Bitcoin and Ethereum, those kind of networks tend to be slow and too expensive to maintain due to their public nature. Private and consortium blockchains could benefit from stripping down some mechanisms and thus a new design emerged for business and enterprise environments where all participants are known, cutting down unnecessary overheads and architectural choices used in permissionless or public blockchains.

As Vitalik Buterin defined consortium blockchains in [25]:

> *A consortium blockchain is a blockchain where the consensus process is controlled by a pre-selected set of nodes; for example, one might imagine a consortium of 15 financial institutions, each of which operates a node and of which 10 must sign every block in order for the block to be valid. The right to read the blockchain may be public, or restricted to the participants, and there are also hybrid routes such as the root hashes of the blocks being public together with an API that allows members of the public to make a limited number of queries and get back cryptographic proofs of some parts of the blockchain state. These blockchains may be considered "partially decentralized".*

In the upcoming subsections we discuss how consortium blockchains differ from permissionless on each of their attribute and why they can still work. The main

criteria for designing such networks are incentive compatibility, responsiveness and finality of transactions. We follow the structure of comparison of this work [26].

We should state that blockchains, especially in Europe, need to be designed in such a way that respect users' data privacy under the European General Data Protection Regulation. Consortium blockchains need to be compliant with such regulations [27].

3.1.1 Immutability

Immutability in permissionless blockchains comes from energy expenditure of Proof of Work and from the protocol that each node builds and propagates the longest chain.

In consortium blockchains, since validator nodes are under known parties it could be infeasible for the majority to agree and change the history of the chain. Could be done but only if everyone agrees. Also a hash could be periodically published on permissionless chain in order to know what the current state of the consortium chain was. Recomputing the hashes would be trivial without Proof of Work (PoW). However if members can come to an agreement, they could alter the history of the blockchain, since such actions are considered controversial and go against the immutability aspects of blockchains, a major selling point, a periodic public hash would suffice to let everyone know that history has been modified. As we mentioned earlier there might be a good reason to modify a chain, and that is due to regulations. Since members are known they can be fined by governments and even shut them down if they do not comply. In a permissionless network that would be impossible.

3.1.2 Access and Privacy

A blockchain is based on a P2P network that propagates transactions and new blocks, fundamentally linking its members. Each new node on the network is connected to a neighbor and start receiving blocks, in the meantime it validates each block and cross references it with other neighbors. In figure 3.1 we see the distribution of bitcoin nodes in Europe. Each user uses their private key to derive their address, which is their public identity and uses a node to transact with a network. Users might run their own full nodes or use third party services and light nodes.

In the consortium setup each user is identified by its node and its certificate that is registered with a Membership Service Provider (MSP). More details on how that works on 4 where we dive in our consortium blockchain of choice, Fabric. Participant nodes build the ledger from the genesis block. When a new member joins the network, depending on the use case, previous blocks could be gained through a similar P2P approach. We should state that this restricted setup ensures data privacy by default. Encrypting transactions is not easily doable since they are needed to verify the previous blocks. There is a solution with zero knowledge proofs that allows for the authentication of transactions without giving away any personal information of either the sender or receiver[28].

GLOBAL BITCOIN NODES DISTRIBUTION

Reachable nodes as of Sat Feb 09 2019 14:14:56 GMT+0100 (CET).

10418 NODES

24-hour charts »

Top 10 countries with their respective number of reachable nodes are as follow.

RANK	COUNTRY	NODES
1	United States	2555 (24.52%)
2	Germany	1987 (19.07%)
3	France	707 (6.79%)
4	Netherlands	538 (5.16%)
5	China	409 (3.93%)
6	Canada	382 (3.67%)
7	United Kingdom	362 (3.47%)
8	Singapore	323 (3.10%)
9	Russian Federation	292 (2.80%)
10	Japan	240 (2.30%)

More (102) »

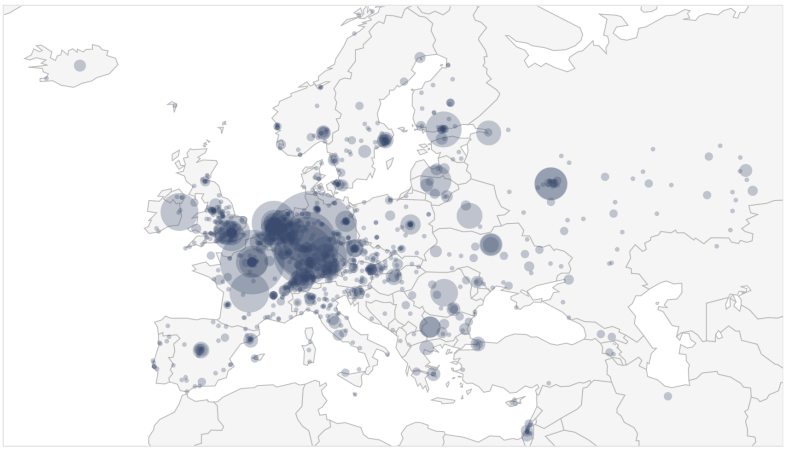


Figure 3.1: Bitcoin nodes distribution in Europe [6]

3.1.3 Mining, consensus and incentive compatibility

On of the most important aspects of a blockchain network is how it reaches finality and how it incentivizes its participants to play along. Permissionless and Permissioned networks are based on a whole two different approaches. One is based on cryptocurrency economics[29], where the currency is a crucial attribute of the network whereas the other is based on reputation and lawsuits.

As we have previously stated, miners are nodes that are sharing, or collaborating with a mining pool, their computational power in order to solve the PoW. Their reward on Bitcoin and Ethereum are some BTC or ETH per block, specifically as of February 2019, 12.5BTC and 3ETH per block. Each transaction has a fee paid to the miners for their expenditures on hardware and electricity and also to incentivize them to include their transactions in the next block. This protocol and game, secures the network against 51% attacks. That means that an adversary can rewrite the history only if it holds the 51% of the network's hashing power, but that on most occasions is counter-incentivized, the gains would be lower that what he spends. Practically as low as 33% for a double spend attack [30].

On the contrary, in consortium blockchains validating nodes can be trusted in some degree and everything about their identity is known. In these setups, consensus can be simplified and there is no need for such secure and energy wasteful mechanisms as PoW. Security and trust comes from the permissioned network which is, by definition, Sybil proof, something that public blockchains need to mitigate[31] [32].

3.1.4 Forks, responsiveness and finality of transactions

In permissionless chains, blocks are added to the chain with an interval, commonly ranging between some seconds for Ethereum and a few minutes for Bitcoin (on an average, 15 seconds and 10 minutes respectively). Finality of a transaction comes after some time where the probability of a number of blocks to appear and make the current chain the longest is negligible. For bitcoin we wait for six confirmations,

or six blocks (1 hour) and only then we consider a transaction as finalized. This is all done due to forks, where the longest chain is accepted on the network. That means if someone could compute blocks given a point in the past and continue their version of the chain, if they succeed in making a heavier chain then this will be the new chain.

In figure 3.2, we have an orphan block[1](purple), the main chain (blue) that everyone uses and a secret, unpublished chain (red). The secret chain is longer than the main chain and has the potential, when published, to be accepted by everyone, thus having the capacity to revert old transactions[2].

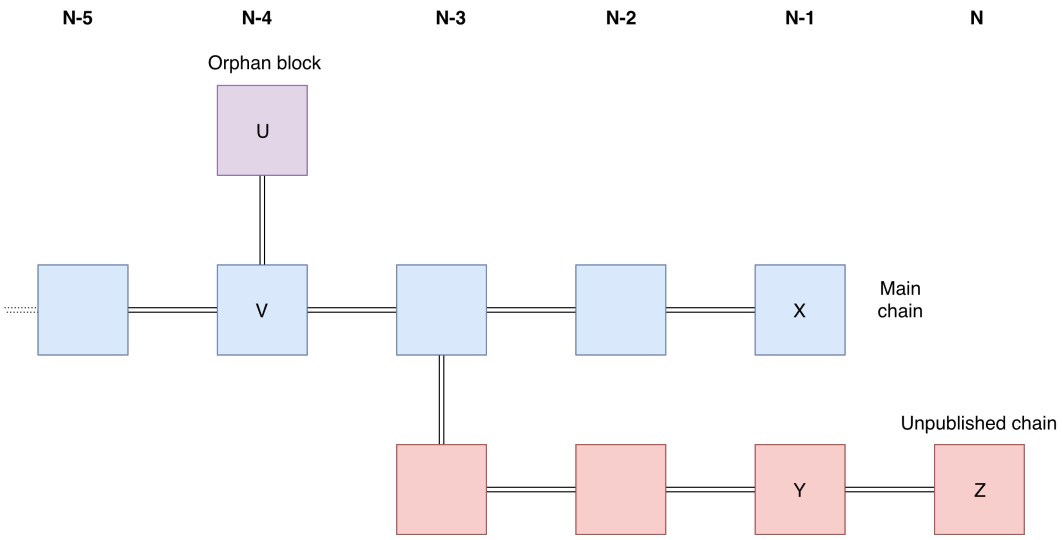


Figure 3.2: A new longest chain (red) reverts transactions on main chain (blue).

On the other hand, in consortium blockchains, we do not have a risk of a new set of blocks appearing and making the old chain obsolete, every transaction upon ordering is considered as final. In addition, we do not need a block interval as well as to wait for confirmations, or new blocks on top of the associated transaction, thus making the network a lot more responsive. Blocks are created on demand and everyone works on the main chain. We could say that when participants are known, everything is more predictable and linear.

3.1.5 Smart Contracts

Bitcoin offers very strict functionality for security, while Ethereum offers Turing completeness, it is rarely used due to undecidability [33]. Contracts in order to pass security audits needs to be deterministic and every state easily analyzed.

In consortium blockchains we can have more complex functionality that does cost orders of magnitude less to operate and it is easily scalable, since smart contracts do not need to be executed on every single node but only on nodes that have them

[1] An orphan block, in the context of Bitcoin, is a valid block that was computed but not accepted by the network because another valid block referencing the previous arrived first.

[2] Letters within the blocks denote a different set of transactions.

installed and are instructed to run them. In this way we achieve greater level of parallelazation and scalability.

To conclude this section, due to the known participants on a consortium network it is possible to strip some expensive mechanisms that offer security and stability in permissionless networks, simply because they are not needed. By adapting the permissioned networks, we achieve higher throughput, greater responsiveness and flexibility.

3.2 Internet of Things

In this section, we provide an introduction to Internet of Things (IoT) and their position in industry and business as well as examples of leveraging said devices and blockchain. We further discuss weak security points and how blockchain can strengthen them.

