Assessing blockchain technology for Transport Data Logger

BJÖRN JOHANSSON
MASTER´S THESIS
DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

Assessing blockchain technology for Transport Data Logger

Master's Thesis

By

Björn Johansson

Department of Electrical and Information Technology
Faculty of Engineering, LTH, Lund University
SE-221 00 Lund, Sweden

LUND UNIVERSITY

2017

Abstract

A proof of concept permissioned blockchain system that holds data from the Transport Data Logger, TDL, was developed. Transport chain actors who hand off TDL-equipped goods between one another use the TDL app to upload a digital handoff containing their identities and log data from and signed by the TDL to a Hyperledger Fabric blockchain network.

The current Transport Data Logger system consists of a sensor-equipped device that interacts with a companion smartphone app using Bluetooth. The device is fastened to goods and will log temperature, humidity and more during transport. Upon delivery the recipient can connect to the TDL device with the smartphone app and retrieve the log data. The TDL system lacks cryptographic data protection and attribution of data points to specific transport chain actors other than in the form of comparing timestamps.

The promise of blockchain is network-distributed, decentralized and immutable data storage and transaction conduction. Many very differing implementations of the blockchain concept exist, with their common factor being the sorting of data into an append-only list of blocks chronologically and cryptographically linked to one another in linear sequence. The greatest divisor of blockchain technology is between permissioned and permissionless blockchains. A permissioned blockchain system limits participation to known and approved entities, allowing much better performance at the cost of increased centralization. Full trust between participants is still not required, which is the great advantage of blockchain technology.

There is potential for a gap in historical truth, what happened, and documental truth, what is documented as having happened. A blockchain is fully capable of protecting its data, or the documental truth, from tampering but closing the gap between the documental and historical truths requires more than just using blockchain technology.

Blockchain technology is still nascent, and mainstream use beyond cryptocurrencies is years off. There is a large reliance on hype and signaling with blockchain technology, which means that much research and development of standards is needed before it can be properly treated as the cryptosystem it should be interpreted as. The thesis recommendation on blockchain adoption is two-fold: start now if getting in on the ground floor and helping to develop the underlying systems and standards is important, otherwise there is everything to gain from waiting until that work has been done, as the blockchain technology sector will be very volatile until then.

Acknowledgements

I would like to thank my friends and family whose emotional support helped make this thesis possible. Special mention to my beloved companion Lisa the Golden Retriever.

Björn Johansson

Contents

Abstract ..... **i**

Acknowledgements ..... **iii**

Contents ..... **v**

List of acronyms ..... **ix**

Popular Science Summary: Storing and attributing log data using blockchain technology ..... **xi**

1. Introduction **1**

1.1. Background 1

1.2. Problem Statements 2

1.3. Method 2

1.4. Related Work 4

1.5. Thesis Outline 5

Part I. Theory **7**

2. Cryptography **9**

2.1. Kerckhoffs' principle 10

2.2. Hashing 10

2.3. Symmetric cryptography 11

2.3.1. Exchanging secret keys 12

2.4. Asymmetric cryptography 13

2.4.1. Key pair 13

2.4.2. Asymmetric encryption and decryption 13

2.5. Signing 14

2.6. Tying public keys to entities 15

3. Blockchain technology **19**

3.1. Concept 19

3.2. Consensus 22

3.2.1. Byzantine nodes 23
3.2.2. Sybil attacks 24
3.3. Chaincode and smart contracts 24
3.4. State of blockchain security 25
3.5. Permissionless blockchains 26
3.5.1. Proof-of-work 27
3.5.2. Proof-of-stake 29
3.6. Permissioned blockchains 29
3.6.1. The Hyperledger project 29
3.6.2. Proof of elapsed time 30
3.6.3. Byzantine fault tolerant consensus 31
3.7. Hyperledger Fabric 33
3.7.1. Consensus and node responsibilities 33
3.7.2. Transaction process 34
3.7.3. Participation and permissions 35
3.8. Summary 36
4. Transport Data Logger 37
4.1. Tracking conditions during shipping 37
4.2. Baseline version 38
4.3. Security solution 38
4.3.1. Assumptions 39
4.3.2. PIN protection 39
4.3.3. Data protection 40
Part II. Evaluation 41
5. TDL blockchain system 43
5.1. Reasons for using blockchain 43
5.1.1. Tracking responsibility 43

5.1.2. Data reliability 44
5.1.3. Assumptions 44
5.2. Choosing blockchain type and system 45
5.2.1. Important criteria 46
5.2.2. Decision tree 46
5.3. Proof-of-concept implementation 48
5.3.1. Versions used 48
5.3.2. Use case 49
5.3.3. Technical overview 54
5.3.4. Cryptographic materials 58
5.3.5. TDL data summary 58
5.3.6. Semi-automatic PIN handling 58
5.4. Summary 59
Part III. Conclusions 61
6. Discussion 63
6.1. Blockchain technology 63
6.1.1. Maturity, hype and signaling 63
6.1.2. Permissionless vs permissioned blockchains 64
6.2. The TDL blockchain system and its proof of concept 65
6.2.1. Historical vs documental truth 66
6.2.2. Other blockchain system solutions 67
7. Future work 69
8. Summary 71
8.1.1. Question 1 – Permissioned vs permissionless blockchains . 71
8.1.2. Question 2 – Data reliability 71
8.1.3. Question 3 – Data attribution 72
8.1.4. Recommendation on blockchain technology adoption 72

References 73

List of acronyms

API	Application Programming Interface
BFT	Byzantine Fault Tolerance (alt. Byzantine Fault Tolerant)
CA	Certificate Authority
CPU	Central Processing Unit
HTTPS	Hypertext Transfer Protocol Secure
MSP	Membership Service Provider
PBFT	Practical Byzantine Fault Tolerance
PKI	Public Key Infrastructure
PoET	Proof of Elapsed Time
REST	Representational State Transfer
SDK	Software Development Kit
SGX	Software Guard Extensions
SHA	Secure Hash Algorithm
SSH	Secure Shell
TDL	Transport Data Logger
TLS	Transport Layer Security

Popular Science Summary: Storing and attributing log data using blockchain technology

Blockchain technology is incredibly young, even the most mature of the blockchain technologies are either functionally complete but very simple or lack important features. The thesis has assessed blockchain technology for implementation with the Bosch Transport Data Logger, TDL.

The TDL is a sensor-equipped logging device with a companion smartphone app. The device is fastened to packages during transportation and logs how the packages have been handled while in the care of post and transport companies. For example, unacceptable temperatures, humidity and manhandling are logged.

The two main types of blockchain technology, permissionless and permissioned, are found to have inherent trade-offs between open/closed participation and poor/good performance. Hyperledger Fabric, most mature of the permissioned blockchain systems, was used to build a TDL Blockchain System proof of concept. Log data and other information from and about the TDL devices, transferred by the transport company employees using the companion app, are stored on the blockchain.

Generally, just saving sensor values is not enough to accomplish anything. To give credence to the logged data and to be able confidently to act upon it, the data must be made trustworthy. In an environment where data is generated while the sensor device is in the hands of different organizations, trustworthy connections between each data point and the organization responsible must be made. This to prevent blame-shifting of the type “it was like that when I got it”. To bring historical and documental truth closer to one another, further links between the real world and the data must be made, provided in part by devices needing to digitally sign the data logged by them.

The thesis can be used by Bosch to learn about blockchain technology and its applicability for the TDL. If they or others want to extrapolate they could use the thesis results to draw conclusions for other similar devices and systems. The general thesis recommendation is two-fold. If an organization simply wants to implement blockchain technology applications and not help develop the underlying technology the blockchain technology sector is not yet ready for them, otherwise now is a good time to jump in.

1. Introduction

As the steps in a transport chain increases the more distrust between transport chain actors (companies) can be expected as those who commit errors while handling sensitive goods would most likely want to shift the blame to another actor in the transport chain. The Bosch Transport Data Logger, TDL, was created to determine what conditions a package equipped with a TDL device is exposed to during transport.

This thesis will look at how blockchain technology can be used to make the data generated by the TDL more reliable and how it can be used to pinpoint what transport chain actor had responsibility for the package when any sort of unacceptable conditions were logged.

1.1. Background

The Transport Data Logger system consists of a sensor-equipped device that interacts with a companion smartphone app. The device is fastened to goods during transport and set up using the app to tolerate certain threshold sensor values including temperature, impacts and humidity (and more), with readings exceeding those values being logged as violations. The recipient can then connect to the TDL device with the smartphone app upon delivery of the goods and then view (and share) the log data.

The problem of “it was like this when I got it” can potentially be solved by logging if and when conditions damaging to the goods occur at any time during transportation, recording every handoff between actors in the transport chain and protecting the recorded data from manipulation.

In its current form the TDL system lacks cryptographic protection or attribution of violation data to specific transport chain actors, other than in the form of manually comparing timestamps.

Data verification today relies heavily on having a common trusted third party that vouches for the dependability, source and integrity of that data.

Blockchain technology promises to provide network-distributed, decentralized and immutable data storage and transaction conduction. This would eliminate the need to completely trust a single third party to verify the integrity or existence of some data. Instead trust is put in the collective that is made up of entities collaborating in a blockchain network.

Some form of consensus building methodology is needed to make sure that the blockchains at all entities nodes are identical, making the figurative

collective blockchain stable. All good consensus building methods are tamper resistant in that for tampering to occur, many different actors on the blockchain network must collaborate. With a traditional single centralized database only direct changes to the database would be needed for tampering to succeed.

The brute-force nature of creating new blocks for proof-of-work consensus blockchains has led to that form of block creation being called “mining”.

1.2. Problem Statements

These three questions are good to keep in the reader’s mind as they are the guiding lights of the thesis report.

Question 1. What are the different strengths and weaknesses of permissioned and permissionless blockchains?

To decide what type of blockchain system should be used for the TDL Blockchain System, the defining characteristics of the two main types of blockchain technology must be determined. Identifying how blockchain technology works is also important in being able to answer the other questions in the problem statement.

Question 2. How can blockchain technology be used to increase the reliability of the Transport Data Logger data?

Question 3. How can blockchain technology be used to attribute specific data points from the Transport Data Logger to specific transport chain actors?

To be able to use the data generated by the TDL to make well-informed decisions that data must be determined to be reliable, meaning accurate and unmanipulated. It must also be possible to directly attribute specific data to transport chain actors, preventing them from shifting blame away from themselves.

1.3. Method

This thesis is exploratory in nature, at the beginning a vague approximation of what would later become Question 2 and Question 3 as well as the intent to develop a proof of concept was used to create a schedule and method outline (a project plan).

As the author could not for several very important reasons commit any more time to the thesis than the standard one semester this thesis was from the start conducted to fit inside of those 20 weeks (5 months). Other thesis

projects known to the author had exhibited a tendency to expand time to fit the scope and to prevent that from happening care was taken from the start to create and stick to a reasonable schedule, method and scope. The schedule outline looked roughly like the following: development finished by end of month 3, complete draft of thesis report before end of month 4, presentation done before end of month 5. As time progressed this schedule outline was filled in with considerably more details. The phases of the thesis project are illustrated in Fig. 1.

Month 1	**Month 2**	**Month 3**	**Month 4**	**Month 5**
Research	Development	Development	Writing	Writing Presenting Finishing up

Fig. 1. The method outline showing the 5 months of the project and the phases they contained. This is a simplified version of reality as for example outline drafts of the thesis were created during the research phase and research in one form or another was conducted during almost all project phases.

The method outline was to spend the first month exploring the blockchain technology field and doing much of the necessary research for the thesis. This included Lund University Libraries[1] and Swedish thesis report[2] searches for terms like, or related to, "blockchain". It also included looking at interesting references and technologies from the results of those searches. The initial research phase also included looking through Bosch internal development documentation for the TDL, exploring the TDL's capabilities, and setting up all the practicalities required for development of the proof of concept. Initial decisions around the design of the TDL Blockchain System and its proof of concept based on the research findings were made by the end of this phase.

The next phase was the development phase that consisted of two months of software development of the TDL Blockchain system proof of concept. As everything in a blockchain system is dependent on the chaincode development started with that and then moved on to the network, and then client. Development progress and planning was strongly influenced by

[1] The LUBSearch service at website lubsearch.lub.lu.se was used.

[2] The National Library of Sweden Libris service for finding student papers called Uppsök at website http://uppsok.libris.kb.se/sru/uppsok was used.

difficulties encountered during the process. Decisions were always made with the knowledge that time was a limited resource.

Once the main use case for the TDL Blockchain System was supported the development phase was over. It was followed by a month-long thesis report writing phase. Writing the thesis was mostly done chapter by chapter. An outline of the thesis chapters and their contents had been worked on during the research phase and writing continued from that as well as the lessons learned during development. While focus was kept on one chapter at a time the outline of the entire thesis was still worked on all throughout the writing process. Research complemented the actual writing work all throughout this phase.

The final month of the project was spent finishing and polishing the thesis report based on feedback, preparing and performing presentations and working to complete all other requirements associated with the final phase of a thesis project. Opposition could unfortunately not be arranged before the end of the five months and as such spilled into the subsequent semester. This was unfortunate but ultimately acceptable.

All throughout the thesis project a daily diary was kept and weekly status reports to the supervisors were made. This, along with the rigorous use and iterative development of the project plan containing the schedule and decisions around method and scope, was crucial in keeping the project on track and on time.

1.4. Related Work

Jeppsson and Olsson did a Master's thesis on the usage of blockchains for tracking goods during transportation [1]. Their thesis focuses on the impact to the transport chain actors, offering an excellent complement to this thesis which focuses more on the security and technical aspects of such a blockchain system. Their thesis is highly recommended reading.

In another related Master's thesis Jansson and Petersen developed a framework for evaluating blockchain as a supply chain traceability system [2]. While their framework is not used in this thesis important topics brought up by them are addressed such as the difference between documental and historical accuracy.

Yli-Huumo et al mapped out existing areas of blockchain research up until 2015 [3]. They concluded that 80% of research focused on Bitcoin specifically instead of the general field of blockchain technology. They also

summarize many areas of concern for blockchain technology such as throughput, latency, resource use, attack vectors and privacy.