When we refer to IoT, we address small-scale devices that are exchanging data over the Internet or directly to one another M2M. Their purpose is to offer insights with their measurements or automate procedures with an intelligent factor based on history and current data. They have the ability to measure, communicate and act all over the world. Often the functionality of said devices is limited, as well as their capabilities, they are dedicated to one function and their purpose is to deliver it to a central database or act based on user instructions. There are many applications in the wild, some notable ones are warehouse inventorying, healthcare, transport, food safety, monitoring and operating a smart home [34]. Industry have them deployed as well, sharing many common characteristics and goals. They are being referred as Industrial Internet of Things (IIoT) and they cover many uses, industries and applications. Initially, focusing on the optimization of operational efficiency, intelligent manufacturing and smart industry. Particularly, connected machines and devices in industries such as oil and gas, transportation, power generation, and healthcare, where there is more at stake; an unplanned downtime, security breach, a system failure can result in life-threatening, financial damaging or high-risk situations. It is expected, within the next year, for the connected devices to reach 20 billion [35], mainly because of our need to control and have a digital representation of the world.

In the rest of the subsections, we cover the basics of IoT, applications and issues that have arisen in the last years with the spike of unsecure interconnected things.

3.2.1 Security

IoT are usually low-powered devices that lack the hardware and computational power to support state-of-the-art security models, making them vulnerable to cyber attacks [36]. One way to mitigate those attacks is through standardization of protocols and design in order for everyone to work on making them more secure by developing specialized hardware and software solutions. Blockchain can play a vital role in this standardization and race of securing the IoT. In late 2016, a DDoS attack was launched with the Mirai botnet, causing a huge disruption in internet services that

affected major companies. The Mirai botnet exploits the weaknesses of IoT devices, allowing them to be hijacked and used to flood a target with requests[37].

Authors of [38] suggest a framework for IoT, built on Ethereum to mitigate DDoS attacks. They propose a smart contract on Ethereum where it acts as an intermediary for all the communication done with traditional servers. Every time an IoT wants to send a message, it gets authenticated by the smart contract and eventually sends its message, both operations cost gas on Ethereum. If a device would act maliciously and send multiple requests, it would deplete its available gas and effectively mitigate the DDoS attack.

We continue by providing the building blocks of IoT Security, as stated by [1].

Device Identity

Device Identity holds the foundations of security. The device identifier must be immutable, unique and attestable. Certificates must be issued for devices and all important operations should take place in Trusted Execution Environment (TEE) and the keys should reside in a Hardware Security Module (HSM).

Provisioning and Management of Devices

Assigning certificates to devices makes possible to revoke access and also provide automated ongoing compliance with current security standards. Furthermore secure authentication, authorization and accountability minimizes the potential to compromise a device during the on-boarding process, where the devices get their access to the network. Certificates enable this safe bootstrapping to the network.

Confidentiality

It is mandatory to ensure sensitive information remains private and inaccessible to unauthorized parties. A device should encrypt locally stored sensitive information, the device should protect they keys as stated with HSM and not be available across the bus. Finally end-to-end encryption should be used for all sensitive communications.

Integrity

The data created or received by a device must be trustworthy and tamper proof. Non-repudiation methods should be established to ensure the integrity, origin and delivery of data.

Availability

IoT devices should be designed to function in a predictable and expected manner if there is a loss of connectivity with the Internet. In addition, devices should limit

requests from unknown or unauthorized entities to further limit attack surfaces and establish an ”available to authorized users only” policy.

Life-cycle Management

As the industry evolves, more vulnerabilities will be discovered and fixed. It is crucial to establish a design where software updates are automated and not rely on any consumer action. To protect end-users and third parties, there should be a way to handle end of life of devices, by limiting device functionality where it is considered unsecure. That might be a result of outdated hardware to keep up with software updates. Disclosure of such actions should be clear and reach the users.

3.2.2 Communication Protocols

We find important to present some sets of communication means and protocols that are suitable for IoT, with respect to our use case.

MQTT

Message Queuing Telemetry Transport (MQTT) is a machine-to-machine (M2M) publish-subscribe based messaging protocol on top of TCP/IP, designed for bandwidth and power efficiency. Publish-subscribe scheme is event-driven and enables messages to be pushed to clients through an intermediary, a central communication point, an MQTT broker. Each client that publishes a message to the broker, includes a topic where the topic acts as the routing information for the broker. Each client that wants to receive messages subscribes to a certain topic and the broker delivers all messages from this topic to the client. Therefore, the clients do not have to know each other, they only communicate over the broker.

3.3 Case study

In this section, we are going to analyze a case study, but first we are discussing the integration of blockchain and IoT.

3.3.1 Integration of Blockchain and IoT

Blockchain can enhance the IoT by providing a trusted sharing platform where information can be reliable and traceable. Blockchains, due to their strict architecture and standards, can offer a well-needed standardization for IoT trusted communications. In the case that data needs to be shared among participants with conflicting interests that need to trust each other, blockchain can be a perfect fit. One case would be food safety, where food is tracked from the manufacturer to consumer.

Authors of [39] offer some benefits of Blockchain and IoT integration, which we summarize and add to them below.

Decentralization and scalability

As we stated before, current IoT ecosystems rely on centralized models where all devices are identified, authorized and connected through cloud servers owned by leading companies. A decentralized blockchain approach would adopt a standardized P2P communication model to support the billion transactions of IoT devices. It will help scenarios where a conglomerate controls the processing and storage of the information of a huge part of the population. Blockchains are fault and crash tolerant, and IoT could benefit from such characteristics.

Identity

Using a single platform to connect all devices will enable participants to identify every single device. All data that is being transacted in the blockchain could be uniquely identified with the device that provided them thus correlating and verifying origin of the data. Furthermore, blockchain provides a way to distributed authentication and authorization of devices for IoT applications. For example, giving rights to a person to unlock a smart lock. Work is being done on self-sovereign identity for humans and autonomous devices that aim to give decentralized identities with minimal disclosure of information on the blockchain [40] [41].

Autonomy

Smart contracts, combined with IoT, can enable smart autonomous assets and hardware as a service where someone could rent but not own the hardware and pay through the blockchain. Additionally devices are capable of interacting with each other without the need of intermediaries.

Reliability

Device readings and information are attestable and immutable, living in the blockchain forever. Data is easily verifiable, traceable and thus accountable.

Security

Communications can be secured on the blockchain layer and each transaction or message exchanged between devices can be verified, an important application for vehicle communication is presented here [42]. In addition, PKI can be decentralized on the blockchain as described here [43].

3.3.2 Cold Chain Monitoring

Cold chains are used to guarantee a specific range of temperatures during freights. Applications range from pharmaceutical to food safety, chemicals to human blood and organs, primarily anything temperature-sensitive and intolerant. Fundamentally, they are temperature controlled supply chains and can also be a part of a supply chain.

A device with a temperature sensor, such as a Raspberry Pi, with an approved temperature probe, can measure a container's temperature and submit periodical transactions to a consortium blockchain. The blockchain network consists of entities such as pharmaceutical manufacturers, transportation companies, hospitals and pharmacies. Every participant needs to be able to verify that the temperature of goods was within the acceptable range as well as to automatically transfer ownership of goods based on the handler and location. One important benefit of blockchains is that information and quality control are easily auditable from the interested party, since data is linked and immutable. Hence, the acceleration of the quality control is achieved.

Transactions can include a timestamp, location, monitoring goods, ownership and temperature. A device has its own identity, stored in a HSM, and uses it to sign the transactions it is broadcasting, thus each transaction is tamper-proof and uniquely identified by a device.

The problems we address here are:

1. How can we accelerate a supply chain?
2. How do we trust that data came from a specific device?
3. How do we ensure data integrity?
4. How do we achieve non-repudiation for ownership of good during freight?

Below, we proceed to elaborate on how with our approach on blockchain and IoT can solve these problems.

Acceleration of supply chain

Acceleration comes from the network that all parties have joined. They can monitor the state of the goods and automatically change ownership from each handler with the use of smart contracts. For example, when a container leaves the manufacturer and is now handled by the transportation company, upon receiving the container an RFID is scanned, and it triggers a smart contract on the blockchain, where ownership is changed.

Trusting the origin of data

Trusting a device is associated with device's identity. Every member of the network needs a certificate, such certificates are assigned from a certificate authority and they get an associated entry on a membership service provider, which as explained

in 4.1.2, maps certificates to identities on the network. Hence, each device has gone through an auditing process, and is now authorized and uniquely identified to access and transact with the network.

Data integrity

Since every device has its own key, it is able to sign the transactions and send them to the network for ordering. Transactions are accepted when various parameters are satisfied, e.g transaction must be signed by a white-listed entity. More details on how transactions work and propagate through the network could be found on 4.

Non repudiation of ownership

Non-repudiation of ownership means that a party upon receiving the goods, or any sort of information, cannot deny it. Every transaction is an action between parties, that might be a change of ownership or an update on the state, such as a temperature reading. Blockchains are immutable and information on the ledger is available to every participant, thus no one can deny that something happened as long as it lays on the chain.

Chapter 4

Hyperledger Fabric

Hyperledger Fabric [13] is a framework to create permissioned distributed ledger networks, open source and hosted under the Linux Foundation, with major contributions from IBM, Digital Asset and Secure key, among others. It is developed with a modular design where each component is pluggable in order to create an adaptable solution suitable for each use case. A thriving environment for Fabric is enterprise business applications like supply-chains [44], financial transactions [45], asset management [46], food safety [47] and digital identity [48].

4.1 High-Level Architecture

In this section, we provide a thorough explanation of the components of Fabric and we mainly focus on how transactions are created and propagated through the network.

4.1.1 Background

Major blockchain networks are established with an *order-execute* pattern, i.e network participants use the consensus mechanism to order transactions and only once the ordering is finalized (into blocks), all transactions are executed sequentially. Thus, essentially implementing active state machine replication and enabling fault-tolerant services in distributed systems [49].

Hyperledger Fabric (HLF) follows another approach and establishes an *execute-order-validate* flow of transactions. After an initiation of a transaction, it gets executed against the chaincode on a peer level. After execution, transactions are grouped into blocks in a sequential manner by the orderers. Finally, the blocks end up to peers where they validate the state changes from each transaction against the endorsement policy. Validation is deterministic since all peers validate transactions in the same order.

Quoting the authors of [13]

> *"In this sense, Fabric introduces a novel hybrid replication paradigm in the Byzantine model, which combines passive replication (the pre-consensus computation of state updates) and active replication (the post-consensus validation of execution results and state changes)."*

In the following subsections we analyze Fabric's components.

4.1.2 Terminology

Ledger

The ledger component at each peer maintains the ledger on persistent storage, and enables simulation, validation, and ledger-update phases. Broadly, it consists of the blockchain, where all history is recorded in an immutable fashion and the world state where it represents the current value of the set of key-value pairs.

Chaincode

A chaincode, or else smart contract, implements the business logic with general programming languages (Go, Java, NodeJS) and gets invoked during the execution phase. It is a core element of a Turing complete DLT network such as HLF. Chaincode is executed locally on peers and it is available for everyone to review. Contradictory to Ethereum it is not a part of the blockchain.