Important aspects of performance and scaling in blockchain systems is addressed by Scherer [4]. Many of his results, as supported by other sources, are important for highlighting the differences between permissioned and permissionless blockchain technology in this thesis.

There is quite a bit of research and development going into blockchain applicability for supply chains as the area is considered “promising” [5]. Most of that work is primarily focused on moving the physical paperwork surrounding transportation to a digital system based on blockchain technology, and to then be able to trace a product through the entire supply chain with that blockchain data. As examples, [1] and [2] are already mentioned. See also [5, p. 23], [5] itself, and from industry see for example [6] and [7]. It is notable that much of this work is less than a year old, speaking to the emergent nature of blockchain technology.

1.5. Thesis Outline

This thesis is divided into three main parts. 0 deals with cryptography, blockchain technology and explains the current TDL security solution. Part II deals with the TDL Blockchain System and its proof-of-concept implementation. Part III contains a discussion of blockchain technology and the TDL Blockchain System as well as potential future work and the thesis summary.

Part I. Theory

2. Cryptography

An understanding of the basic concepts of modern cryptography is required understanding for many important aspects of blockchain technology. The field of modern cryptography is highly mathematics-based but those specifics are not required understanding for blockchain technology.

To see roughly what section of cryptography aids understanding of what section of other cryptography and blockchain technology see Fig. 2.

It helps to know this	**To understand the following**
Hashing	Proof-of-work Signing
Symmetric cryptography	Asymmetric cryptography Sharing secret keys
Asymmetric cryptography	Signing Digital identities Permissioned blockchains
Signing	Proving origin/integrity of data Permissioned blockchains
Digital identities	Signing Permissioned blockchains

Fig. 2. The cryptography areas on the left help in understand the areas of cryptography and blockchain technology on the right.

Because of the basic nature of this section on cryptography the information is largely sourced from the lecture notes for EIT060 Computer Security held in 2017 at Lund University, Faculty of Engineering [8].

In cryptographic scenarios, it is common to use “Alice” and “Bob” to denote trusted parties wishing to communicate in some way, and “Eve” as someone wishing to eavesdrop or otherwise compromise the scenario.

2.1. Kerckhoffs' principle

Cryptographic algorithms use keys to protect data, and Kerckhoffs' principle states that a cryptosystem should remain secure even if everything about it, excluding its key, is public knowledge [9].

Related to Kerckhoffs' principle is the general principle of only using cryptographic primitives that have been thoroughly tested using extensive research, also known as "don't roll your own crypto" [10] [11].

2.2. Hashing

Hash functions, $\mathrm{hash}(m) = h$, have two defining properties. Firstly, they should map a bit message of arbitrary length m to a fixed-length output h, see Fig. 3. Secondly, they should be computationally light.

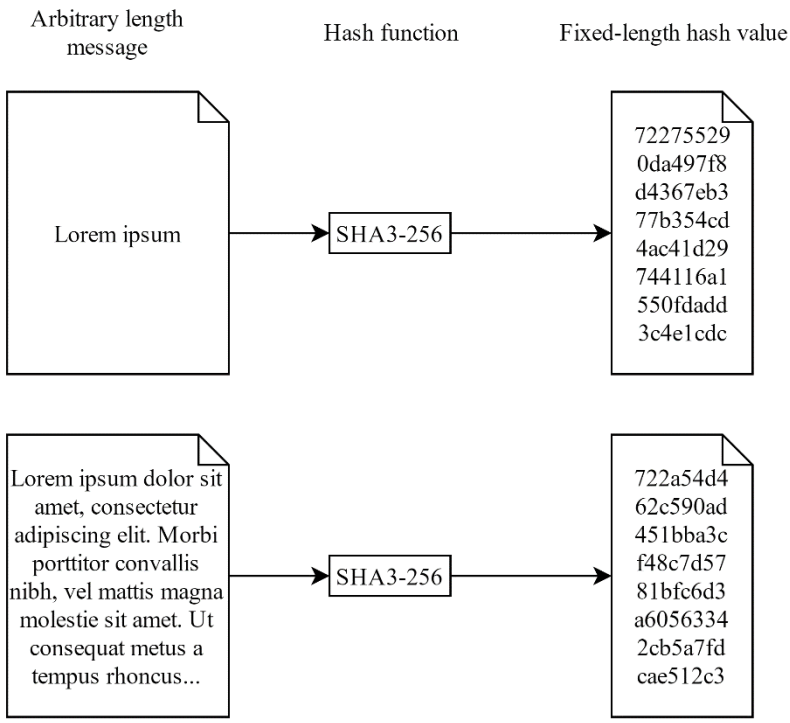


Fig. 3. Messages of varying length are run through the cryptographic hash function SHA3-256 generating hash values of the fixed length 256 bits.

To make hash functions suitable for use in cryptography, three additional properties are needed.

- Pre-image resistance: given a hash value h, it is infeasible to find m such that $\text{hash}(m) = h$.
- Second pre-image resistance: given a message m, it is infeasible to find m' such that $\text{hash}(m) = \text{hash}(m')$.
- Collision resistance: it is infeasible to find m and m' such that $\text{hash}(m) = \text{hash}(m')$.

Cryptographic hash functions provide small representations of potentially much larger data. This is very valuable when signing data as without representing the data with its hash value the signature would have to be just as large as the data itself. Section 2.5 deals with signing data using asymmetric cryptography.

2.3. Symmetric cryptography

Cryptography has historically only consisted of what we today call symmetric cryptography. It is used to send encrypted messages back and forth between trusted parties sharing a common secret key. This key is, as its name suggests, kept secret from everyone else.

Alice and Bob trust each other and share a secret key k. Alice encrypts a plaintext message m using the symmetric encryption function $\text{encrypt}(m, k)$ and sends the resulting ciphertext c to Bob over some communication channel. After receiving the ciphertext Bob uses the corresponding decryption function $\text{decrypt}(c, k)$ providing it the same secret key that Alice encrypted the message with. The result, the original message m, can now be read by Bob. This is illustrated in Fig. 4.

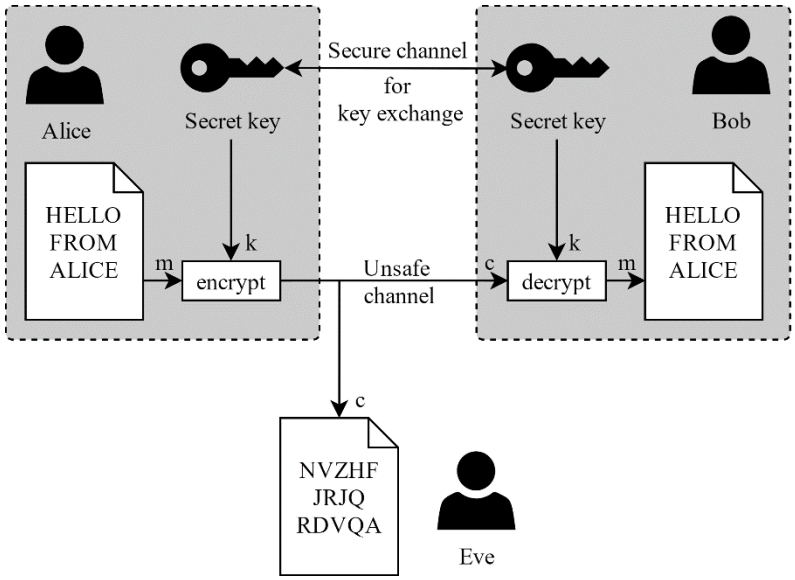


Fig. 4. Alice encrypts the message “HELLO FROM ALICE” using a secret key and sends it to Bob who uses that same key to decrypt the ciphertext “NVZHF JRJQ RDVQA” into the original message. Eve, who only has access to the ciphertext and not the key, is unable to access the message.

The main advantage of symmetric encryption is speed and resource use. It is, in general, computationally lighter to encrypt and decrypt data with symmetric encryption than it is to use asymmetric cryptography, described in Section 2.4.

2.3.1. Exchanging secret keys

A problem is the need to share the secret key outside of the cryptosystem, which creates an attack vector. When describing a symmetric encryption system, it is generally assumed that the key is already shared between the trusted parties, leading to the alternate name “pre-shared key”. When describing symmetric encryption schemes, keys are assumed to be distributed in a safe manner outside the cryptosystem.

It is common to use the slower asymmetric cryptography to exchange a secret key and then switch to symmetric cryptography for the rest of the session.

2.4. Asymmetric cryptography

Instead of being based on sharing a common secret key, asymmetric cryptography is based on the concept of having a pair of mathematically linked keys. Every participant holds one of these key pairs, keeping their "private key" secret while actively sharing their "public key".

One of the most famous asymmetric encryption schemes is RSA, named after its three creators Rivest, Shamir, and Adleman. RSA uses a form of trapdoor function, taking advantage of the computational difficulty of finding the factors of large numbers. It is easy to multiply two large prime numbers to generate a very large number, but to find the two prime factors only knowing the large number is very difficult.

Asymmetric cryptography can, in addition to its use in encryption, also be used to create digital signatures that ties data to a specific source. Signatures can also be used to validate that the data has not been changed since it was signed.

2.4.1. Key pair

The public and private keys are cryptographically linked in such a way that if you encrypt a message with the public key only someone who holds the private key can decrypt it. It is also important that it is infeasible to find the private key if you're given the public key. These properties make it easy to verify that someone has access to a certain private key, you simply need to encrypt some random data with a public key and if they can decrypt it then you know that they have the associated private key.

Note that the names "public" and "private" refer to how the keys are used.

2.4.2. Asymmetric encryption and decryption

A message encrypted with the public key can only be decrypted with the private key.

Alice has generated her key pair consisting of her public key pub_A and private key $priv_A$. Bob has, in turn, generated his own key pair pub_B and $priv_B$. Alice and Bob have both shared their public keys pub_A and pub_B with one another in such a way that they both know that the respective key belongs to the other.

Alice wants to send a plaintext message m to Bob so she uses Bob's public key pub_B to encrypt the message into ciphertext c using $c = \text{encrypt}(m, pub_B)$. She then sends c to Bob who decrypts the message using $m = \text{decrypt}(c, priv_B)$. This is illustrated in Fig. 5.

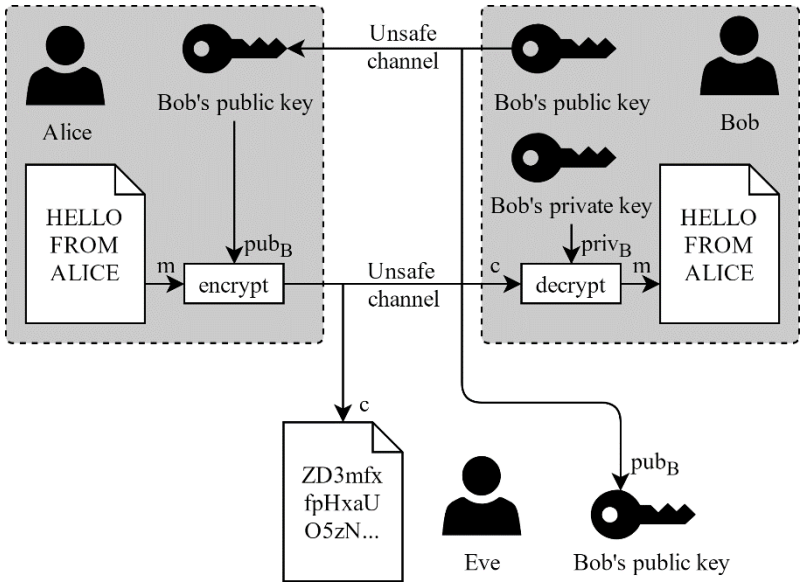


Fig. 5. Alice encrypts a message “HELLO FROM ALICE” using Bob’s public key. She sends the resulting ciphertext to Bob who decrypts it using his private key. Eve who knows both the ciphertext and Bob’s public key is unable to decrypt the message as she does not know Bob’s private key.

It should be noted that the message is in practice not encrypted/decrypted immediately, some sort of pre- and post-processing should be implemented to prevent the identical messages from being encrypted to the same ciphertext, a fact that can otherwise be used by Eve to gain information about the messages sent [12].

2.5. Signing

To prove the origin of a message it is common to send a digital signature along with the message itself.

The signing process begins with Alice in possession of her own key pair, pub_A and $priv_A$, and her having shared her public key in such a way that everyone knows that she is really the owner of it. Alice wants to send a message m to Bob so that Bob can verify that it came from her.

Alice produces a hash value $h = \text{hash}(m)$ of the message m and generates a signature $s = \text{sign}(h, priv_A)$ by signing h using her private key $priv_A$.

Alice now sends not only the message m to Bob but also the signature s. Bob now verifies the signature, verify(m, s, pub_A), using Alice's public key. He can then conclude whether the message he received was sent by Alice or not. This is illustrated in Fig. 6.

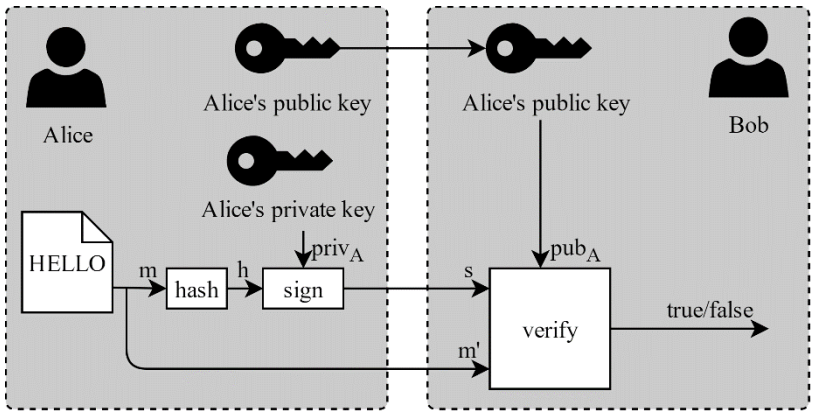


Fig. 6. Alice sends the message "HELLO" to Bob. She also encrypts the hash of that message using her private key. Bob can then verify that the message he receives is the one Alice signed by decrypting the signature with Alice's public key and comparing the hash value from it with his independently computed hash value of the message he received.