Endorsement Policy

Rules that specify the correct set of peers that are responsible for the execution and approval, endorsement, of a given chaincode. These peers are called endorsing peers or endorsers and govern the validity of the chaincode execution results by singing those results. Endorsement policies are defined with logical expressions such as:

$$Org1 \wedge (Org2 \vee Org3).$$

Organizations

Organizations (Org) are members that are invited to join the network by the blockchain service provider. They are given access by having their MSP to join the network. Physically, an organization scales from a multinational corporation to a an individual. Organizations use the peers as a gateway to transact with the network. We should note that while all of the members can be characterized as organizations, not every organization will be a part of a consortium.

Fabric Node Types

1. **Clients** - network nodes running the application code which coordinates transaction execution
2. **Peers** - maintain a record of transactions within append-only data structure (ledger), responsible for execution of chaincode and its lifecycle. In order to allow load balancing and scaling not all peers are responsible for execution of chaincode, but only a subset of peers. These peers are defined by Endorsement policy and called endorsing peers or endorsers.
3. **Ordering Nodes** - a cluster of the replica nodes which exposes an abstraction of atomic broadcast to establish total order in all transactions within HLF. Ordering nodes are completely oblivious to the application state and do not participate in transaction validation or execution.

Membership Service Provider

The MSP is responsible for creating identities for peers and users of the organization. The identities of peers must be configured in an existing network in order for a new entity to participate in a channel. Fabric Certification Authority is an implementation of MSP and provides a mechanism for registering users and provide them with identities (X.509 certificates).

Orderers

A cluster of nodes that orders transactions into blocks on a first-come-first-serve fashion for all channels of the network. The ordering service is consensus agnostic thus it is easy to support different consensus implementations. The ordering service binds the network together, it holds the identity material tied to each member of the network. It is designed to support pluggable consensus implementations.

4.1.3 Privacy

In this subsection, we define Fabric's privacy components

Channels

Channels isolate transactions from the main Fabric network, they provide a private communication pipe between specific network members where they can conduct private and confidential transactions. A channel is defined by members, anchor peers, the shared ledger for the channel, chaincode, and the ordering service. Each transaction on the network is executed on a channel, where each party is authenticated. They also bring inherent parallelism to transaction processing since multiple channels can run on the same peer. We could think of channels as subsets of the main

HLF network where they share common services, but each channel acts as its own entity.

Identity Mixer

Credentials and other cryptographic material provided by the MSP can be implemented with Identity Mixer (Idemix), a cryptographic protocol suite, which provides zero knowledge proofs. Idemix offers these privacy-preserving features:

1. **Anonymity**: the ability to transcact without being associated with the identity of the transactor.
2. **Unlinkability**: the ability of a single identity to send multiple transactions without revealing that the transactions were sent by the same identity.

4.2 Transaction Flow

In this section, we elaborate on transactions, how they are created, how they are verified and how they propagate through the network to finally end up to a block.

4.2.1 High level view of Transaction Flow

1. Client leverages Fabric's Standard Development Kit (SDK) [50] to form a Transaction Proposal (TXP), which includes: the channel name, the chaincode name to invoke and input parameters for the chaincode to be executed. Next, a client sends TXP to all endorsing peers to satisfy the endorsement policy for the given chaincode.

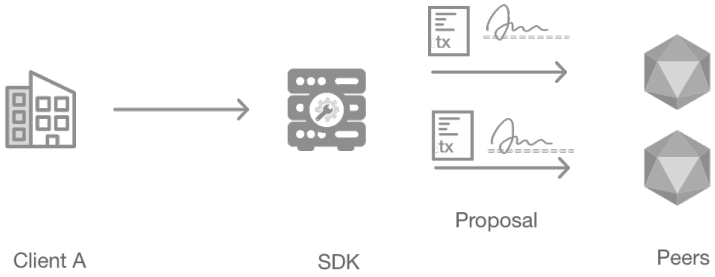


Figure 4.1: Client creates a TXP.

2. Endorsing peers simulate the transaction based on the parameters received from the client, by actually interacting with chaincode to record state updates and produce output in form of read-write set, followed by signing the read-write set and returning the results back to the client, Transaction Proposal Response (TXPR).

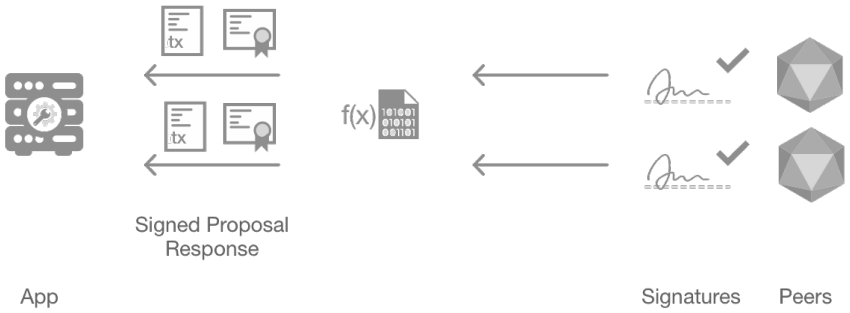


Figure 4.2: Endorsing peers simulate and sign TXP.

3. Client collects responses from endorsing peers, validates that results are consistent from every endorsing peer, e.g all endorsing peers have signed the same payload, followed by concatenation of all signatures of the endorsing peers along with the read-write sets, creating a transaction which is submitted to the ordering service.

Figure 4.3: Client collects signed TXPR.

4. Ordering service collects all incoming transactions, it does not need to inspect the entire content of a transaction to perform its operation. It receives the transactions from all channels in the network, orders them chronologically by channel, and creates blocks of transactions per channel.

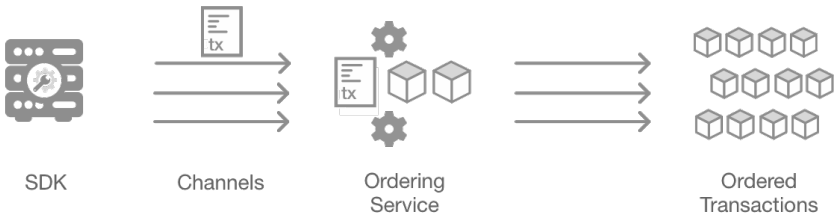


Figure 4.4: Ordering service creates block.

5. Peers of each organization, pull new blocks from the ordering service and disseminate them by using scalable middleware for ledger replication, whose implementation is based on an epidemic diffusion based protocol - gossip.

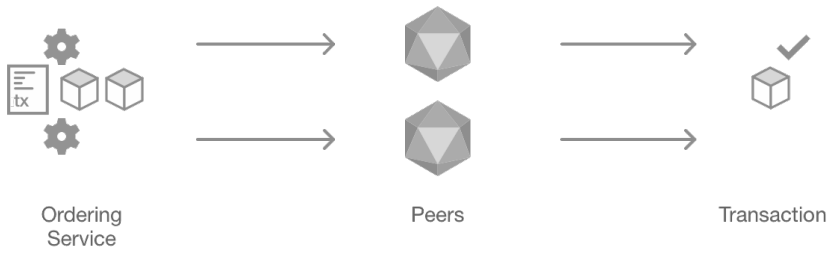


Figure 4.5: Peers get new blocks and gossip about them.

6. Each peer upon receiving a new block, iterates over transactions to:
   (a) validate the endorsement policy and
   (b) perform multi-value concurrency control checks.

   Once the transaction validation finishes, the peer appends the block to the ledger and updates its state, based on valid transactions. After the block is committed, it emits event to update the client connected to it.

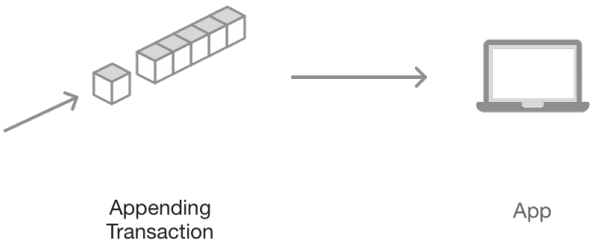


Figure 4.6: Blocks are added to the ledger.

Thus Fabric establishes an architecture of *execute-order-validate*.

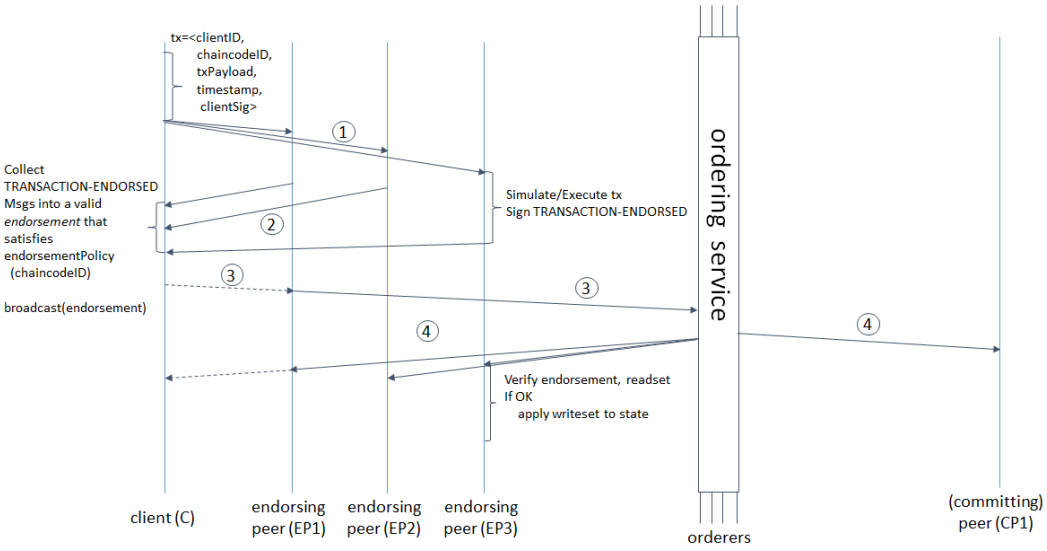


Figure 4.7: Flow diagram of a transaction.

4.2.2 Low level explanation of transaction flow

We have two types of transactions, *deploy* transactions which they create and install a new chaincode taking the program as an input, and *invoke* transactions which they refer to a chaincode function for either modifying the corresponding state or returning an output read-write.

To invoke a function of a chaincode, the client sends a `PROPOSE` message to a set of endorsing peers of its choice. Available endorsing peers are known by the endorsement policy of a particular `chaincodeID`. If they are online, they can choose whether to endorse the transaction or not. The client has to satisfy the policy expression with the endorsers available.

4.2.3 Propose message format

The `PROPOSE` message format is `<PROPOSE,tx,[anchor]>`, where `tx` is mandatory. `tx=<clientID, chaincodeID, txPayload, timestamp, clientSig>`

- `clientID`: ID of the submitting client
- `chaincodeID`: ID of the chaincode to which the transaction is applicable to
- `txPayload`: the payload containing the submitted transaction itself
- `timestamp`: monotonically increasing integer for every new transaction maintained by client
- `clientSig`: signature of client on other fields of `tx`

The details of the `txPayload` differ between invoke or deploy transactions. Specifically for an invoke transaction:

`txPayload = <operation, metadata>`

- `operation`: indicates the chaincode function and arguments
- `metadata`: indicates attributes related to the invocation

and for the deploy transaction:

`txPayload = <source, metadata, policies>`

- `source`: indicates the source code of the chaincode
- `metadata`: indicates attributes related to the invocation
- `policies`: contains endorsement policy ID and its parameters for the policies related to the chaincode that are accessible to all peers.

The hash of `tx` is used by all nodes as a unique transaction identifier `tid` and the client stores it in memory awaiting for responses from endorsing peers.

4.2.4 Blockchain data structure

State

The latest state of the blockchain, called World State, is modeled as a versioned key-value store (KVS). The stored data (value) and the key information for identifying the data (key) are stored as a pair. The values are stored and retrieved using a key that uniquely identifies the value and is used quickly to find the data within the database. These records are altered only by the chaincodes running on the blockchain through `put` and `get` KVS-operations. State is only maintained by peers. Keys can be recognized from their name by a particular chaincode, thus they can alter only the keys that they "possess".

Ledger

The ledger provides a verifiable history of all valid and invalid transactions, successful and unsuccessful changes to the state correspondingly. It is constructed by the ordering service as a totally ordered hash-chain of blocks. This hash-chain imposes the total order of blocks in a ledger and each block contains an array of totally ordered transactions, thus enforcing order across all transactions. Finally, every peer is responsible to maintain his ledger and it allows them to replay the history of all transactions and reconstruct the state.

Block

A block is batched, ordered transactions and contains some metadata and a link to the previous block via the block header. The main purpose of the block is to have a definite ordered entity to hash in order to include it in the next block. In figure 4.8 we present a visualization of the contents of a block.

4.3 Performance

Since we are interested in integrating IoT devices with Fabric we should elaborate on the state of Fabric's scalability, transaction throughput and performance overall. According to authors of [7] and of [13], Fabric can achieve over 2,500 transactions per second, which should more than suffice in a permissioned set up for IoT. Author's of [51] design and implement several architectural optimizations based on common system design techniques and improve Fabric's transaction throughput by sustaining 20,000 transactions per second.

This research points to a future Fabric that can be used as a large-scale consortium network having the capacity to handle a lot of transactions. For a perspective, VISA's network handles 24,000 [52].

4.4 Chaincode

In this section, we provide a brief example of an *invoke* chaincode and how a transaction would work from client's perspective.

In figure4.9 we present a basic chaincode to demo how to read/write to the ledger. It is installed on a peer and the functions in it get invoked by a client using the SDK.

First, we instantiate the chaincode and we put a key-value pair for the ledger to interact with it later. Then, we construct the functions that offer the utilities needed, read and write. Mostly, they are error checking and input sanitizing code, what we are interested in is `GetState` and `PutState`. Those two functions read and write to ledger if everything is ok. Functions, starting, with a small letter are private (Golang convention), this is why we have a gateway function `Invoke` that calls all the others.

Unlike Ethereum smart contracts, HLF allows a continuous integration and development. The cost to deploy a new smart contract, or update an already deployed one, is considered trivial.

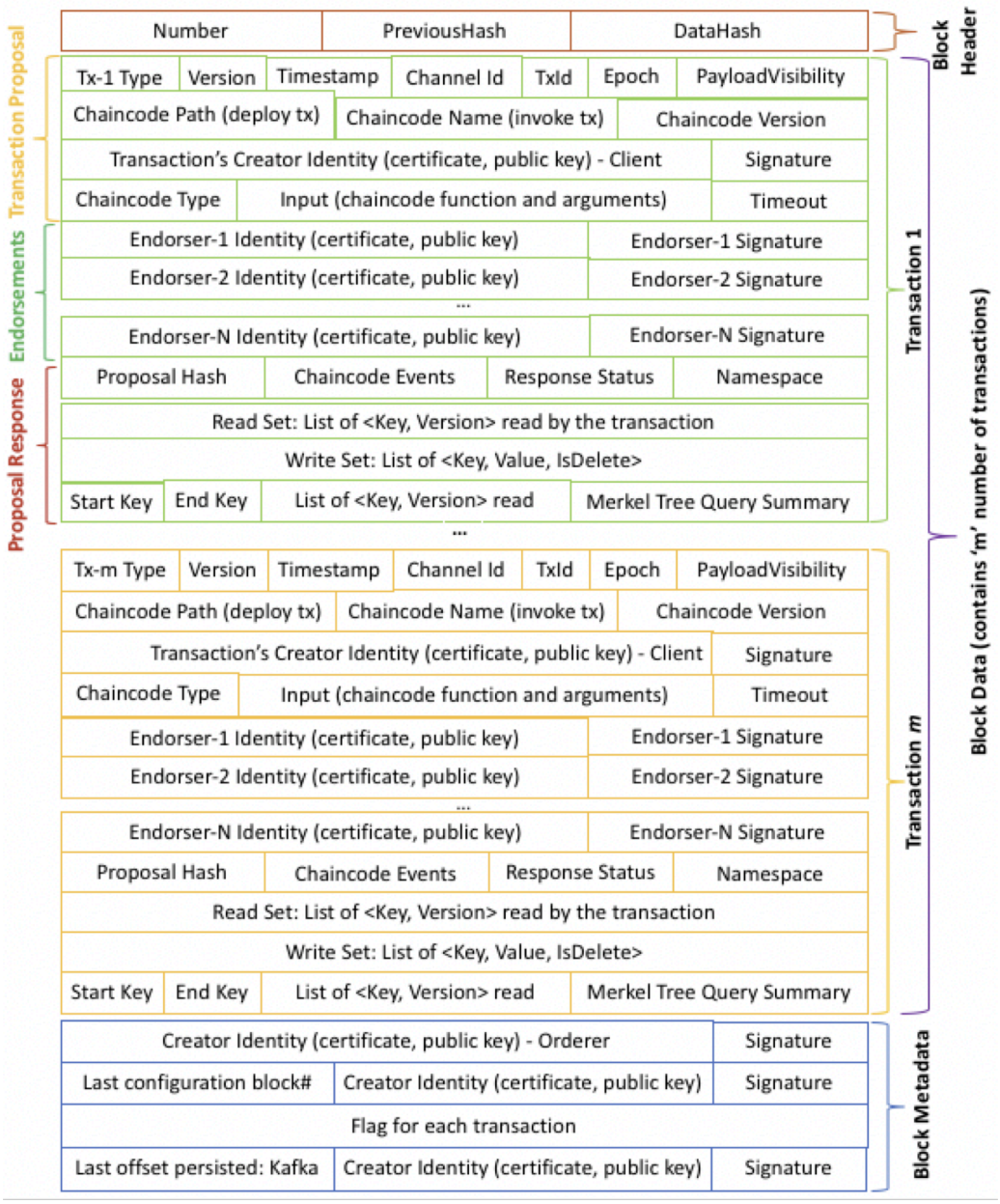


Figure 4.8: Inside a block [7]

```go
package main

import (
   "fmt"
   "github.com/hyperledger/fabric/core/chaincode/shim"
   pb "github.com/hyperledger/fabric/protos/peer"
)

type ExampleChaincode struct{
}

func main(){
   err := shim.Start(new(ExampleChaincode))
   if err != nil {
      fmt.Printf("Error starting example chaincode")
   }
}

// Instantiation of the chaincode
func (t *ExampleChaincode) Init(stub shim.ChaincodeStubInterface)
    pb.Response {

   // Get the function name and the parameters from the request
   functionName, _:= stub.GetFunctionAndParameters()

   if functionName != "init"{
      return shim.Error("Unkown function call")
   }
   // Put data in the ledger for demo key/value
   err := stub.PutState("key", []byte("value"))
   if err != nil {
      return shim.Error(err.Error())
   }

   return shim.Success(nil)
   }

// The gateway function for invokations.
func (t *ExampleChaincode) Invoke(stub shim.ChaincodeStubInterface)
    pb.Response {
   functionName, args := stub.GetFunctionAndParameters()


   if functionName == "init" {
      return t.Init(stub)
   } else if functionName == "read" {
      return t.read(stub, args)
   } else if functionName == "write" {
      return t.write(stub, args)
   }

   fmt.Println("invoked unknown functionName named " + functionName)
   return shim.Error("invoked unknown functionName named " +
       functionName)
}
```

Figure 4.9: An example chaincode demonstrating how to read and write using golang

```
1  func (t *ExampleChaincode) read(stub shim.ChaincodeStubInterface,
       args []string) pb.Response{
2     if args[1] == "hello" {
3        // Get the state of the value matching the key "key" in the
             ledger. It should be "value"
4        state, err := stub.GetState("key")
5        if err != nil {
6           return shim.Error("Failed to get state hello")
7        }
8
9        // return the value corresponding to the request as a response
10       return shim.Success(state)
11    }
12
13    return shim.Error("unknown query action, check the second arg")
14 }
15
16 func (t *ExampleChaincode) write(stub shim.ChaincodeStubInterface,
       args []string) pb.Response{
17
18    if args[1] == "key" && len(args) == 3 {
19       // Write the new value in the ledger
20       err := stub.PutState("key", []byte(args[2]))
21
22       if err != nil {
23          return shim.Error("Failed to update state of key")
24       }
25
26       return shim.Success(nil)
27    }
28    return shim.Error("Unknown invoke action, check the second
          argument.")
29 }
```

Figure 4.10: Callable functions of the chaincode, they define the functionality and the purpose of the chaincode

Chapter 5

Related Work

Recently, platforms and applications that tend to merge IoT and blockchain have emerged from many diverse areas. This chapter presents related work companies and academia with some of the most characteristic applications that combine the two technologies.

5.1 Companies in the space

5.1.1 Filament's Blocklet

Filament is the company that produces Blocklet[53], a small scale device that enables various machines to transact with popular blockchain networks such as Fabric and Ethereum. Specifically they provide two similar devices, one with a USB interface for common machines and another one with OBD-II connector for vehicles with an additional functionality, such as economic transaction and freight monitoring.

5.1.2 Modum

Modum [54] is a Swiss-based company that produces a small-scale device that transacts with the Ethereum blockchain. It aims to provide data integrity for physical products, mainly focused on improving supply chain processes. Sensors on device record environmental conditions during freights and create transactions that target an Ethereum smart contract. This contract verifies sensor's readings each time products change ownership and it validates that the transaction meets the customer's standards. The solution includes devices that collect data, a mobile application to register, activate the sensor and a monitor tool to analyze the collected data.

5.1.3 Riddle & Code

Riddle & Code [55] has developed a hardware and software solution to help physical objects interface with a blockchain. At the heart of their services is a crypto-chip,

where the private key is stored in a HSM accessible and known only by the device. The purpose of this chip is to be embedded on a medium that offers wireless connectivity. Depending on the use case that might be a NFC tag on a tamper-proof sticker, an RFID or a fully fledged IoT device with BLE. Their main focus is to give identity to objects and handle data that come from IoT. Digital twins is a notable use case, they are virtual replicas of physical objects or systems that it's purpose is to handle information during the whole life cycle of a product [56].