Note that signing does not provide encryption of the message sent, that must be done separately. See [12] to learn more about the differences between signing and encryption with RSA.

A final note on replay attacks, in which the same data and signature is provided multiple times. Consider a situation where you send status reports every five minutes. If you have data and signature for when 'everything is fine' you could send that same data and signature multiple times to avoid exposing that everything is not fine. To prevent you from doing that including a "number used once", nonce, with the data would mean that the data and signature would not be accepted other than the first time it is sent.

2.6. Tying public keys to entities

Being able to verify the connection between a public key and a private key is not same as verifying who the physical (or legal) entity behind that key pair is. A separate trusted third-party certificate authority is often used to attest that a specific key pair belongs to a specific entity. The certificate authority, CA, verifies that Alice is the entity associated with Alice's public

key and creates a certificate asserting that to be true. The CA signs this certificate using its own private key. This is illustrated in Fig. 7.

Instead of independently doing so, Alice, Bob and any other entities networking with each other place their trust in the certificate authority to verify that pub_A belongs to Alice, pub_B belongs to Bob, and so on. The certificate provides a link between a physical identity and a digital identity (public key).

For Alice to prove her physical identity in a digital system she can provide the certificate along with proof that she is in possession of the private key $priv_A$.

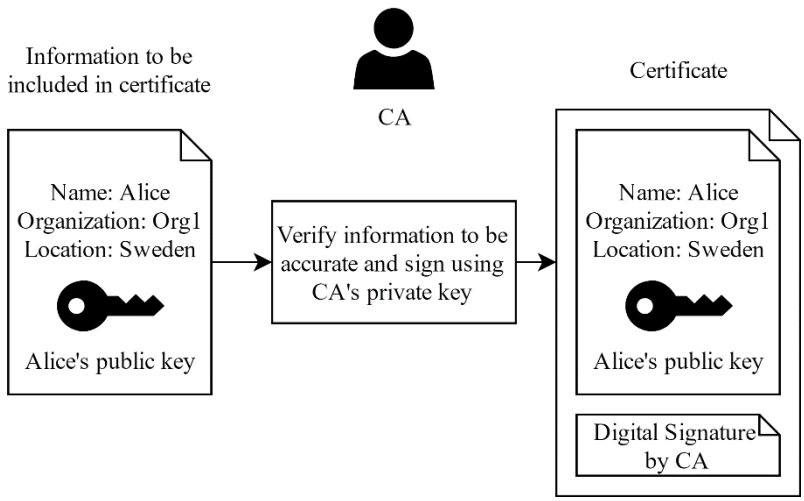


Fig. 7. A certificate authority verifies the link between Alice, the physical person, and Alice's public key. The CA then creates a certificate stating that fact, signing it using the CA's own private key.

The certificate authority does not have to independently verify the identity of and create certificates for every single entity. Instead the certificate authority can, for example, create a certificate for an entire organization and then that organization can in turn create their own certificates for individuals within the organization. The way these certificates are linked together is illustrated in Fig. 8. This chain of certificates can then be accepted as identity verification.

When verifying those individuals' certificates, the verifier checks the chain of certificates all the way back to the original certificate authority, see Fig. 8. As a CA is considered the root of trust they sign their own certificate, thus forming the final link of the chain.

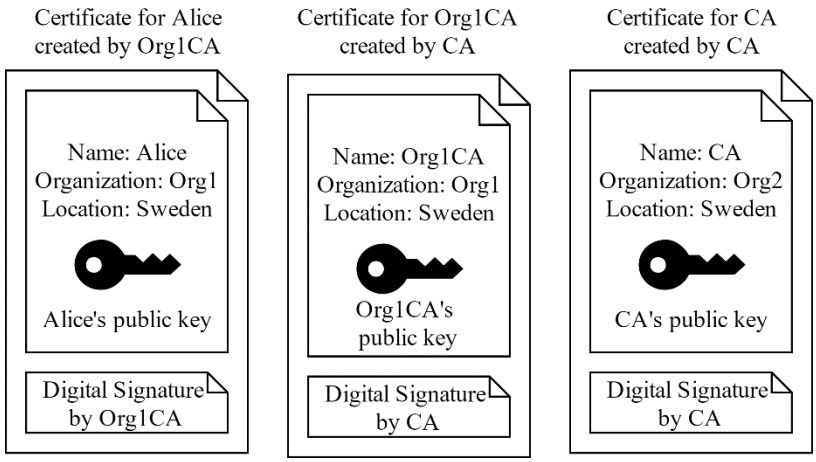


Fig. 8. Alice has a digital certificate of her physical identity and public key created by Org1CA. A separate certificate tying together Org1CA's public key and physical identity is signed by CA. Finally, CA has a self-signed certificate.

The importance of the role of certificate authorities as the root of trust in public key infrastructures, PKI, cannot be understated. Much of web security, such as HTTPS and TLS, depend on CAs. Additionally, many Swedes are likely familiar with BankID which is a digital identification service used by individuals to identify themselves to banks and authorities. This service is also dependent on certificate authorities.

3. Blockchain technology

A centralized database is hosted by a single entity and trust is put in that entity to maintain the security and integrity of that database. A distributed database provides redundancy, but full trust between all participants is still needed. Where blockchain technology comes into play is in areas where full trust of that kind is considered unachievable or undesirable.

The basis of blockchain technology comes from two things. First is the data structure consisting of a sequence of data blocks tied cryptographically to one another in chronological order. The second is consensus-driven blockchain network distribution, with the data being replicated across mutually distrustful nodes.

3.1. Concept

Blockchain technology promises network-distributed, decentralized and immutable data storage and transaction conduction. The information contained within the blocks make up a database where adding or changing information in that database comes in the form of appending new blocks to the blockchain. A block consists of a header, containing mostly metadata, and transactions that hold the actual blockchain information making up the database.

Because of the append-only nature of blockchain technology any previous state of the database can be extracted by looking at the blocks up to a certain point. While a current value in the database can change, a history of all database operations is intrinsic to the append-only design of blockchains.

Many very differing implementations of the blockchain concept exist, with their common factor being the sorting of data into an append-only list of blocks, chronologically and cryptographically linked to one another in linear sequence. Entities participating in a blockchain network, see Fig. 9, are all collectively responsible for maintaining the blockchain.

Some form of consensus building methodology is needed to make sure that the blockchains at the network nodes are identical to one another, making the figurative collective blockchain stable. All good consensus methods are tamper resistant, meaning many separate malicious entities would have to collaborate for tampering to succeed.

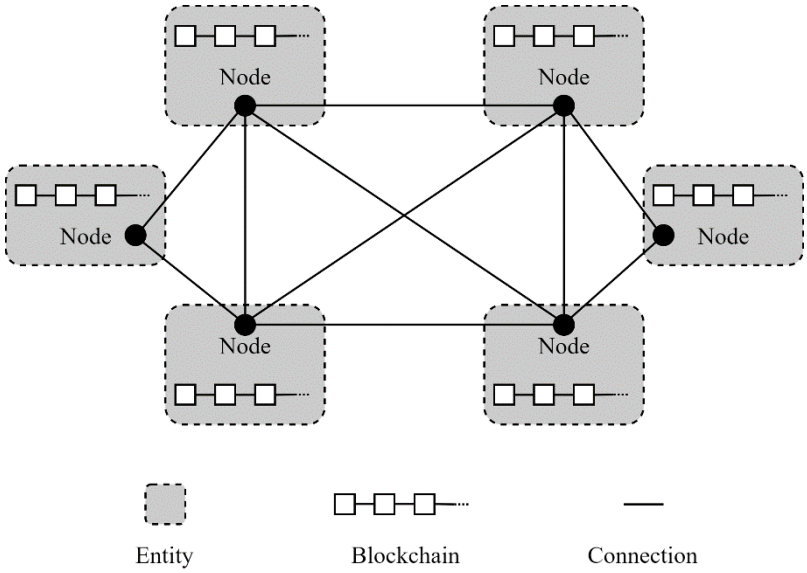


Fig. 9. A simplified blockchain network layout. Each entity controls a network node. These communicate with one another in some way, with all of them holding their own blockchain. Any one network node's blockchain should match that of the rest of the correct nodes, making a stable collective blockchain.

Every block in the chain, except for the first "genesis block", contains a cryptographic hash of the preceding block in the chain, illustrated in Fig. 10 [3] [13] [14]. Depending on the specific blockchain system, certain demands can be put on these hashes. For example, in proof-of-work as implemented by Bitcoin the hash must contain a certain number of leading zeroes [15].

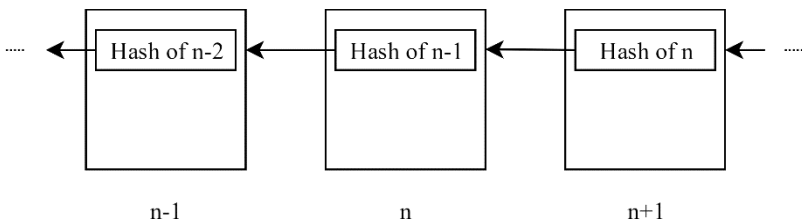


Fig. 10. The blockchain is examined at height n. The block at height $n + 1$ contains the hash of, or references, the previous block at height n. Similarly, the block at height n references the block at height $n - 1$. Note that no block references a subsequent block, all references are to preceding blocks. This means that while the block at n only

references the block at $n - 1$, any number of blocks could contain a reference to n.

These cryptographic hashes are what tie blocks together and make the blockchain immutable. Depending on what specific blockchain technology is used this is a statement with some definite asterisks. The consensus finality property says that if a node appends a block to its blockchain no other node will append any other block before that one to their blockchain [14]. Without consensus finality, the blocks that make up the blockchain can change even though the blocks themselves are unchanged. The only difference is what blocks are considered part of the blockchain. For example, in the proof-of-work system Bitcoin the series of blocks that are collectively accepted as constituting the blockchain can change if a longer sequence of blocks is found, even if it excludes blocks currently seen as part of the blockchain [14] [15]. This is known as the "longest chain" rule, illustrated in Fig. 11. Other scenarios exist where blockchains become similarly split and the question of which branch counts as "the one" is raised. If a blockchain has consensus finality, then once a block has been added to the blockchain it cannot be removed or replaced (by the consensus process).

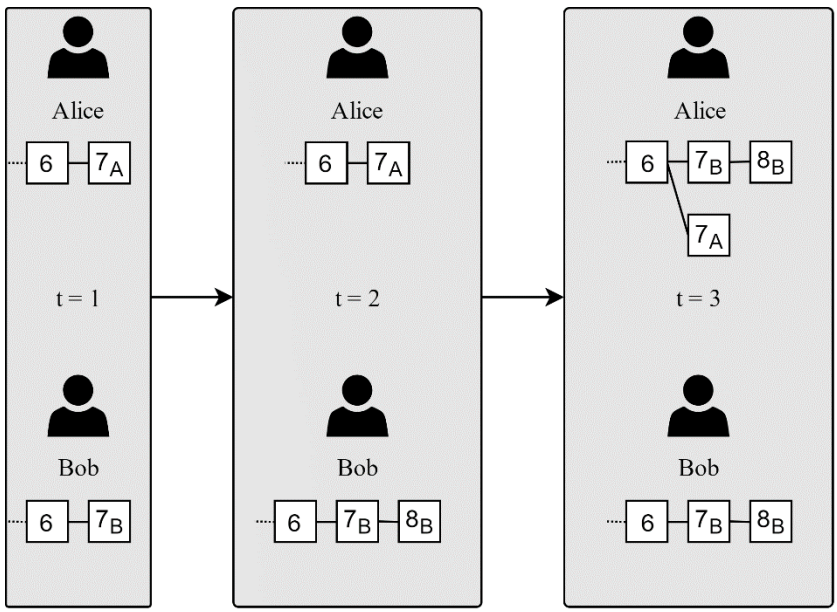


Fig. 11. Alice and Bob are participants in a blockchain network that uses the longest chain rule. At $t = 1$ they both have matching blockchains up to the last block height, 7. There they separately created new valid

blocks at around the same time meaning either of their blockchains could be considered "the" blockchain. This constitutes a "fork" in the collective blockchain. The different blocks are considered different "branches". At $t = 2$ Bob happens to be the first of the two to mine a new height 8 block making his branch longer. Alice, who becomes aware of Bob's new block at $t = 3$, will now use the blocks from Bob as per the longest chain rule, abandoning the block 7_A that she previously considered part of the blockchain.

3.2. Consensus

The term consensus in the context of blockchains is the process of making sure all "correct", or alternatively "honest", nodes have a shared and consistent blockchain [16]. This requires two abstracted elements [17]. The first is a state machine containing some form of logic, illustrated in Fig. 12. A state machine consists of state variables and commands [18]. The state is encoded using the state variables, and the commands are the state transitions from one valid state to another. In the case of blockchain technology, the state is the blockchain itself and the state changes are the additions of new blocks to that blockchain.

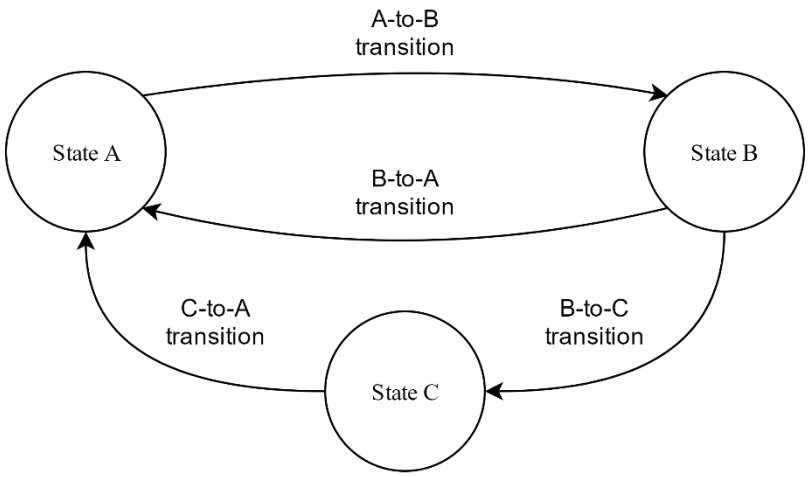


Fig. 12. A simple state transition diagram. The diagram contains three states, A, B and C. The state transition arrows indicate possible changes of the current state. Using the terminology of state variables and commands there is one variable that has the possible values of "A", "B" and "C". Commands can change that variable value from A to B, B to A, B to C and C to A.