5.1.4 Crypto-anchors

Crypto-anchors [57] are under development by IBM, abstractly they are tamper-proof digital fingerprints, easily verifiable, than can be embedded to products and linked to a blockchain. Particularly, it is a protocol that leverages custom made solutions for each family of use cases. They bridge the physical to digital world by using Artificial Intelligence and spectrometers that can be embedded in common smartphone cameras [58], optical codes or tiny computers. These solutions aim to fight counterfeit products, provenance of food, automotive parts and any alteration of a product through the supply chain.

5.1.5 Proxy-SDK

The Proxy-SDK [8] is a lightweight SDK that enables IoT devices transact with a Hyperledger Fabric network. It is developed by Gero Dittmann [2] and Jens Jelitto [3] from IBM Research Zürich and it is based on `fabric-sdk-go` [50] which is developed by SecureKey and IBM. In the near future, it is going to be open source and available for public use. This thesis explores *Proxy-SDK* and offers an architecture overview as well as an analysis with respect to performance, a crucial part to scale down and make it more efficient for IoT. The Proxy-SDK was presented at Hyperledger Global Forum 2018 [8]

5.2 Academic Work

This section discuses some publications on use cases.

5.2.1 Food Supply Chain

In paper, [59] the author proposes a system and demos how it works on the food supply chain with Hazard analysis and critical control points from the FDA. This system delivers real-time information to all supply chain members on the safety status of food products, reduce the risk of centralized information systems, and bring more secure, distributed, transparent, and collaborative attributes to the supply chain.

5.2.2 Vehicle to Vehicle communication

In paper [42], authors discuss the use of blockchain for road safety. Specifically Cooperative Intelligent Transportation System (C-ITS) enables inter-networking of vehicles to exchange messages and coordinate their actions. Current communication is built with a centralized PKI. The authors suggest that with certificates on the blockchain, the presence of a single point of failure would be avoided. Furthermore, the computational overhead would be reduced since it is only needed to lookup the certificate and not verify to the root of trust. The system would support vehicle registration, admission, misbehavior notification and revocation in a completely decentralized manner.

5.2.3 Energy Trading

Authors of [60] present a solution for efficient and secure energy trading in microgrids, energy harvesting networks and vehicle-to-grids [61]. Their solution is based on a consortium blockchain network with IIoT nodes trading their surplus energy with other nodes in a P2P fashion to locally satisfy energy demands, improve efficiency and decrease transfer loses.

Chapter 6

Analysis of Proxy-SDK

In this chapter, we present the architecture of the Proxy SDK, the proof of concept on an already deployed pharmaceutical project, potential use cases and we conclude with our performance findings.

6.1 Architecture

In this section, we present a high level overview of the Proxy SDK.

6.1.1 Overview

The architecture comprises two proxies, a stateful and a stateless, one has memory and the other does not. An MQTT broker acts as an intermediary for buffering transactions (stateful) and a forward proxy (stateless) which forwards transactions and events.

6.1.2 Changes on Transaction Flow

The Proxy SDK adds one more layer of communication to the regular transaction flow as stated in subsection 4.2.1. Everything is taking place before a TXP reaches the Fabric network. Particularly, a device leveraging the Proxy-SDK, makes a TXP and sends it to an MQTT broker. A proxy which is subscribed to the associated topic gets the transaction, meanwhile the device can go offline, since the proxy interacts only with the broker. The proxy then sends the TXP to the corresponding endorsing peers and the transaction flow continues as normal. In a few words, TXPR are delivered to the proxy from endorsers, the proxy publishes them to the MQTT broker, the clients gets the TXPR verifies them and assembles the final transaction. The final transaction is sent again to the MQTT broker, to the proxy and straight for ordering. Upon successful ordering and updating the ledger the clients receive a notification regarding the state of their final transaction.

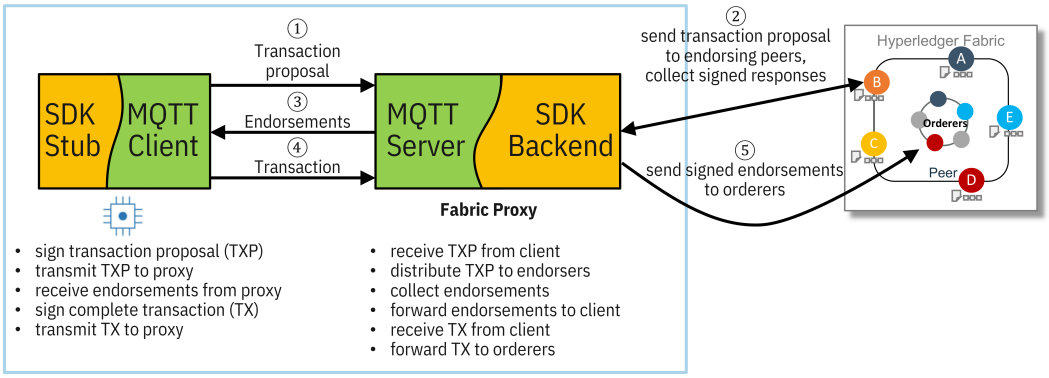


Figure 6.1: Proxy-SDK architecture [8]

6.2 Setup and Tools

In this section we present the setup for the proof of concept as well as tools used for the performance analysis.

6.2.1 Proof of Concept

As we presented in our case study in 3.3.2, the Proxy SDK is used in a blockchain prototype solution for tracking, tracing and cold chain monitoring of pharmaceutical products that was developed by IBM Research Zürich. Particularly, sensors measure the temperature of containers that carry medicine and pharmaceutical products. Transactions of temperature and humidity sensor data are carried out by a Raspberry Pi which serves as a development platform for small-scale IoT devices. Transactions are sent to an MQTT broker and the proxy listens to the relevant topics from the broker. The Fabric network consists of several peers (actors in the supply chain network) with a simplified single orderer scheme, running solo.

We continue with a demonstration of the application. In 6.2, the client gets authenticated, subscribes-publishes to the relevant topic and initiates a transaction. The proxy in 6.4, gets authenticated, subscribes to the relevant topic and listens for incoming transactions. The transaction, initiated by the client, eventually arrives at the proxy. In the meantime, proxy receives endorsement responses as seen in figure 6.5 and the client receives them from proxy. Finally in figure 6.3, the client sends the final transaction to the proxy. The proxy 6.6 receives it and sends it to the network for ordering.

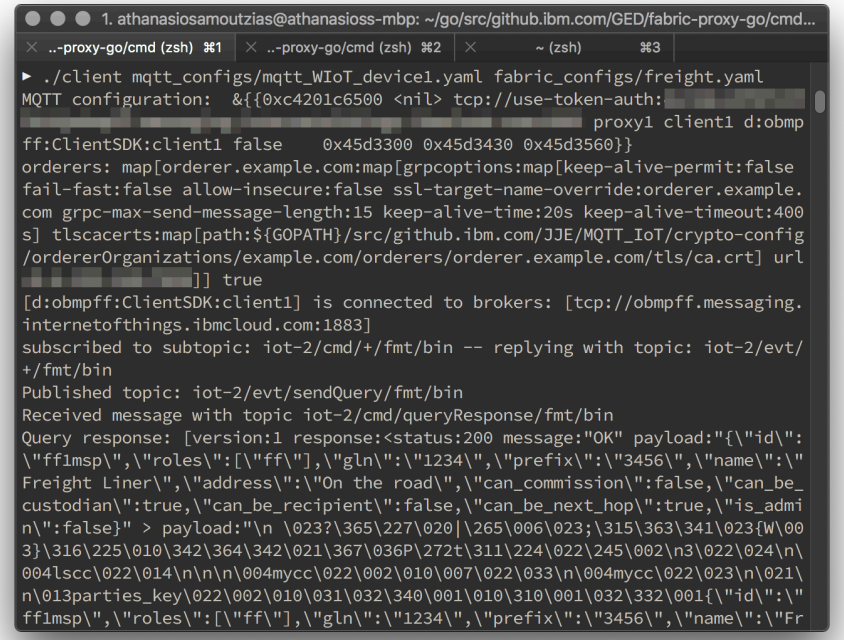


Figure 6.2: Client subscribes-publishes to a topic and initializes a transaction.

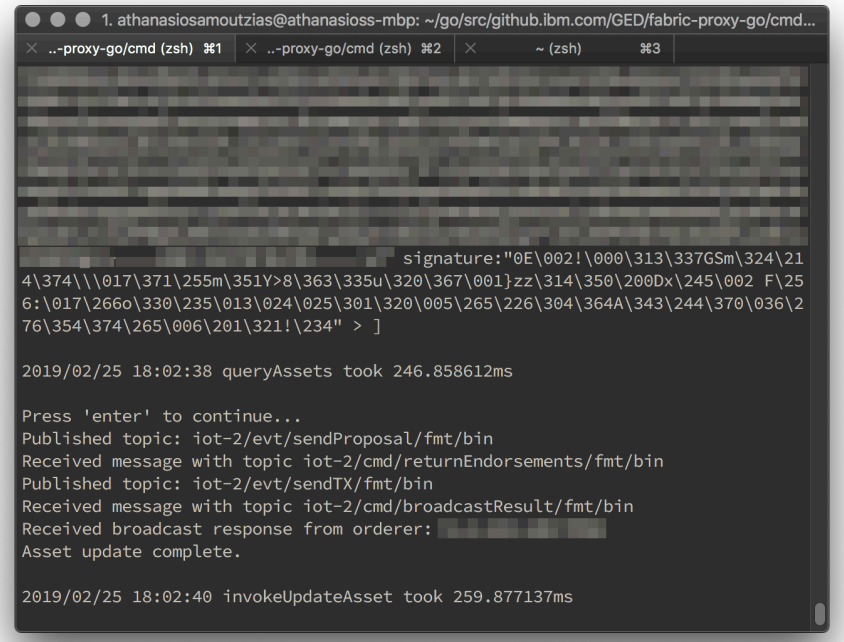


Figure 6.3: Client sends the final transaction.

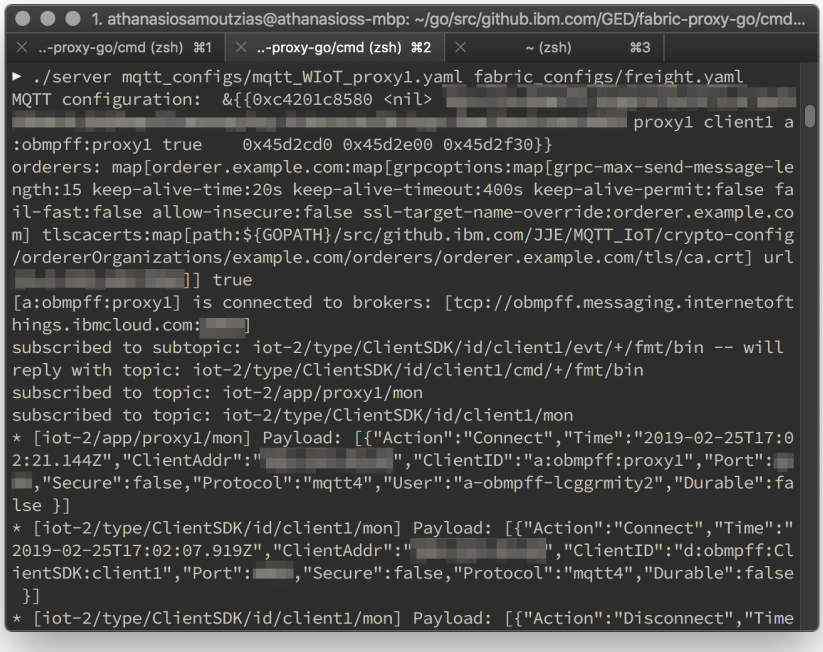


Figure 6.4: Proxy subscribes to topic and gets transaction from client.

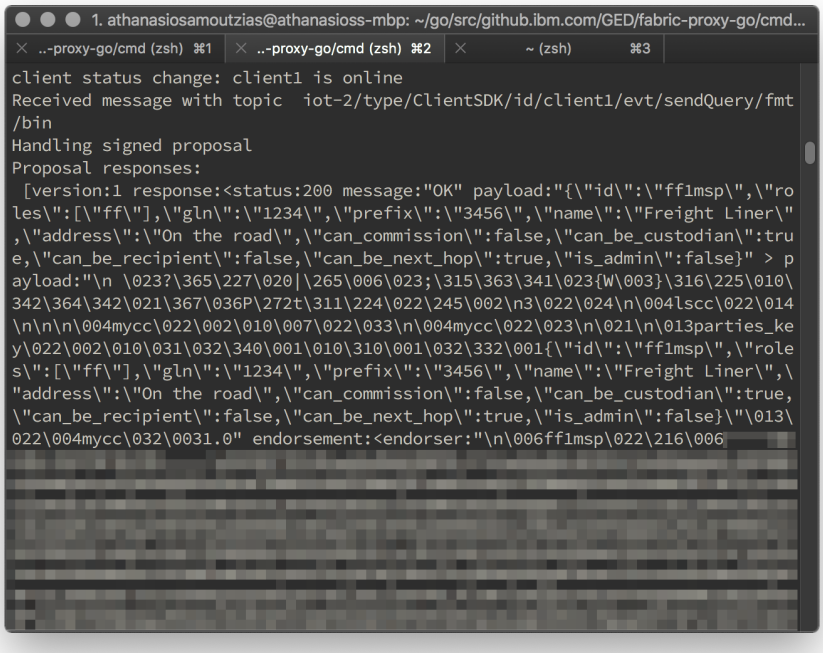


Figure 6.5: Proxy gets responses from endorsers.

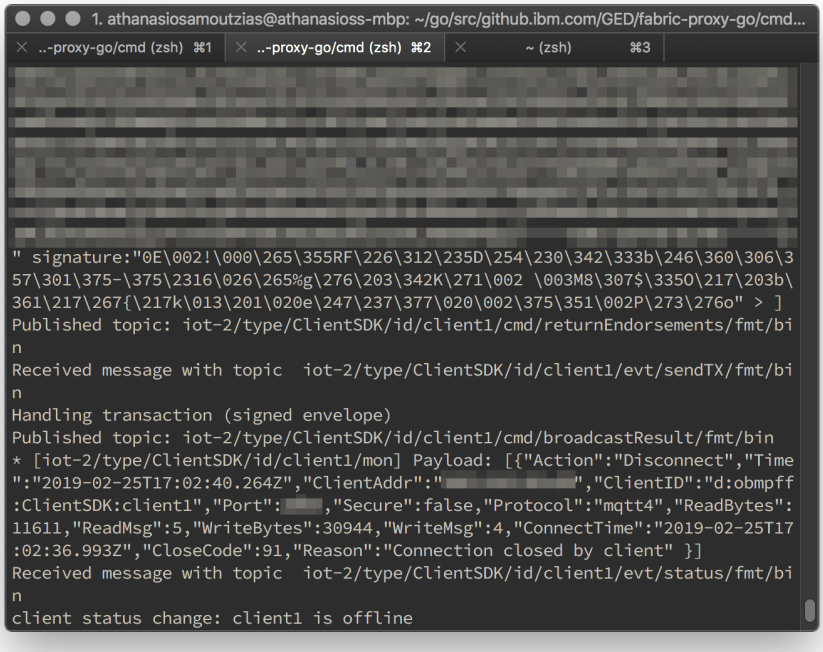


Figure 6.6: Proxy sends the final transaction to the Fabric network.

6.2.2 Tools

Wireshark

Wireshark is a packet analyser tool. It was used to troubleshoot the connection of the client to the proxy and Fabric network as well as check packet frequency and contents [62].

Activity monitor

MacOS native activity monitor was used to cross reference results from Wireshark and profiling on network bandwidth and RAM usage [63].

Golang

Go is a language developed by Google and it's built with high performance networking and multiprocessing in mind [64].

pprof - Golang's profiling tool

Profiling is a form of dynamic program analysis that measures the space complexity, the usage of particular instructions, frequency and duration of function calls and it

serves as an aid to program optimization. Profiles can track heat-points of the program and help the user look for methods that can lead to improvements. Google's Go language uses its own profiler called `pprof`. It reads a collection of profiling samples in `profile.proto` format and generates reports to visualize and help analyze the data. It can generate both text and graphical reports [9].

Hardware used

A Raspberry Pi used as a client and a laptop as a proxy. We should note here that Raspberry Pi does not offer cryptography acceleration hardware as modern fully fledged CPUs and that makes the cryptography algorithms slower and less efficient. [1]

Architecture	Model	Cores/Threads	Frequency	Machine
x86	Intel i7-4770HQ	4/8	2.2GHz	Macbook Pro
ARMv8	Broadcom BCM2837	4/4	1.2GHz	Raspberry Pi 3B

Table 6.1: Hardware used for benchmarking and testing.

6.3 Performance findings

In this section, we present our performance analysis results by first explaining the metrics used, then presenting the results and finally discussing them. The analysis was done with the lightweight client and results are from its perspective.

6.3.1 Metrics

Times

1. **User**: Amount of time CPU spent in user space, outside the kernel, within the process. The time the process spends blocked does not count towards this metric. It represents only the actual CPU time used in executing the process.
2. **System**: Amount of time CPU spent in the kernel space within the process. Mostly spent in system calls within the kernel, as opposed to library code, which is still running in user-space.
3. **CPU**: Total processor time used by process since it started, calculated as

$$cpu = user + system$$

. We use it as an indicator to compare different binaries with a given CPU.
4. **Wall**: Elapsed time of an executable, from start to finish.

[1] ARMv8, the architecture used in Raspberry Pi CPU, supports cryptography acceleration instructions but they are not implemented due to licensing costs.

5. **Utilization**: Percentage of actual work done by the cpu during execution of the program defined as

$$util = \frac{cpu}{wall}$$

Call Graphs

A control flow graph which represents calling relationships among functions in a program. Each node represents a procedure and each edge (f,g) indicates that procedure f calls g. A circle in the graph indicates recursive procedure calls. Profiler outputs dynamic call graphs that represent this execution of the program. Leaves account for the end functions that CPU spent most of its time.

Methodology and Approach

First, we start with CPU profiling. Inside the client's `main()` function we add the CPU profiling snippet shown in figure 6.7. Our main focus on profiling is the call graph where we investigate heat-points and how much time each part of the program consumes. Then, without the profiling snippet, since profiling introduces an overhead, we use the commands `time` to get our executable times. We continue with profiling the memory usage and get our call graphs. Again, with a fresh start, we capture packets and measure bandwidth usage.

6.3.2 CPU

Go is built for concurrency, the program uses all cores during execution. We considered to emulate a one-core architecture with no cryptography acceleration to emulate a Microcontroller Unit (MCU) but it turned quite difficult and time consuming.

In table 6.2, we see a notable decrease in CPU time from 130ms to 60ms making it a 53% decrease in CPU resources. The wall time for each transaction is higher due to the network overhead, since we introduced two more intermediaries, the MQTT broker and proxy.

SDK	on	user	sys	cpu	util	wall
Full	PC	70	60	130	7%	1.7s
Lite	PC	40	20	60	1%	3.1s
Lite	IoT	140	30	170	2%	7s

Table 6.2: CPU times in milliseconds.

Our conclusions from figure 6.8 [2] are that cryptographical operations account of 28% of the total executable usage, which is rather low considering that the Raspberry Pi does not have cryptography acceleration hardware. That means that cryptography operations are only a few and they introduce a small overhead. Accountable for

[2] Graph is zoom-able

```
func main() {
   flag.Parse()
   if *cpuprofile != "" {
      f, err := os.Create(*cpuprofile)
      if err != nil {
         log.Fatal("could not create CPU profile: ", err)
      }
      if err := pprof.StartCPUProfile(f); err != nil {
         log.Fatal("could not start CPU profile: ", err)
      }
      defer pprof.StopCPUProfile()
   }

   // ... rest of the program ...

   // memory profiling
   if *memprofile != "" {
      f, err := os.Create(*memprofile)
      if err != nil {
         log.Fatal("could not create memory profile: ", err)
      }
      runtime.GC() // get up-to-date statistics
      if err := pprof.WriteHeapProfile(f); err != nil {
         log.Fatal("could not write memory profile: ", err)
      }
      f.Close()
   }
}
```

Figure 6.7: Profiling method [9]

the rest of the usage are data structures, memory operations and the MQTT service. The flow of the program is straight forward and we do not observe any unnecessary communication back and forth.

We should note that profiling procedures that are fast and computationally inexpensive can be quite troublesome. One main reason is the resolution of the tool which is at 100Hz and we cannot easily observe leaks and inefficiencies that could accumulate over time.

6.3.3 Memory

In table 6.3, we observe that memory-wise both SDKs are at the same order, the executable size and the RAM they consume hardly differ.

SDK	RAM	Size
Full	3.1	25.7
Lite	3.5	27

Table 6.3: RAM consumed during execution and binary sizes, values in MB.

6.3.4 Bandwidth and Latency

In table 6.4, we see a notable decrease in network usage. Specifically, a 21% decrease in packets sent and a 81% decrease in packets received making it a 71% overall improvement. Since every KB in remote set-ups cost, a 71% reduced bandwidth is quite significant.

SDK	Sent	Rcvd	Sent Packets	Rcvd Packets
Full	14KB	90KB	47	112
Lite	11KB	17KB	28	31

Table 6.4: Bandwidth consumed for one transaction.