The second element is a replication management, or consensus, protocol that coordinates node interactions with the state machine. Consensus needs to be achieved and then maintained. First some common state must be shared between all state machines. Second, an agreement must be met on any new state change to make sure that all nodes reach the same new state. Important for the decentralized nature of blockchain technology is that this is done without trusting any one entity fully.

This description of consensus is abstract for a reason. The area of blockchain technology is sprawling and very disparate forms of consensus building exist. This makes it hard to describe them collectively without referring to some abstracted common ground.

It is also a good idea to assume that not all nodes behave correctly but that they can also be "Byzantine". The term, explained in Section 3.2.1, in short means that a node can fail or misbehave in a way that threatens the integrity of a network-system not protected against Byzantine behavior.

A fundamental threat is if some sufficiently large number of malign blockchain network participants collaborate to subvert the integrity of the blockchain system [13] [14]. Participants holding some large enough network power would in extension obtain power over the blockchain itself. Power that would, depending on the specifics of the system, include the ability to replace parts of the blockchain and to censor any new transactions for any reason [13] [19]. Having some large enough majority of non-Byzantine nodes is critical to both proof-of-work and Byzantine Fault Tolerant consensus [14].

A single entity could potentially connect a disproportionate number of nodes to the blockchain network. Section 3.2.2 contains information on attacks where multiple identities are used by the same entity.

The two main types of blockchains, permissionless and permissioned, are explored in Sections 3.5 and 3.6. Their specific consensus methods are explained in Sections 3.5.1, 3.5.2, 3.6.2 and 3.6.3.

3.2.1. Byzantine nodes

Using Byzantine as a shorthand for misbehaving nodes is terminology originating in the Byzantine Generals Problem. The problem deals with agreeing on some common data in a distrustful environment.

The problem is summarized in [20] as "a group of generals of the Byzantine army camped with their troops around an enemy city. Communicating only by messenger, the generals must agree upon a common battle plan. However, one or more of them may be traitors who will try to

confuse the others. The problem is to find an algorithm to ensure that the loyal generals will reach agreement."

Protection against byzantine nodes is always in relationship to the total number of nodes in the network. See Sections 3.5 and 3.6.3 for specific thresholds for tolerance of Byzantine nodes.

3.2.2. Sybil attacks

A single entity has the potential to present several different and seemingly unrelated identities to gain some unproportionate amount of power in a peer-to-peer system, something that is known as a Sybil attack [21].

Because the resilience of blockchain technology lies in the data structure and distribution network in combination, one being subjectively useless without the other, a blockchain system using distribution susceptible to Sybil attacks would eliminate any trust for that blockchain system.

The distribution of blockchain network power could, using a Sybil attack, be shifted so a single entity gains control over the collective blockchain. Two current solutions to this problem exist, one being that network participation is limited to verified identities, and the other that blockchain network power is proportional to the resources being contributed to the consensus process, see Section 3.6 and 3.5.

3.3. Chaincode and smart contracts

Some form of logic is needed to decide what constitutes a transaction on the blockchain. Many cryptocurrency blockchain systems have this logic built-in as it only needs to support a small set of features such as transferring cryptocurrency tokens from one owner to another. To support more general use cases, the concept of "chaincode" running on the blockchain network was introduced.

The related term "smart contract" comes with some baggage, see [22], and to avoid the implications that come with that name the Hyperledger Fabric term chaincode will be used unless specifically referring to a smart contract.

Chaincode is described in [23] as "software, running on a ledger [blockchain], to encode assets and the transaction instructions (business logic) for modifying the assets". This is a pretty concise description and for a programmer chaincode doesn't look all that different from any other software implementing some form of business logic.

With the risk of introducing confusion by making two similar analogies chaincode can also be described using state machines. The assets of chaincode constitute the state machine variables and the changes made to those assets are the state transitions.

In a permissionless blockchain system executing chaincode is associated with payment in the form of cryptocurrency but no such payment is required in a permissioned system [5].

3.4. State of blockchain security

There has been an explosion in research of blockchain technology in the past few years although a large majority of that research has been centered on Bitcoin as opposed to the technology in general [3].

It is argued that while blockchain technology should be developed and evaluated like a cryptographic system, many of the blockchain systems currently in existence have been created in ignorance of such development processes [17].

Neha Narula, who is one of the discoverers of a rather obvious but very severe security vulnerability in the blockchain system IOTA, argues the following [11]:

"Though the [blockchain] technology is exciting, the due diligence required to make sound investments in the technology isn't keeping up with the pace of the hype. Aside from the financial risk, I don't think developers and investors are thoroughly evaluating these systems technically, either. Many investors are relying on signaling—if enough well-known institutions like universities or large companies sign on as investors or advisors, it indicates approval of the project and its software. The problem is that some of these technologies have serious issues, and the large companies and well-known individuals either aren't doing due diligence and investing the resources and time needed to evaluate the projects with which they are partnering, or aren't sharing their findings with everyone else. The cryptocurrency space still doesn't have a good way to assess these projects."

The security vulnerability in question was a result of IOTA developers breaking the mantra against rolling one's own crypto, see Section 2.1, as they had created their own hash function that could not withstand differential cryptanalysis [24]. Narula puts her finger on a subjectively large issue with blockchain technology today. There is a lot of work being done in the field but little of that work is dedicated to making sure the blockchain systems are secure and trustworthy.

Not only attacks to the blockchain systems themselves need to be considered. Attacks on chaincode could lead to results just as devastating as if the target had been the blockchain system itself.

As part of a larger survey of attacks on smart contracts for the blockchain system Ethereum the somewhat infamous “DAO attack” is investigated in [25]. The DAO was a form of decentralized investment system where investors of the DAO could vote to choose where their collective investments should go. They raised 10.7 million in Ethereum cryptocurrency Ether, valued at USD 120 million [26]. A large portion of that investment was later stolen using vulnerabilities present in the DAO smart contract [25]. The attack led to a schism in the Ethereum community as a hard fork was made to reverse the transfers that constituted the theft [27]. Ethereum Classic, a version of Ethereum “free from censorship, fraud or third party [sic] interference”, was created as a reaction to the perceived wrongdoings of this hard-fork solution to the DAO attack [28].

Beyond the implications to chaincode security considerations, this sequence of events is a great illustration that the immutability associated with blockchain technology is not some absolute property but a result of consensus among network participants. This is not limited to just the participating network nodes but extends to the actual people in charge of those nodes.

3.5. Permissionless blockchains

The primary identifier of a permissionless blockchain system is its open participation model where anyone can join or leave the blockchain network at any time [14]. The largest problem associated with such anonymous networks, Sybil attacks, is solved by linking participation to real-world resource use [14]. Bitcoin and Ethereum are the two prominent systems used to exemplify permissionless blockchain systems in this thesis.

An adversary hoping to gain blockchain network control must control most of the total CPU-cycles being spent “mining” blocks for the blockchain to be successful, something known as a “51%” or “majority” attack [19]. This means that the more decentralized the network the less likely a majority attack is to succeed. It was however shown in 2014 that it is possible to achieve a successful “Selfish Mining strategy” attack on the Bitcoin system with a 25% minority [13].

Because the reliance of a permissionless system is based on the network being decentralized the organization of “miners” (nodes) into “mining pools” for sharing block creation awards effectively increases centralization and thus decreases the resilience of the system [13] [14].

As all blockchain data is shared to the network and network participation is completely open any data stored on a permissionless blockchain should be considered public. See [5, p. 25] for a short discussion on blockchain data privacy. The authors of [5] advocate storing sensitive data "off-chain" with only the hash "on-chain". This allows for verification but not direct protection as the off-chain stored data is most likely more vulnerable to, for example, deletion. Finding what hash is linked to what data could also be a source of additional system complexity.

3.5.1. Proof-of-work

Pseudonym Satoshi Nakamoto introduced proof-of-work consensus with the original Bitcoin whitepaper [15]. The generation of blocks depends on solving a hard cryptographical puzzle where any solution is easy to verify [29].

As previously mentioned each block in a blockchain contains the hash of the previous block. The specific data included in that hash varies with the specific protocol used but in general a summary of the block data along with some block metadata is used. Each block's hash must match a certain pattern, in Bitcoin this pattern is a certain amount of leading zero bits [15]. Finding a match to the pattern is the cryptographical puzzle.

As hashing the same data always leads to the same hash value a nonce is included as part of the block. Using different nonce values changes the hash of the block. This is illustrated in Fig. 13.The pattern rule is enforced by the network nodes as they only accept blocks generated by other nodes if, in addition to the other verifications, the hash of that block matches the pattern.

Attempt	Block data	Nonce value	Hash value	Matches pattern?
1478998384	3b96bb7e…	869292030	43862b5ec7…	No
1478998385	3b96bb7e…	169950584	ee084c2518…	No
1478998386	3b96bb7e…	208296255	0000000027…	Yes

Fig. 13. A few attempts at finding a hash value that matches the pattern required by proof-of-work, in this example case the pattern is 8 leading zero bytes. The block data remains the same for each attempt but the nonce value changes leading to the hash value of them both combined changing between each attempt. On the bottom attempt in the figure the hash value contains 8 leading zeroes and as such

matches the pattern. The nonce value and the other block data will now be propagated to the network as a complete block.

Every bitcoin block mined awards the miner with a bitcoin currency reward and this is what incentivizes entities to engage in the mining process [15]. As the number of bitcoins awarded for every block decreases, the long-term mining incentive is expected to be transaction fees [15]. Currently, Ethereum has a fixed reward but this may change in the future considering the planned move to proof-of-stake [30] [31].

Proof-of-work systems can scale to thousands of mining nodes and clients [4] [14]. This goes well with the goal of decentralization as supporting thousands of nodes also means that thousands of separate entities can participate in the blockchain network. The pooling of resources into mining pools does go against this decentralization principle.

Despite supporting thousands of network nodes and clients, performance and scalability of another kind becomes a large issue in proof-of-work systems [4].

The closer a transaction's block is to the end of the blockchain the less mining work is protecting it from being replaced using another longer sequence of blocks [15]. This lack of consensus finality, see Section 3.1 for a definition, has given rise to the recommendation that for Bitcoin one should wait for 6 blocks to be appended to the one with the transaction in it, taking roughly 60 minutes, for the transaction to be considered properly committed [14]. This results in transactions having a very high latency. This is in addition to the already low transaction bandwidth that with Bitcoin has a theoretical high of 7 transactions per second [14].

The mining process uses a large amount of CPU-cycles[3] and, in extension, electricity. This massive resource use is comparable to that of entire countries such as Denmark and Ireland [32] [33]. This can arguably be considered a massive waste.

The proof-of-work derivation used in Ethereum is called Ethash [34]. It mitigates some issues associated with Bitcoins proof-of-work consensus [4] [34]. Ultimately, the proof-of-work performance limitations, including high latencies and low bandwidth, are inherent to the method itself [14].

[3] CPU-cycles is used generally in this context. Currently much of Bitcoin mining is done with purpose-built mining hardware.

3.5.2. Proof-of-stake

In contrast to proof-of-work there is no “mining” in a proof-of-stake system. Block confirmation is based on in-system investment using the assumption that anyone with enough investment into the system would want to see it continue functioning. One example of a cryptocurrency that uses proof-of-stake consensus is BlackCoin [35].

Most proof-of-stake protocols currently proposed are susceptible to several attacks and proof-of-stake only accounts for less than 2% of cryptocurrency market capitalization [36]. Because of this assessing the usability of proof-of-stake in blockchain systems is, can, and will be the subject of entire theses.

Ethereum is likely to adopt a proof-of-stake system called Casper in their future Serenity milestone [37]. The basic idea behind Casper is that consensus will be reached by the participants (validators) placing bets on which block is going to become the next to be included in the blockchain. After enough validators have placed large enough bets on a single block it is considered finalized and every bet on any other block will be distributed as winnings to the validators who bet on the ‘winning’ block. The aim with Casper is to encourage betting on blocks that are likely to be accepted by the majority and to punish those who bet against the majority [38].

3.6. Permissioned blockchains

In contrast to permissionless blockchains where anyone can set up a node and connect to the blockchain network a permissioned blockchain network limits participation to known and approved entities. This is in practice achieved by a collaborating group of organizations. The main advantage compared to permissionless blockchains is increased transaction bandwidth and lower latency [4].

Permissioned blockchains will, in this thesis, be exemplified using mostly Hyperledger Fabric. The term “ledger” is often used in Hyperledger documentation and terminology to refer to the blockchain-based data storage but this thesis will continue to use the term “blockchain”.

3.6.1. The Hyperledger project

There are several different blockchain projects in varying levels of development, from incubation to production, all organized under the Hyperledger project from the Linux Foundation. Several high-profile

companies have chosen to gather their permissioned blockchain efforts under the Hyperledger banner [39] [40].

The Hyperledger project is envisioned as being decades long and the goal is to gather all viable permissioned blockchain solutions, allowing them to grow and to see which technologies emerge as the eventual "winners" [41]. Fabric and Sawtooth are the two projects furthest along in development, with a few production-level Fabric-based projects and some Sawtooth "late-stage pilots" [41].

Hyperledger Fabric is originally contributed by IBM and is the only project to have a 1.0 release at the time of writing [42]. It will be explained in more detail in Section 3.7.

Hyperledger Sawtooth uses a consensus algorithm based on "Proof of Elapsed Time", see Section 3.6.2. It uses a trusted execution technology to provide this proof and the network node responsible to provide the proof is chosen through a lottery system [17] [43].

Hyperledger Iroha has the goal of providing reusable components (code libraries, consensus, chaincode/smart contracts) to the other Hyperledger projects.

Hyperledger Burrow is contributed by Monax and it features an Ethereum Virtual Engine using a proof-of-stake Tendermint consensus engine. See [17] for more information about this specific consensus method.

Hyperledger Indy "provides tools, libraries, and reusable components for providing digital identities rooted on blockchains" [39].

Fabric, Sawtooth and Iroha are currently in active development while Burrow and Indy are in incubation status [44]. Considering the goals and development status of the Hyperledger projects, only Fabric and Sawtooth will be discussed further.