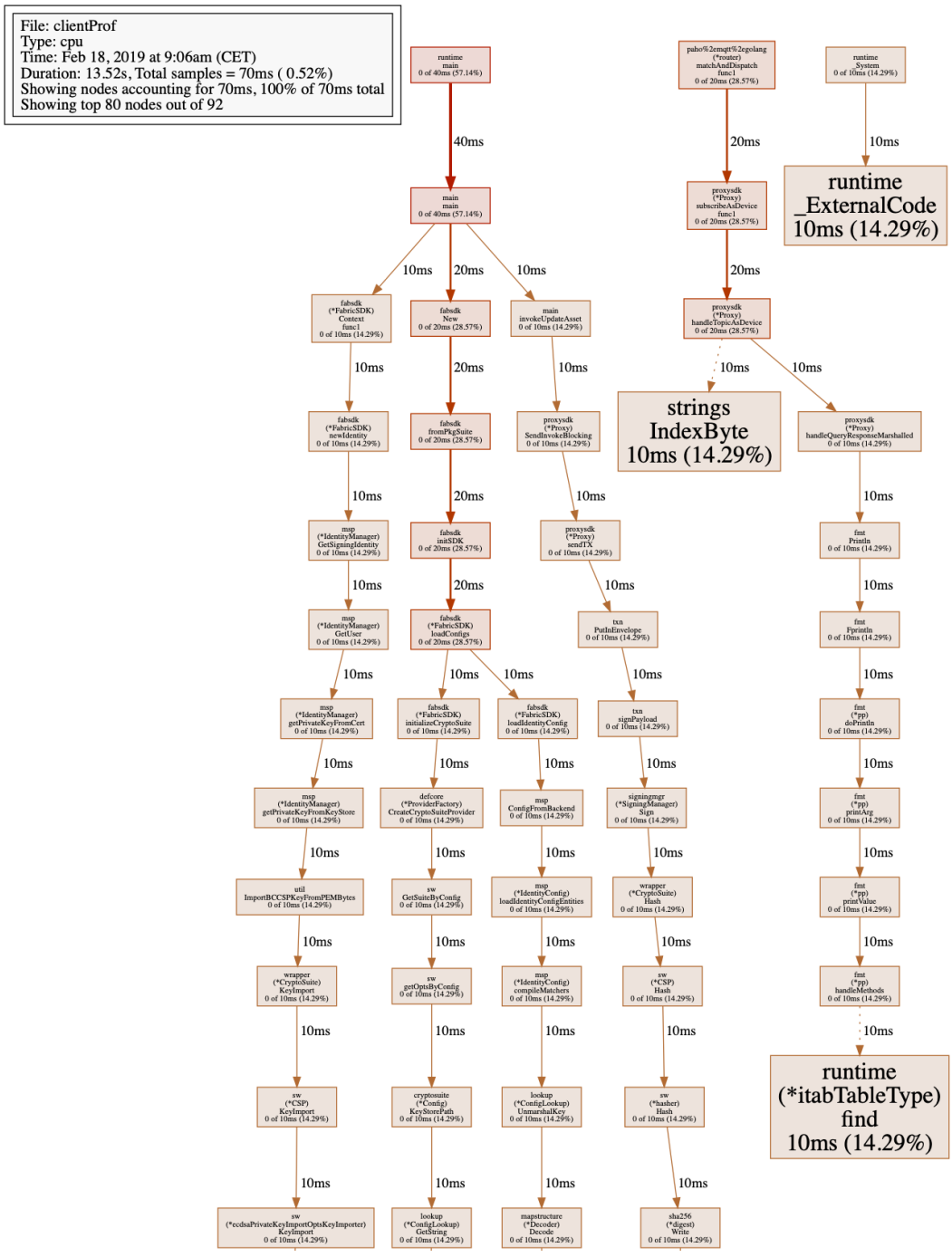


Figure 6.8: Call-graph of client on Raspberry Pi, leaves represent where the CPU spent most of its time.

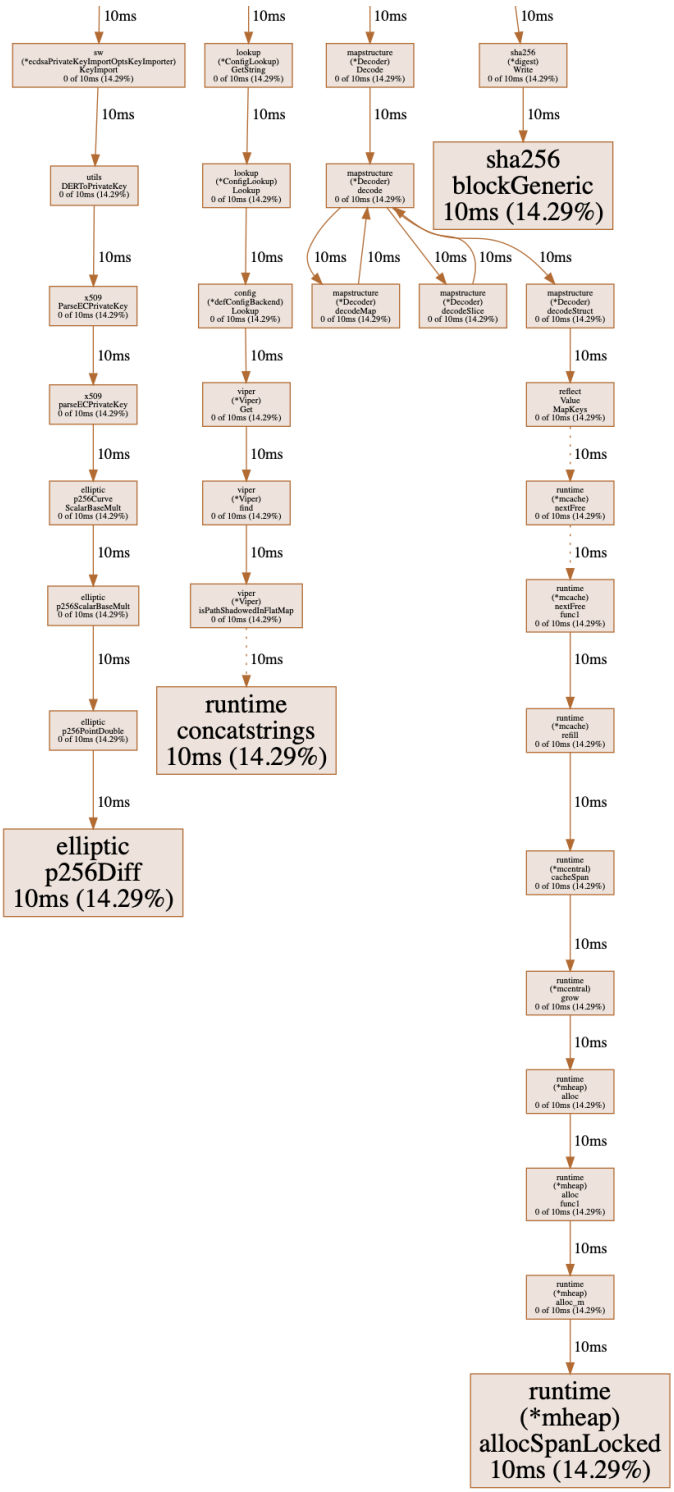


Figure 6.9: Bottom part of call-graph.

Chapter 7

Conclusions and Future work

7.1 Conclusions

The Proxy-SDK is indeed lighter than the Full SDK and can be used in a pilot program on an IoT device. Consortium blockchains are a good fit when a shared database is needed upon multiple incentive conflicted participants. Additionally, IoT can benefit from such networks, be more secure, define a common way of interacting and deciding on public data.

7.2 Future Work

Scaling down and minimizing computations is important for IoT devices, even more when resilience and a 24-7 operation are needed. A next step could be to port the client component of the Proxy-SDK in C and use a MCU coupled with a sensor to create transactions. There are solutions for cryptographic acceleration hardware [65], TEE and HSM to secure the keys in the MCU. Another important aspect to analyze is the behaviour of the proxy, how does it scale, do we need more proxies in order to secure up-time for the clients? Finally, bootstrapping techniques should be investigated to initialize devices on the network. []

Chapter 8

Bibliography

[1] CableLabs. A vision for secure iot, 2017. https://www.cablelabs.com/vision-secure-iot/[Online; accessed 11-March-2019].

[2] Gero dittmann ibm research, 2019. https://researcher.watson.ibm.com/researcher/view.php?person=zurich-GED[Online; accessed 11-March-2019].

[3] Jens jelitto ibm research, 2019. https://researcher.watson.ibm.com/researcher/view.php?person=zurich-jje[Online; accessed 11-March-2019].

[4] Wikimedia Commons. File:digital signature diagram.svg — wikimedia commons, the free media repository, 2018. [Online; accessed 25-February-2019].

[5] Ethereum evm illustrated, 2019. https://github.com/takenobu-hs/ethereum-evm-illustrated/tree/master/src[Online; accessed 19-March-2019].

[6] Bitnodes. Bitcoin nodes distribution, 2019. https://bitnodes.earn.com[Online; accessed 11-March-2019].

[7] Parth Thakkar, Senthil Nathan, and Balaji Vishwanathan. Performance benchmarking and optimizing hyperledger fabric blockchain platform. *arXiv preprint arXiv:1805.11390*, 2018.

[8] Gero Dittman, Jens Jelitto. Fabric proxy sdk, 2019. https://hgf18.sched.com/event/G8s7/sdk-proxy-hyperledger-fabric-identities-for-lightweight-iot-devices-g [Online; accessed 11-March-2019].

[9] Google. Profiling tools for golang, 2019. https://github.com/google/pprof[Online; accessed 11-March-2019].

[10] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008. https://bitcoin.org/bitcoin.pdf[Online; accessed 11-March-2019].

[11] Stuart Haber and W Scott Stornetta. How to time-stamp a digital document. In *Conference on the Theory and Application of Cryptography*, pages 437–455. Springer, 1990.

[12] Vitalik Buterin et al. Ethereum white paper, 2014. 2013. https://github.com/ethereum/wiki/wiki/White-Paper[Online; accessed 11-March-2019].

[13] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference*, page 30. ACM, 2018.

[14] Gemalto. Almost half of companies still can't detect iot device breaches, 2019. https://www.gemalto.com/press/Pages/Almost-half-of-companies-still-can-t-detect-IoT-device-breaches-reve aspx[Online; accessed 11-March-2019].

[15] Jonathan Katz, Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 1996.

[16] NCSU. Pki explained, 2018. http://discovery.csc.ncsu.edu/Courses/csc574-F09/slides/T07.2_PKI.6pp.pdf[Online; accessed 11-March-2019].

[17] . Bitcoin transaction fees, 2019. https://bitinfocharts.com/comparison/bitcoin-transactionfees.html[Online; accessed 11-March-2019].

[18] . Transactions in bitcoin, 2019. https://en.bitcoin.it/wiki/Transaction#Output[Online; accessed 11-March-2019].