3.6.2. Proof of elapsed time

Proof of elapsed time, PoET, was created as a response to the knowledge of proof-of-work serving as a resource hungry distributed timestamping machine. PoET was developed to achieve the same result but with massively decreased resource use. The result in question is trust in that a certain amount of time has passed between each block. It leverages Intel Software Guard Extensions, SGX, to provide attestation that a certain amount of time has passed, with the assumption that the trusted execution technology cannot be subverted. This is potentially problematic as during the writing of this thesis security flaws subverting Intel's trusted computing technologies were

identified [45]. This is in addition to previous research on attacks against SGX [46] [47].

Because of proof-of-work similarities, PoET has the potential to be used in permissionless blockchains as well as permissioned ones [17]. Where proof-of-work uses CPU-cycles as the ticket of entry, PoET uses the availability of hardware supporting Intel SGX [17] [48]. As CPU-cycles are a far more ubiquitous resource this puts a lot of trust and some measure of centralization with Intel. The centralization comes from the fact that the distribution of Intel SGX compatible hardware will in practice form a barrier of entry not present in proof-of-work [17].

Finding a valid block hash in proof-of-work can be compared to simply waiting an amount of time and then propagating a block to the network with the proof of that time having elapsed, namely the block hash matching a certain pattern [17] [48]. Proof of elapsed time takes this literally and uses SGX to wait for some time and once that time has passed a node can use that SGX-generated proof as part of the block data and propagate that block to the rest of the network [17] [48]. Like with proof-of-work, PoET does not provide consensus finality as forks are created when new blocks are generated in close time proximity to one another [17].

PoET eliminates the massive resource use associated with proof-of-work at the cost of putting trust and power in the hands of a single hardware vendor [17] [48].

3.6.3. Byzantine fault tolerant consensus

Requiring nodes to have known identities allows the use of Byzantine fault-tolerant, BFT, consensus protocols [14]. These types of database replication protocols have a several decades long history and much research exists in the area [14] [49] [50]. BFT protocols have several important advantages over proof-of-work protocols, most importantly performance and resource use.

According to both [4] and [14] there is an apparent inherent tradeoff between transaction performance and node scalability. BFT protocols have great performance, tens of thousands of transactions per second [50] and low latency overhead of less than 100ms [51], but is only tested up to twenty nodes [14].

When one becomes aware of the $O(n^2)$ inter-node communication involved, see Fig. 14, the reason behind the limited node scaling of BFT protocols becomes apparent.

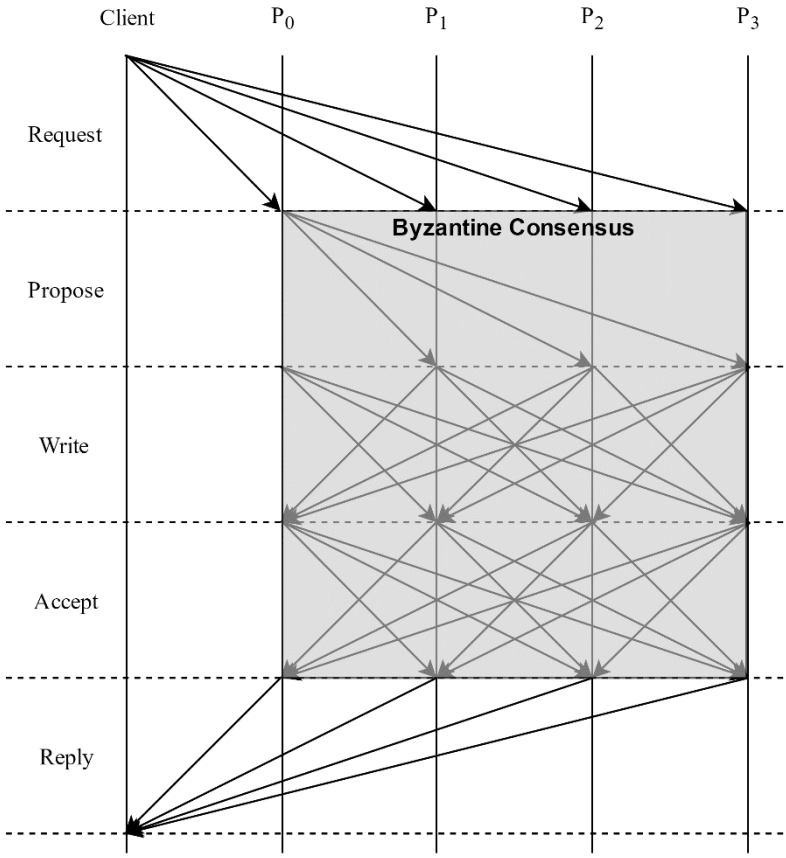


Fig. 14. Visualization of communication between nodes (P_0-P_3) performing BFT consensus. Adapted from [50, p. 148. Figure 2] and [52, p. 87. Figure 9].

Historically the protocol Practical Byzantine Fault Tolerance, PBFT, has been the earliest and almost only real-world usable BFT protocol [50]. More recently BFT-SMaRT has, with several improvements over PBFT, taken over as a more current implementation of Byzantine fault tolerance and it is used by several blockchain systems [17] [50].

BFT protocols protect against Byzantine nodes, but only if they do not make up more than a third of the total number of nodes [14].

3.7. Hyperledger Fabric

Hyperledger Fabric can currently be considered the flagship Hyperledger project as it is the longest-running and is the furthest along the development process.

3.7.1. Consensus and node responsibilities

Consensus in Fabric is a bit more involved than 'just' nodes propagating valid blocks to the rest of the network as with, for example, Bitcoin [53]. In earlier stages of development Fabric used native PBFT but architecture changes mean that the current 1.0 release does not include BFT consensus [17]. As adding BFT-SMaRT is part of current development, it should be noted that the current lack of protection against node subversion is a product of release scheduling and not inherent to Fabric [17]. This thesis will assess Fabric as if it uses BFT consensus.

The architecture changes for the 1.0 release were to separate some node responsibilities into different types of nodes [17]. The current division is into three types of nodes: clients, peers and orderers [54].

Before the node responsibilities are specified further, note that transactions in Hyperledger Fabric come in the form of "invoking" chaincode and then having the result of that invocation committed to the blockchain [54]. The network can contain several different chaincode stored on different subsets of peers [54].

A client node is the active driver of the transaction process, explained in Section 3.7.2. The client invokes a certain chaincode at some peer, requesting the peers' endorsement of the transaction [54]. The client then sends the endorsed transaction to the ordering service that performs a total-order broadcast to the network [54].

A peer holds their copy of the blockchain with new blocks being provided to them by the ordering service [54]. Peers can serve as "endorsers", where they endorse transactions before they can be committed. An endorser 'emulates' the execution of chaincode invocations and the result is either endorsed or rejected [54]. Chaincode has individual endorsement policies, requiring the endorsement from peers from a certain specified set of participating organizations for the network to consider a transaction involving that chaincode valid [53].

The orderers form the ordering service and can, if the system is configured to do it, keep their own copies of the blockchain [54]. The name

quite simply comes from the service performing the total-order broadcast of all transactions to the rest of the network [54].

Even invalid transactions are stored on the nodes' blockchains, although without affecting the actual blockchain database, to preserve the history of all transactions [54]. The blockchain database in Fabric is known as the state, modeled as a key-value store [54].

Once BFT-SMaRT is implemented, Fabric will be able to withstand up to a third of participating ordering nodes being subverted [17]. Remember that chaincode still needs to be properly designed, maintained and updated, read Section 3.3 for more. In addition, providing a chaincode endorsement policy that matches the stakes of the network is important. If an endorsement policy only requires a single organization to endorse transactions for that chaincode, nothing will prevent them from performing arbitrary changes to the key-value store associated with that chaincode.

3.7.2. Transaction process

Clients are the drivers of the transaction process. For the purposes of this explanation assume that the network is already up and running with all necessary steps taken to enable some type of example transaction. For an illustration of the transaction process, see Fig. 15.

The transaction process starts with a client sending a transaction proposal to the network peers, specifying what chaincode should be invoked, and any input parameters required by the specific invocation [55].

The peers run a simulation of the chaincode invocation and make sure that the result is valid as defined by the chaincode [55]. The result in question is a set of read/write instructions for the key-value state [55]. After performing all the necessary verifications, the peers send a proposal response back to the client, including their result and endorsement [55].

The client gathers the proposals into a transaction after verifying that the results are all the same [55]. This transaction is then sent to the ordering service, the orderer nodes, who collect it with any other transactions, checks for conflicting read/write sets, and forms them into a block [55]. The ordering service then propagates the block to the peers and they all commit the block to their copies of the blockchain. Not until the block is committed are the read/write sets of the chaincode invocations used to update the state of the key-value store.

The transaction process is concluded by the client receiving an event message broadcast by the network peers acknowledging the committal of the transaction.

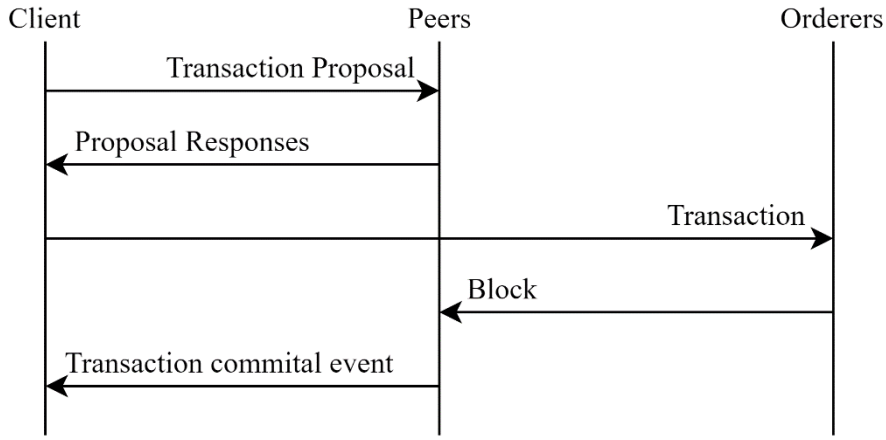


Fig. 15. A simple overview of the transaction process. A client submits a transaction proposal to the peers who respond with proposal responses. These responses are sent together as a transaction to the orderers who gather multiple transactions into a block. The block is sent to the peers who notify the client that its transaction has been committed to the blockchain.

Note that identities, results, signatures, endorsements etc. are checked by the involved peers and orderers at each step of the transaction process, even when not mentioned explicitly above [53] [55].

3.7.3. Participation and permissions

In the general case organizations are responsible for maintaining their own public key infrastructure, PKI. The participating organizations' public key infrastructures are connected to the larger Fabric network and serve as the providers and verifiers of digital identities and signatures used therein [53]. The connection between the Fabric network and the PKIs is done with Membership Service Providers, MSPs [56]. The connection between the MSPs and the actual blockchain is configuration blocks, including the genesis block [56]. Configuration blocks also contain other information needed to set up a network or channel [23].

A single network can contain one or more channels, a feature introduced with the 1.0 architecture change that aims to allow separation of more or less private data and/or logic from the rest of the network [53]. For this purpose, Fabric blockchains exist on a per-channel basis instead of a per-network basis [53].

The network and channels can be configured to allow for specific organizations to have different levels of permissions. Some organizations could, for example, only be permitted to run clients and some could be allowed to install/update chaincode [57] [56].

Unfortunately, a very crucial feature that has yet to be implemented is chaincode-level access control [58] [59]. This is a planned feature that would allow chaincode to react differently to a specific chaincode invocator (client) belonging to a specific organization or holding some specific title/role at that organization. The current lack of such a feature is problematic for many use cases, including the one in this thesis' proof of concept. Lu and Xu put it mildly when they say that "unauthorized users could accidentally trigger a permissionless function" [5].

3.8. Summary

Blockchain technology achieves consensus for, and immutably stores, some sort of data in an environment where entities in separate, mutually distrustful, trust domains want to achieve just that. This functionality is dependent on careful considerations and practices in developing and maintaining the blockchain system, network, the participating actor power balance, and the chaincode.

4. Transport Data Logger

The Transport Data Logger, TDL, was developed by Bosch Connected Devices and Solutions. Its functionality is to gather environmental sensor data for goods while they are being transported.

4.1. Tracking conditions during shipping

The TDL use case is illustrated in Fig. 16. The TDL device is mounted to a package that is going to be shipped and it is controlled by a smartphone companion application communicating over Bluetooth Low Energy (BLE) [60]. The TDL is first paired to the smartphone of the package sender who sets up the logging conditions [61]. This includes specifying the limits of what the shipped goods should be able to handle without a violation being recorded by the TDL [61]. Examples of violations are too high/low temperature, humidity, violent shocks and/or the package being moved using an incorrect orientation (tipping) [61]. At setup, a PIN-code is set that is used by the TDL device to protect changes to the setup configuration, resetting the TDL to factory settings, retrieval of the complete log data and to control if logging is turned on or not [61].

After setup, the TDL starts logging the orientation, temperature and humidity at the time interval specified during setup [61]. Any shocks are recorded at the time they happen. In this thesis, threshold value violations are all referred to as data points and collectively as the log data. The package with the TDL is then sent through the entire transport chain without any further communication with the outside world until it reaches the ultimate destination. Anyone can still pair with the TDL at any time and view the summary information that is available without use of the PIN [61].

The final recipient pairs the smartphone application to the TDL and can now choose to stop logging as the PIN is known to them [61]. After that they can download the complete set of log data and share it however they see fit.

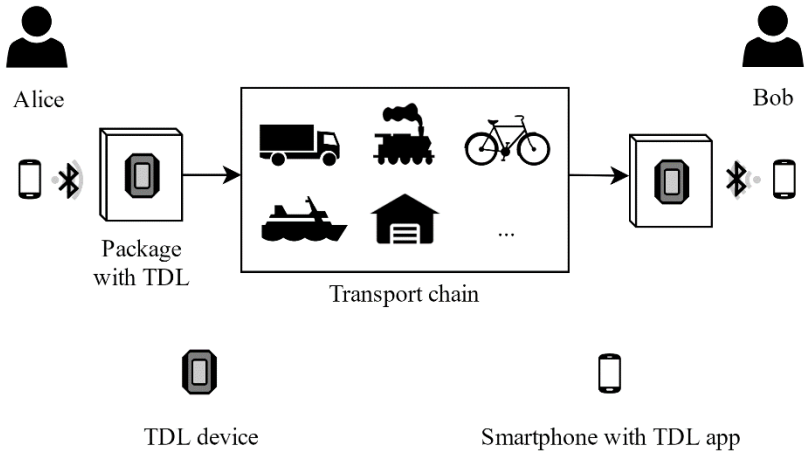


Fig. 16. Alice sets up the TDL parameters and starts the TDL logging functionality using the TDL app on her smartphone. The TDL, attached to its package, goes through the entire transport chain. Once Bob receives the package he can use the TDL app on his smartphone to stop the logging and download the data using the PIN supplied to him by Alice.

Violations are described as “clearly assignable to the stations throughout the entire supply chain” in the marketing materials for the TDL system [62]. This is based on manually comparing violation timestamps as recorded by the TDL device with any external documentation available in regards to the transportation of the TDL-enabled goods.

Many details regarding the TDL (and transportation in general) is left out of this thesis as data and computer security is the main area of interest.

4.2. Baseline version

As the TDL is in active development at Bosch at the time of writing a baseline was chosen for the thesis to prevent the problem of working towards a moving target. TDL model TDL110 with firmware version 1.0.15 and TDL Android app version 1.2.3 was chosen as those were the most recent stable external releases at the time the thesis project started in August/September 2017.

4.3. Security solution

Unless it was made clear previously the software security solution of the TDL is completely dependent on the use of a PIN to protect against

unauthorized changes. This protects against many trivial manipulations over Bluetooth such as disabling logging. It however does not protect against more advanced schemes such as opening the TDL case and doing direct manipulations to the flash storage of the device. Such an attack would be of much higher sophistication.

Other scenarios exist where the TDL can be successfully manipulated using much lower sophistication. The sensors themselves could be tricked or disabled as, for example, the humidity sensor can be obstructed [61]. Removing the battery would stop logging and possibly lead to data corruption [61]. If one wanted to it would be possible to remove and/or destroy the TDL. These sorts of attacks on the TDL are not considered further and it is, for the purposes of this thesis, generally assumed that the TDL records data that can be relied upon.

4.3.1. Assumptions

The sender and recipient are assumed to have complete trust between one another. The PIN code is set by the sender and is then shared with the recipient so that they can interact fully with the TDL. This PIN must be kept from all others as anyone with knowledge of the PIN in contact with the TDL device has complete power over the device's capabilities. This includes clearing the TDL data or resetting it to factory settings.

The details of the threat model used for designing and assessing the adequacy of the PIN protection can only be found in internal Bosch documentation.

4.3.2. PIN protection

PIN creation and sharing relies completely on the human factor. This opens several avenues of attack such as social engineering and attacks on the human to human PIN exchange. All privileged interactions are shielded with the same PIN code, set by the humans.

Humans are known to set and store PIN codes and other passwords in insecure ways [63]. It is not unlikely that the same person would set up several TDLs to use the same PIN code, as reuse of passwords is common [64]. The way the PIN is shared would likely be using some form of telecommunication such as e-mail, text messaging or phone calls. Leaving written copies of the PIN code is an obvious source of potential distribution of the PIN to actors of malicious intent, especially if these copies are unprotected and/or unencrypted. Speaking the PIN code out loud on the phone can quite literally

be eavesdropped by another human who happens to overhear the conversation.

In addition to the human element the connection between the device and smartphone app using Bluetooth must also be secure against eavesdropping of the PIN code as well as other attacks. The state of Bluetooth security was as examined 4 years ago in [65] generally bleak, and fundamental threats remain today [66]. This includes general inability to adequately protect Bluetooth network traffic, such as sending the PIN from the phone to the TDL device, from eavesdropping.

4.3.3. Data protection

While on the TDL device, the log data is stored using flash memory on the circuit board. This data storage contains no other security features other than the relative sophistication needed by an attacker wishing to directly manipulate the flash memory. The process for modifying NAND flash memory, as done in [67], is considered time-consuming and it also leaves physical traces. It should be considered a rather sophisticated attack.

Once the data is downloaded from the device no further protection or integrity check is provided. The downloaded data is in the form of an unsigned and unencrypted text file.

Being able to download a TDL's data means one also has the power to wipe the data from the TDL after having done so. Doing so makes it impossible to verify the source or accuracy of the data in any way.

It should be noted here that data signing is one of the features under current development at Bosch, but this thesis is, as mentioned previously, based on the baseline version.

Part II. Evaluation

5. TDL blockchain system

As per the problem statement of this thesis, this section will deal with how blockchain technology can be used to attribute TDL data points to specific transport chain actors and how the reliability of that data can be increased.

5.1. Reasons for using blockchain

Blockchain systems provide a framework for actors in different trust domains to reach consensus over some set of data.

This holds true in the transportation sector where several different companies cooperate to deliver goods from sender to recipient. While cooperating they still do not trust each other in that any one actor would not want to take blame for mistakes done by other actors. Reaching consensus on where goods were damaged or mishandled provides benefits to several actors.

Between transport chain actors the benefit is that any negative repercussions are aimed at the appropriate actor. The benefit for sender and recipient comes from knowing if the transported objects themselves survived the transport and being able to choose to not provide further business to misbehaving actors. If a transport chain actor has provided all employees with separate identities then shipping errors could also be traced within a company to a specific employee, if the blockchain system is built to support it.

Drawbacks come from having to develop, set up and maintain the system in co-operation with other companies and from a potentially increased time spent handing goods off between transport chain actors.

5.1.1. Tracking responsibility

As mentioned in Section 1.4, blockchain technology adoption for use in supply chains is promising. Much of that promise comes from providing a common system to reliably trace a package or product in a way that allows relevant parties to view the complete history of it [1] [68]. This common system would, if implemented correctly, provide all the benefits of blockchain technology such as immutability and trustable data storage and transaction conduction in a distrusting environment.

This work is directly related to the problem of attributing TDL data points to specific actors. If these handoffs of packages between actors are already recorded in a secure blockchain system attributing data points to specific

actors would simply entail tying the sensor data generated between each handoff to the responsible actor.

To avoid making assumptions about clock synchronicity and reliability, a specific handoff is tied to the data as it has been factually accrued up to that point in time. This is done by syncing what data is contained on-device as part of the handoff procedure.

5.1.2. Data reliability

A blockchain system can be used to make sure that some piece of data is stored unchangeably. That data must however already be "good", as simply putting it on the blockchain does not transfer any properties to it other than that of immutability. As [27] argues "the 'truth level' of on-chain information is only as good as barriers employed to […] ensure the quality of data being added is high" which means that blockchain technology is in some ways 'just' a container for already trustable data.

Tying a blockchain event or some data on the blockchain to the real world still requires the same PKI-based solutions common today, including certificates created by certificate authorities to attest to some entity's identity.

This means that for TDL data to be considered trustworthy on the blockchain it must already be considered trustworthy beyond the blockchain. This requires the TDL devices to carry their own digital identities attested by a trusted third party, likely Bosch. The TDL device would then have to use this digital identity to sign for any data leaving it, creating the necessary link between device and data needed to produce trust in that the data is genuinely from the device. This would have to include protections against re-use of signatures, see Section 2.5.

Because of the immutability of blockchain technology, manipulations to the TDL data would have to take place before the data is committed to the blockchain. Unless, of course, the blockchain network is completely taken over/down and the on-blockchain data is wiped from existence at all or arguably some majority of peers.

5.1.3. Assumptions

By automating some of the PIN handling procedures, see Section 5.3.5, there is no longer a need for complete trust between sender and recipient. The sender never becomes aware of the PIN code and only the recipient can get the PIN code.

The trust assumptions for the blockchain system itself is a bit more complex. The organizations do not have to trust a common third party[4], but instead need to place at least some measurement of trust in the public key infrastructures of the other organizations. They are trusted to protect their PKI, or looking at it from the opposite end the companies agree to some form of mutual distrust where blame should be placed at the organization tied to the PKI that was responsible for a TDL-enabled package at the point of any recorded violations. Of course, a single identity from that PKI performed the blockchain transaction but in the inter-organizational context that isn't particularly relevant, as the blame is placed organizationally.

Keeping a project open-source, at least between all relevant organizations, is a good way to produce trust in the system, including the chaincode responsible for all transactions. The trust would come from participating actors being able to verify, and contribute to, the system's operation. At the same time, this relies on the participating actors performing their due diligence and that can not be taken for granted to be true.

The issue of trusting software can of course be taken to the impractical extreme, where "you should write your own code, but also your own compiler, and run that on hardware which you designed and engraved yourself: you cannot trust the machine unless you started with a bucket of sand (for silicon, the main component of semiconductors)" [69]. See [70] for a good, but old, discussion on trust in software.

5.2. Choosing blockchain type and system

To choose what blockchain system should be used several important characteristics of blockchains in general had to be identified. The first step was to take a step back and determine whether blockchain is a relevant technology at all. This has already been addressed in Section 5.1, but for clarity the decision tree detailed in Section 5.2.2 will include whether to use blockchain or not.

[4] As mentioned in Section 5.1.2, the TDLs need digital identities. Bosch providing these would in effect make Bosch a trusted third party. Other schemes could be envisioned where these identities are instead provided with the involvement of several organizations in such a way that one of them need not be trusted fully. In practice, the trusted third-party route is probably simpler to implement.

As the choice between permissioned and permissionless includes many considerations it is difficult to isolate the choice to a single factor.

5.2.1. Important criteria

The main criteria used to choose between permissionless and permissioned blockchains is the nature of participation. Using the following reasoning, the decision was made to use a permissioned system.

A system required to track responsibility of some real-world object needs to connect the identities of who interacts with the blockchain and real-world entities. These connections can plausibly be made *in* a permissionless blockchain system by making chaincode use some form of PKI in its decisions. This would in effect just be implementing a heavy permissioned layer on top of the permissionless blockchain system. Instead, linking blockchain network identities to physical entities can be made *with* a permissioned system that is purpose-built for doing just that.

To choose between the prominent permissioned blockchain implementations Hyperledger Fabric and Hyperledger Sawtooth, the most important factors were their choice of consensus and assessed maturity.

Sawtooth uses proof of elapsed time, potentially introducing some of the performance drawbacks associated with permissionless blockchain technology. This is in addition to the reliance on a single hardware vendor and the unproven nature of the consensus methodology. Fabric uses BFT, a both well-explored and well-performing consensus method.

On the point of maturity, a subjective choice had to be made based on availability of example code, documentation, SDK etc. Fabric having a longer history, a 1.0 release, and some supposedly production-level implementations all weighed in its favor against Sawtooth.

5.2.2. Decision tree

The decision tree in Fig. 17 is based in part on similar decision trees from [27] and [71]. The purpose of it is to visualize the decision between traditional databases and blockchain technology and then the choice between a permissioned and permissionless blockchain system.

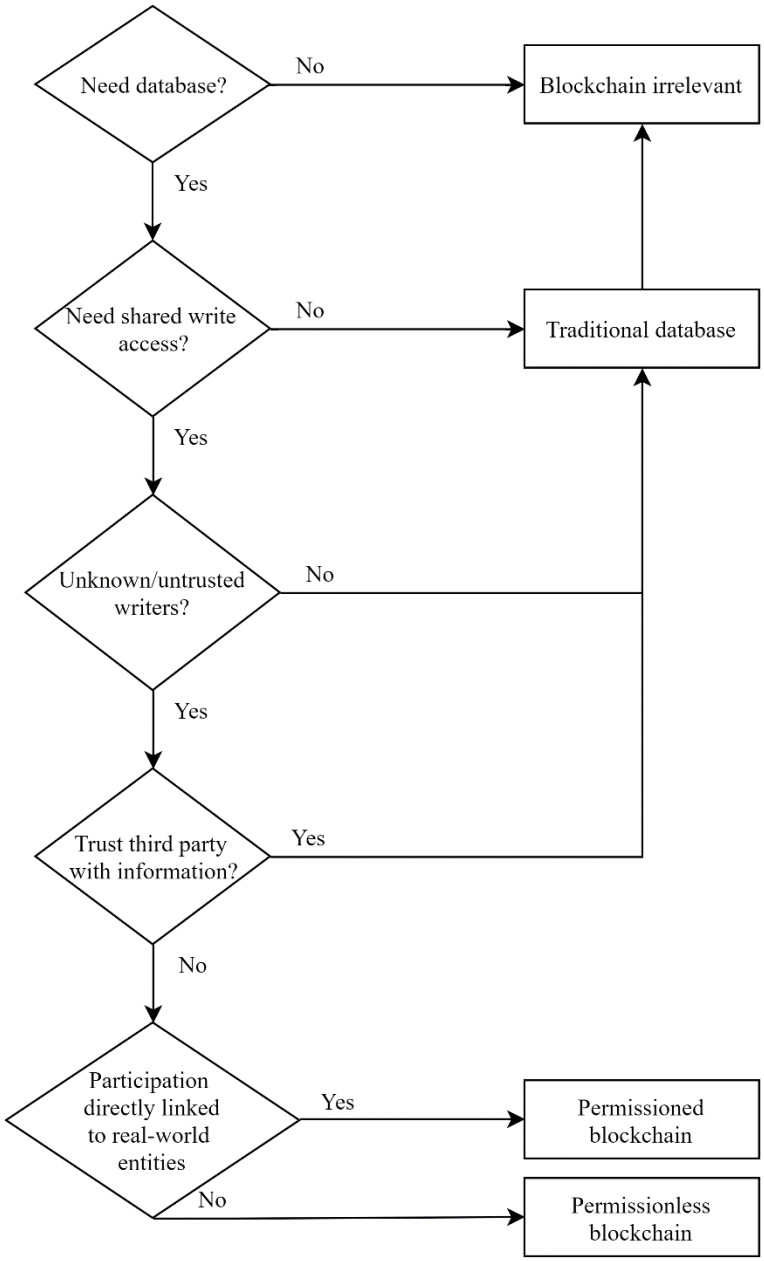


Fig. 17. Decision tree on what general type of blockchain system to use, permissioned or permissionless. To highlight the fact that blockchains are not necessarily applicable in all situations the decision tree also includes whether to use blockchain at all.