[19] Bitcoin mempool statistics, 2019. https://jochen-hoenicke.de/queue/#0,24h[Online; accessed 19-March-2019].

[20] Bitcoin Wiki. Difficulty explained, 2019. https://en.bitcoin.it/wiki/Difficulty[Online; accessed 11-March-2019].

[21] Rory Bowden, Holger Paul Keeler, Anthony E Krzesinski, and Peter G Taylor. Block arrivals in the bitcoin blockchain. *arXiv preprint arXiv:1801.07447*, 2018.

[22] Bitcoin network hashrate, 2019. https://whattomine.com/coins/1-btc-sha-256[Online; accessed 19-March-2019].

[23] Gas prices in ethereum, 2019. https://ethgasstation.info/[Online; accessed 19-March-2019].

[24] Solidity operations and gas cost, 2019. https://github.com/ethereum/homestead-guide/blob/master/source/contracts-and-transactions/account-types-gas-and-transactions.rst#example-transaction-cost[Online; accessed 19-March-2019].

[25] Vitalik Buterin. On public and private blockchains. *Ethereum blog*, 7, 2015. https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/reyna2018blockchain[Online; accessed 11-March-2019].

[26] Omar Dib, Kei-Leo Brousmiche, Antoine Durand, Eric Thea, and E Ben Hamida. Consortium blockchains: Overview applications and challenges. *International Journal On Advances in Telecommunications*, 11(1&2), 2018.

[27] Council of European Union. Council regulation (EU) no 2016/679, 2016. https://eur-lex.europa.eu/eli/reg/2016/679/oj.

[28] Christian Reitwiessner. zksnarks in a nutshell. *Ethereum Blog*, 6, 2016. http://chriseth.github.io/notes/articles/zksnarks/zksnarks.pdf[Online; accessed 11-March-2019].

[29] Okke Schrijvers, Joseph Bonneau, Dan Boneh, and Tim Roughgarden. Incentive compatibility of bitcoin mining pool reward functions. In *International Conference on Financial Cryptography and Data Security*, pages 477–498. Springer, 2016.

[30] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. *Communications of the ACM*, 61(7):95–102, 2018.

[31] Dionysis Zindros. Sybil-attack. https://bitcoin.stackexchange.com/questions/50922/whats-a-sybil-attack[Online; accessed 11-March-2019].

[32] Bennett Garner. What's a sybil attack & how do blockchains mitigate them?, 2018. https://coincentral.com/sybil-attack-blockchain/[Online; accessed 11-March-2019].

[33] Andrew Miller. Ethereum isn't turing complete and it doesn't matter anyway, 2016. https://docs.google.com/presentation/d/1OklfhEV4Rs-wBW_zrWlC6cu4F3fJxL6YOE2MJp5eXTU/edit#slide=id.g1578fe1a7c_1_88[Online; accessed 11-March-2019].

[34] Fran Casino, Thomas K Dasaklis, and Constantinos Patsakis. A systematic literature review of blockchain-based applications: current status, classification and open issues. *Telematics and Informatics*, 2018.

[35] Mark Hung. Leading the iot, gartner insights on how to lead in a connected world. *Gartner Research*, pages 1–29, 2017.

[36] Constantinos Kolias, Angelos Stavrou, and Jeffrey Voas. Securely making" things" right. *Computer*, (9):84–88, 2015.

[37] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, and Jeffrey Voas. Ddos in the iot: Mirai and other botnets. *Computer*, 50(7):80–84, 2017.

[38] Uzair Javaid, Ang Kiang Siang, Muhammad Naveed Aman, and Biplab Sikdar. Mitigating lot device based ddos attacks using blockchain. In *Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems*, pages 71–76. ACM, 2018.

[39] Ana Reyna, Cristian Martín, Jaime Chen, Enrique Soler, and Manuel Díaz. On blockchain and its integration with iot. challenges and opportunities. *Future Generation Computer Systems*, 2018.

[40] Dmitry Khovratovich and Jason Law. Sovrin: digital identities in the blockchain era. *Github Commit by jasonalaw October*, 17, 2017.

[41] Christian Lundkvist, Rouven Heck, Joel Torstensson, Zac Mitton, and Michael Sena. Uport: A platform for self-sovereign identity. *URL: https://whitepaper. uport. me/uPort_ whitepaper_DRAFT20170221. pdf*, 2017.

[42] Noureddine Lasla, Mohamed Younis, Wassim Znaidi, and Dhafer Ben Arbia. Efficient distributed admission and revocation using blockchain for cooperative its. In *New Technologies, Mobility and Security (NTMS), 2018 9th IFIP International Conference on*, pages 1–5. IEEE, 2018.

[43] Conner Fromknecht, Dragos Velicanu, and Sophia Yakoubov. A decentralized public key infrastructure with identity retention. *IACR Cryptology ePrint Archive*, 2014:803, 2014.

[44] IBM. Tradelens: How ibm and maersk are sharing blockchain to build a global trade platform, 2019. `https://www.ibm.com/blogs/think/2018/11/tradelens-how-ibm-and-maersk-are-sharing-blockchain-to-build-a-globa` accessed 11-March-2019].

[45] we.trade, 2019. `https://www.ibm.com/case-studies/wetrade-blockchain-fintech-trade-finance`[Online; accessed 19-March-2019].

[46] IBM. Ibm blockchain world wire: Let's clear and settle cross-border payments in seconds – not days, 2019. `https://www.ibm.com/blockchain/solutions/world-wire`[Online; accessed 11-March-2019].

[47] IBM. Ibm food trust: adding trust and transparency to our food, 2019. `https://www.ibm.com/blockchain/solutions/food-trust`[Online; accessed 11-March-2019].

[48] IBM. Ibm digital identity: Transforming digital identity into trusted identity, 2019. `https://www.ibm.com/blockchain/solutions/identity`[Online; accessed 11-March-2019].

[49] Fred B Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys (CSUR)*, 22(4):299–319, 1990.

[50] Secure Key. Fabric sdk in go, 2019. `https://github.com/hyperledger/fabric-sdk-go`[Online; accessed 11-March-2019].

[51] Christian Gorenflo, Stephen Lee, Lukasz Golab, and S Keshav. Fastfabric: Scaling hyperledger fabric to 20,000 transactions per second. *arXiv preprint arXiv:1901.00910*, 2019.

[52] Visa. Security and reliability, 2019. `https://usa.visa.com/run-your-business/small-business-tools/retail.html`[Online; accessed 15-March-2019].

[53] Filament. Filament blocklet chip, 2018. *URL*, 2018. `https://filament.com/wp-content/uploads/2018/12/Filament-v3.0-White-Paper.pdf`[Online; accessed 11-March-2019].

[54] modum. modum, 2018. `https://modum.io/`[Online; accessed 11-March-2019].

[55] Riddle & Code. The blockchain interface company, 2018. `https://www.riddleandcode.com/`[Online; accessed 11-March-2019].

[56] M Sallaba, D Siegel, and S Becker. Iot powered by blockchain–how blockchains facilitate the application of digital twins in iot. *Blockchain-Institute. On the internet: https://www2. deloitte. com/content/dam/Deloitte/de/Documents/Innovation/IoT-powered-by-Blockchain-Deloitte. pdf, access*, 19:2018, 2019.

[57] IBM. Ibm crypto anchors, 2018. `https://www.research.ibm.com/5-in-5/crypto-anchors-and-blockchain/`[Online; accessed 11-March-2019].

[58] IBM. Pairing ai with optical scanning for real-world product authentication, 2018. `https://www.ibm.com/blogs/research/2018/05/ai-authentication-verifier/`[Online; accessed 11-March-2019].

[59] Feng Tian. A supply chain traceability system for food safety based on haccp, blockchain & internet of things. In *Service Systems and Service Management (ICSSSM), 2017 International Conference on*, pages 1–6. IEEE, 2017.

[60] Zhetao Li, Jiawen Kang, Rong Yu, Dongdong Ye, Qingyong Deng, and Yan Zhang. Consortium blockchain for secure energy trading in industrial internet of things. *IEEE transactions on industrial informatics*, 14(8):3690–3700, 2018.

[61] Jiawen Kang, Rong Yu, Xumin Huang, Sabita Maharjan, Yan Zhang, and Ekram Hossain. Enabling localized peer-to-peer electricity trading among plug-in hybrid electric vehicles using consortium blockchains. *IEEE Transactions on Industrial Informatics*, 13(6):3154–3164, 2017.

[62] Wireshark, 2019. https://www.wireshark.org/[Online; accessed 11-March-2019].

[63] Apple. Performance tools, 2019. https://developer.apple.com/library/archive/documentation/Performance/Conceptual/PerformanceOverview/PerformanceTools/PerformanceTools.html[Online; accessed 11-March-2019].

[64] Google. Golang, 2019. https://golang.org/[Online; accessed 11-March-2019].

[65] Cryptographic co-processor with secure hardware-based key storage, 2019. https://www.microchip.com/wwwproducts/en/ATECC608A[Online; accessed 19-March-2019].

[66] Hyperledger. Hyperledger fabric documentaions. https://hyperledger-fabric.readthedocs.io/en/release-1.4/[Online; accessed 11-March-2019].

[67] Konstantinos Christidis and Michael Devetsikiotis. Blockchains and smart contracts for the internet of things. *Ieee Access*, 4:2292–2303, 2016.

[68] Andreas M Antonopoulos. *Mastering Bitcoin: unlocking digital cryptocurrencies*. " O'Reilly Media, Inc.", 2014.

[69] Andreas M Antonopoulos and Gavin Wood. *Mastering Ethereum: Building Smart Contracts and Dapps*. O'Reilly Media, 2018.

[70] Wikipedia contributors. Pale blue dot — Wikipedia, the free encyclopedia, 2018. [Online; accessed 9-March-2019].

[71] Xiaoyang Zhu, Youakim Badr, Jesus Pacheco, and Salim Hariri. Autonomic identity framework for the internet of things. In *Cloud and Autonomic Computing (ICCAC), 2017 International Conference on*, pages 69–79. IEEE, 2017.

[72] Mqtt, 2018. https://mqtt.org[Online; accessed 11-March-2019].

[73] acmqueue. Flame graphs explained, 2016. https://queue.acm.org/detail.cfm?id=2927301[Online;accessed 11-March-2019].

[74] IBM. Blockchain marbles, 2019. https://github.com/IBM-Blockchain/marbles[Online; accessed 11-March-2019].

[75] Pureswaran Veena, Sanjay Panikkar, Sumabala Nair, and Paul Brody. Empowering the edge-practical insights on a decentralized internet of things. *Empowering the Edge-Practical Insights on a Decentralized Internet of Things. IBM Institute for Business Value*, 17, 2015.

[76] Bitcoin energy consumption index, 2019. https://digiconomist.net/bitcoin-energy-consumption[Online; accessed 19-March-2019].

[77] Ambrosus: Decentralised iot networks, 2019.
`https://ambrosus.com`[Online; accessed 19-March-2019].