To choose between the Hyperledger projects Fabric and Sawtooth several factors were considered. As per the reasoning in Section 5.2.1 Fabric was chosen as the basis of the proof-of-concept TDL blockchain system.

5.3. Proof-of-concept implementation

As part of this thesis a TDL blockchain system using Hyperledger Fabric was developed. The premise is that all transport chain actors agree to use this common blockchain system to track all TDL-enabled packages. The proof of concept uses four mock organizations simply called Org1, Org2, Org3 and Org4. They each run network nodes on a common network and channel, and are all able to conduct TDL-related transactions using the developed chaincode.

Because of implementation scope and limitations, there are several aspects of the system that are not representable of a production-level system. Some of these are inherited from Fabric, some from the TDL system and some from a simple lack of time and/or scope to implement them. Major deviations are the handling of cryptographic materials, using a bridge-gap solution for the app, and the lack of chaincode access control and TDL data signing.

5.3.1. Versions used

The proof of concept is a product of its time and knowing something about the exact version numbers used is useful to ascertain what circumstances surrounded development.

The operating system used for development was Ubuntu-based OSD4. The TDL system version numbers can be seen in Section 4.2.

For Hyperledger Fabric, version 1.0.1 was used for the Java SDK and the Fabric images (ca, tools, couchdb, kafka, orderer, peer, javaenv, ccenv). The Fabric image “baseos” carries the version number 0.3.2 but is the most recent in relation to the others used.

The Fabric images run in Docker version 17.06.2 with Docker-compose at version 1.8.0. Docker provides a form of virtualization of the operating system without needing to use full virtual machines.

The Android app was modified for the proof of concept with Android Studio version 2.3.3, and the Samsung phones used to test the app had Android 7.0 installed.

5.3.2. Use case

Since the system is built on blockchain technology, a technology based and reliant on a decentralized network, the transport chain actors must run and maintain nodes that communicate with one another to make up a blockchain network. These nodes include endorsing peers and ordering nodes.

The TDL app is used by the transport chain actors every time a TDL-enabled package switches hands from one actor to another. This would include intra-organizational transfers of responsibility, if desired by that specific organization. That choice comes more from how their MSP is set up than from a specific blockchain system design choice. For the purposes of this proof of concept a single person/identity will represent their entire organization, at least in terms of TDL app usage. The general use case for the transport chain actors (involved in the transport of a TDL-package) is illustrated in Fig. 18.

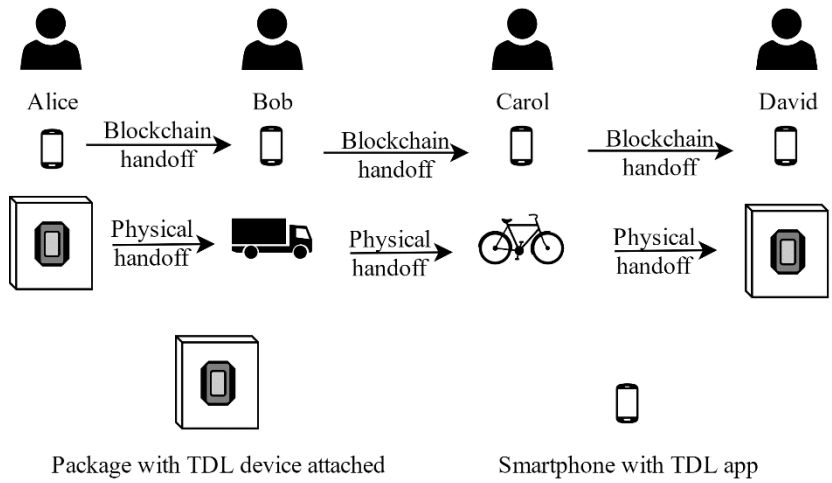


Fig. 18. Alice registers the TDL-package to the Blockchain system with the TDL app, while connected to the TDL device. As part of the physical handoff of the package to Bob they perform the ownership transference procedure (blockchain handoff). The new owner, Bob, connects to the TDL with their app and requests ownership of it. As part of the request a signed data summary from the TDL is included. The current owner, Alice, accepts the request from their own TDL app, although not while connected to the device. This is repeated every step of the way until the final recipient David becomes the owner.

The setup is that the package sender uses the app to set up the TDL sensor parameters. The sender then registers the specific shipment to the TDL blockchain system with the app. The on-blockchain data is illustrated in Fig. 19. As part of registration the TDL app starts the TDL device logging functionality and changes the PIN, see Section 5.3.6.

Once the registration process is completed the package makes its way to the next transport chain actor. That actor will use their smartphone with the TDL app, using their own blockchain network credentials, to request that ownership of the TDL-package is transferred to them. Proposals are requested while connected to the TDL in question. As part of this proposal, the on-device data summary is included. This data summary is signed by the TDL.

The current owner is then able to respond to that proposal using their TDL app. The response is either an accept, where the ownership is transferred, or a rejection, which means the ownership of the TDL is unchanged. This entire sequence of events is illustrated by a state diagram in Fig. 20. In Fig. 21, parts of this sequence are illustrated with two screenshots.

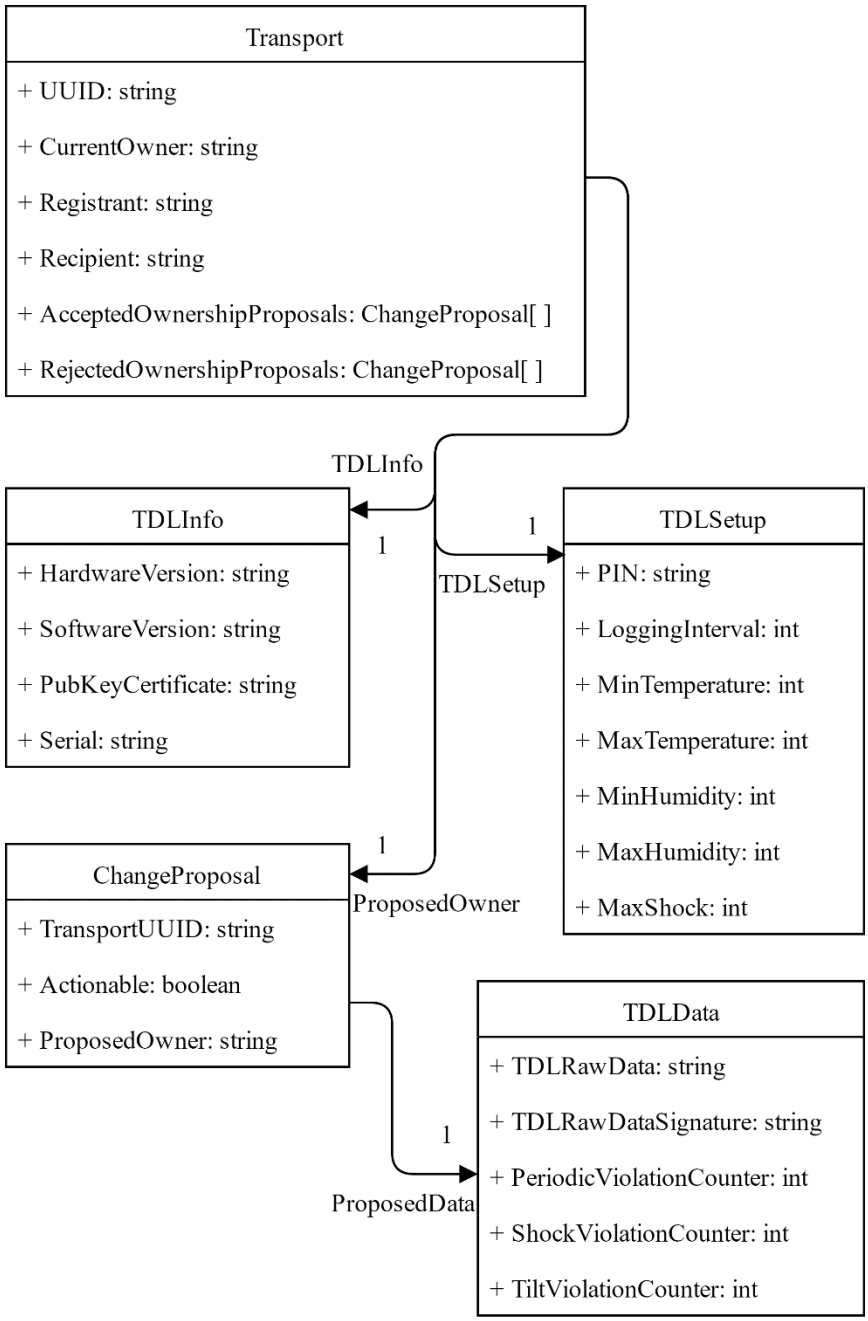
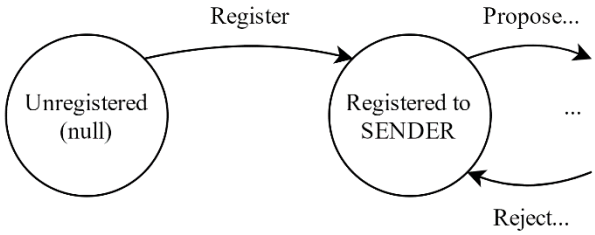


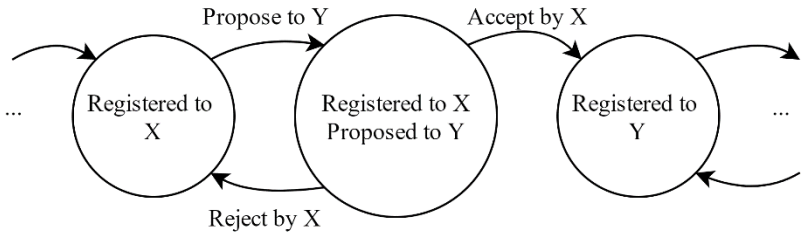
Fig. 19. An illustration of the on-chain data structure. The Transport object represents the entire transportation chain for a specific package,

from registration to final delivery. It does this with data from the actual registration, such as the identity of the TDL and the parameters of its setup. Ownership proposals can be made and will appear as part of the Transport object. These contain the data summary provided to the TDL app by the TDL device. Ownership proposals that are actively rejected or accepted are saved explicitly as being either accepted or rejected. Note that the "Actionable" Boolean variable for ChangeProposal is always set to true for real proposals, it only exists because there was a technical need to allow the adding of an 'unactionable' mock proposal as part of the registration procedure. Note also that all transactions involving the object are saved by the blockchain system as part of the meta-data. Explicitly saving information in directly retrievable objects makes it more easily accessible.

Initial



Ownership change (common case)



Ownership change (final recipient)

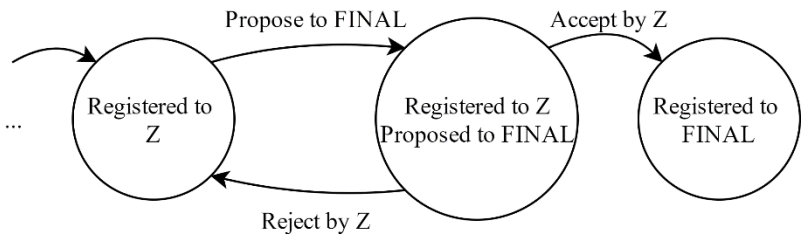


Fig. 20. The different states of a Transport object. The Transport with its unique identifier is registered. Any number of ownership changes can then occur. Once an ownership change to the registration-defined final recipient is made, no further changes can be made.

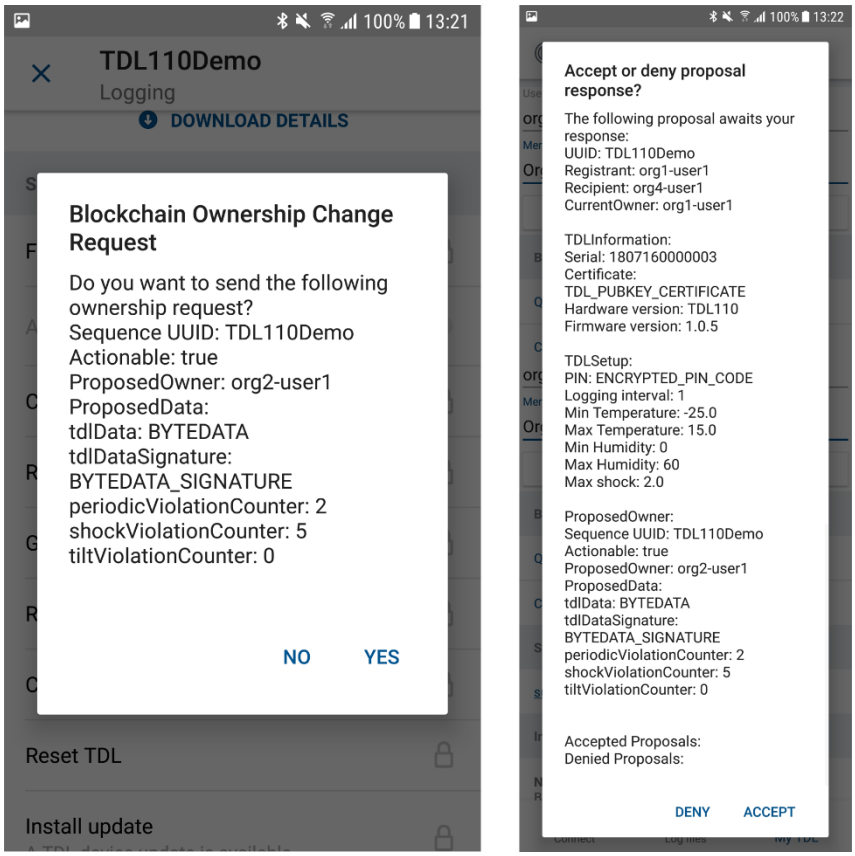


Fig. 21. Left: TDL app with a confirmation pop-up for requesting ownership. Right: The accept/reject pop-up as seen by the current owner. Right screenshot has been altered to show all information at once.

5.3.3. Technical overview

The proof-of-concept network has 4 peers, one for each organization. These all have the proof of concept “Transport” chaincode installed on them so that they can act as endorsing peers. The endorsement policy is set up as a threshold scheme, requiring any three out of the four organizations to endorse a Transport chaincode transaction. The peers and ordering service is provided as part of Hyperledger Fabric and they are written in the Go programming language. The Transport chaincode is written as part of the proof of concept, also in Go.

Because the BFT ordering service has not yet been implemented in the version of Hyperledger Fabric used for development the ordering service for

the proof of concept is of the "single orderer" variety. This means that a single orderer node from the separate mock organization OrdererOrg is used to represent the ordering service. As the lack of a BFT Ordering Service is considered transient, the evaluation will assume the use of a BFT ordering service. The network layout is illustrated in Fig. 22.

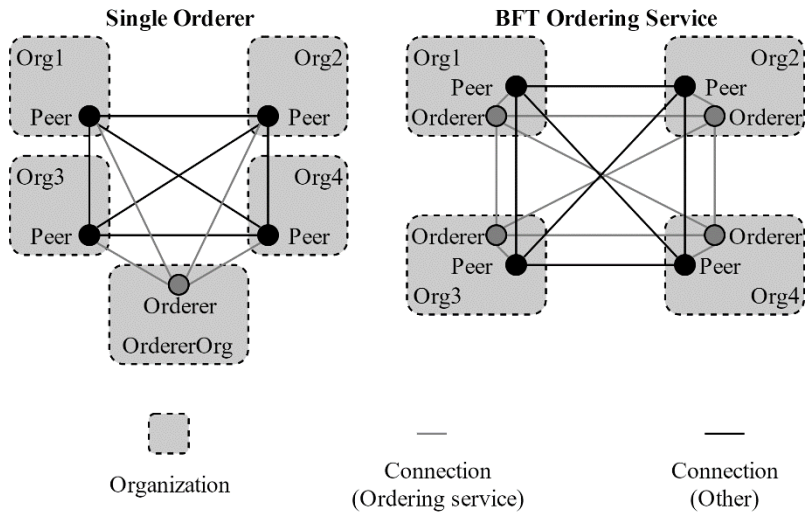


Fig. 22. Left: The single orderer type network used in the proof of concept. Each organization has one peer each and they all connect to the same separately held orderer that represents the ordering service. The single orderer network is for development purposes only. Right: A BFT implementation of the ordering service where each organization runs its own orderer.

In a production-level environment these nodes would all be in different geographic locations (and have redundancy in the form of extra back-up peers) but for development purposes they are all run in Docker containers on the same computer.

A default membership service provider implementation is used. To read more about the MSPs and the use of cryptographic materials in the proof of concept, see Section 5.3.4.

To set up, generate data/credentials for, initialize, start and stop the blockchain network, various bash scripts were created to produce the required results. These were based on scripts included with the Hyperledger Fabric samples.

A blockchain network client (invoker of chaincode) can implement as many checks as it wants but another client can always be written and used to bypass those checks. This means that the Transport chaincode is the only place where trustable verifications of facts can occur. The chaincode must verify that signatures for TDL data are correct, that only those authorized to perform certain invocations can perform them, and perform all other verifications needed to assure that only trustable data enters the blockchain system database. Some of these checks were implemented as part of the proof of concept but as, for example, the chaincode-level access control mechanisms are not available in the current version of Fabric that specific verification functionality is missing.

Effort was made to try and use the Java Fabric Client SDK on Android directly, but it did not work. Porting the SDK to Android as part of the thesis would have been a more time expensive effort than could be afforded.

Speculatively, previous smartphone-related Fabric projects may have used the REST API available in earlier versions of Hyperledger Fabric [72]. Reading [72] highlights some of the issues that are relevant to use of that peer-level REST API.

The bridge-gap solution devised is just for demo purposes, porting the SDK to Android is the acceptable production-level solution. The bridge-gap solution is that a separate PC Java application is used to communicate with the Fabric network. The Android app uses the Java application's command-line interface over SSH. The TDL blockchain system overview is illustrated in Fig. 23.

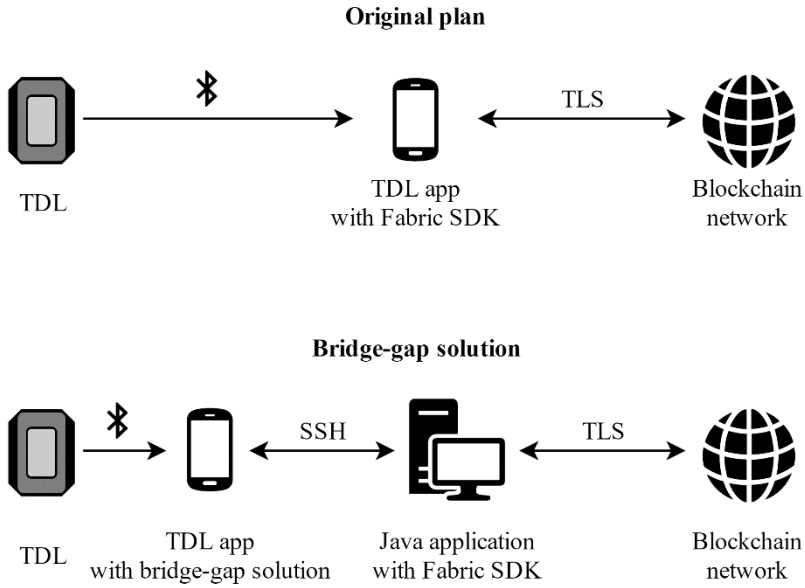


Fig. 23. Above: The TDL acts as a data signer and provides that data to the TDL app over Bluetooth. The app, with Fabric SDK, uses TLS to connect to the Fabric network and conduct transactions. Below: As the current SDK was found to be Android incompatible a bridge-gap solution was devised. An SDK-enabled Java application runs on a computer and communicates with the blockchain network over TLS. The TDL app connects to the computer over SSH and interacts with the command-line interface of the Java SDK-enabled application. The bridge-gap solution is for demonstration purposes only.

The PC client application developed as part of the proof of concept uses the Hyperledger Fabric Java SDK to communicate with the rest of the blockchain system network. It enables the use case described in Section 5.3.2 by invoking the necessary Transport chaincode functions and driving the transaction process as described in Section 3.7.2. The client application is written in Java.

The Android app, was modified from the baseline version, see Section 4.2, to support the use case in Section 5.3.2.

The software architecture, limitations etc. of the TDL device itself was researched using Bosch internal documentation, but no code changes were ultimately made to TDL firmware. The only change would have been the

inclusion of data signing, see Section 5.3.5. The TDL device software is written in C.

5.3.4. Cryptographic materials

The proof of concept makes no attempt to achieve production level standards of handling cryptographic materials. All keys and certificates were generated using the Fabric-provided tool “cryptogen”, and they require no passwords to use. The default MSP implementation is used where the credentials are simply stored locally at each network node. This whole setup is just for development and testing purposes and is not in any way suggested to be used in a production-level environment. In that case the organizations would all use their, most likely pre-existing, PKIs as MSPs.

5.3.5. TDL data summary

The data summary broadcast by the TDL includes counters for each type of violation. These counters are only one byte in size, allowing for the counter to reach 254 (the 255 value is reserved for “violation type turned off”).

Ideally, the summary should include three things: an accurate summary of the data, a signature of that summary and a signature of the complete log data.

Signing of the data summary (or the main data store) of the TDL is not implemented either currently or as part of this thesis. The problem of signing data is an already solved one and implementing it as part of this thesis was not done as it would have taken too much time without furthering the goals of the thesis.

5.3.6. Semi-automatic PIN handling

As opposed to the completely manual PIN handling of the current TDL system the TDL blockchain system would include a sort of semi-automatic PIN handling. The PIN would be generated by the app as part of the registration process. This PIN would, after being set to the TDL, immediately be encrypted using the public key of the final recipient. The ciphertext of the PIN would be included as part of the registration information supplied to the blockchain system.

This would ensure that the only human with access to the PIN would be the final recipient, lowering the possibility of the PIN spreading beyond the person it is needed by. Once the package was transferred to the recipient the app would automatically decrypt the PIN using the recipients private key, and

use the decrypted PIN to change the TDL PIN to some default value such as 0000. It would also stop logging and download the complete log data.

5.4. Summary

Hyperledger Fabric was used to build a TDL Blockchain System. The system records each TDL package handoff by the actors performing a handoff procedure. The procedure includes uploading TDL data to the blockchain system. The handoff procedure is performed using the TDL app to connect to the TDL device and to the blockchain network.

Part III. Conclusions

6. Discussion

The general field of blockchain technology is discussed in Section 6.1, followed by a discussion surrounding the TDL blockchain system and its proof of concept in Section 6.2.

6.1. Blockchain technology

It is hard not to gain an overall positive image of the future of blockchain technology, even with knowledge of its drawbacks. The potential for its use in sectors requiring consensus for data in an environment of distrust is there, but it is ultimately up to business leaders to assess whether that potential is worth pursuing. Down the road of blockchain adoption are several and severe hurdles to overcome.

In a big-picture view, blockchain technology could be trusted to immutably store data. The immutability is extremely dependent on not only the technology itself but also how and by who it is used.

The longest-chain rule of proof-of-work blockchains means that as good as every single transaction on the Bitcoin blockchain could be reversed, even though doing so would be extremely computationally costly. A system that with consensus finality, such as Hyperledger Fabric, has a better technical protection of data immutability, but real-world circumstances such as participating companies deciding to shut the system down could completely render that technical protection useless. As put in Section 3.4, the immutability is not a property inherent to the technology but a result of some larger consensus, even beyond the code that make up the regular consensus process.

6.1.1. Maturity, hype and signaling

A goal of this thesis has been to stick to reality as much as possible in the context of the massive hype surrounding blockchain technology.

Bitcoin is almost a decade old but blockchain technology research and development has not really exploded until the last couple of years. Maturity must be considered low as even the most mature of systems are either extremely basic (Bitcoin) or lack crucial features (Hyperledger Fabric).

News of large companies joining or announcing blockchain technology projects are many, exemplified recently by Microsoft, Samsung and IOTA [73]. As acutely identified by Narula, see Section 3.4, projects having big

names attached to them does not necessarily translate to real-world viability or technical excellence.

Similar thoughts to those expressed in this thesis regarding the maturity of blockchain technology are independently expressed in [74] where you can find a larger discussion and exploration of that topic.

6.1.2. Permissionless vs permissioned blockchains

Permissionless systems using proof-of-work have several drawbacks that make them quite unsuitable for latency averse use cases such as the fast-paced world of transportation. These include having to wait for enough proof-of-work to be accrued to be able to consider a transaction properly committed to the blockchain. Introducing hours of delays for every single package is, even not considering anything else, quite frankly unacceptable. Permissioned systems are much better in this regard, supporting a much larger amount of transactions per second and quickly adding them permanently to the blockchain.

Proof-of-stake is not yet proven to be a dependable form of consensus. It could alleviate many of the resource and capacity problems associated with proof-of-work but at the cost of using an unproven consensus form. Only once proof-of-stake has been properly developed and researched to be reliable can it be considered for implementation in a system such as the one this thesis is about.

All permissionless systems suffer from a fundamental privacy issue. All data supplied to the permissionless blockchain is per definition shared with the public. Encrypting the data before supplying it to the blockchain so it could only be read by some subset of peers could increase the likelihood of bad data being supplied to the system. If the blockchain system chaincode can't read the data, it does not have any ability to assess the reliability of that data. Permissioned systems have a chance of increased data privacy as data is kept between identifiable organizations. But because the data is stored in multiple places and by multiple parties the possibility of a data leak is increased, see the 'weakest link' adage.

There is always a cost associated with running a blockchain system. Some of the cost for a permissionless system comes in the form of hardware and electricity. For a proof-of-work system the electricity demand is huge, meaning that hosting these network nodes comes with a big electricity bill. Even without the cost of running proof-of-work nodes all transactions would still need to be paid for with some form of (crypto-)currency. For example, running chaincode on Ethereum costs "gas", and transactions for Bitcoin are,

at least in the long term, supported by transaction fees to the miners who add the transactions to their blocks.

For permissioned systems, setting up network nodes is a bit more like hosting traditional servers and the cost likely more predictable. However, the total cost for setting up and maintaining the system, as it requires so much inter-organizational cooperation, is likely quite large. The level of cooperation required is an identified hurdle for adopting (permissioned) blockchain technology [1] [5]. The required inter-organizational cooperation cannot benefit from any established standards further increasing the workload on all parties. All participating organizations would have to keep people employed for the explicit purpose of running the network, writing the chaincode etc. Trusting a single actor with these things would effectively centralize the system.The nodes (and chaincode development/analysis) must be in direct and competent control of the individual participating organizations for the system to provide any benefit at all as compared to a traditional database system. Ceding control means abandoning what makes blockchain technology special.

6.2. The TDL blockchain system and its proof of concept

Conceptually, a system like this provides benefits of data storage and attribution but the issues surrounding it are numerous and considerable. The goal of attribution is met conceptually but not in reality, as the blockchain data and metadata cannot be enforced to overlap completely as per the current lack of access control for chaincode.

For large companies with high technical competence the maintenance problem, see Section 6.1.1, is most likely not an issue but for smaller local companies the work needed to maintain such a system could be too advanced and expensive in terms of manpower. Not participating would mean that they could not handle TDL packages as the premise of the system is that everyone who takes responsibility for the TDL-package must also do so as part of the blockchain system. Shifting the maintenance work to other parties would likely increase centralization.

Beyond the maintenance problem the additional costs are potentially very high, though not necessarily directly monetary. Everything regarding the usage of the system must be considered, including the actual human beings that would have to use it every day. Every single organization, and inter-organizationally every employee, who could come into contact with a TDL-equipped package would have to agree (and be trained) to take part in the blockchain system and to always use it to handle TDL packages.

A package can, of course, be completely sent through the transport chain without all handoffs being recorded to the blockchain but that would cause an irreparable rift in the data attribution accuracy, see Section 6.2.1 for further discussions about differences in historical and documental truth. This could be done unknowingly too, a TDL-equipped package could be contained within some larger multi-package container and not immediately apparent to the handoff parties resulting in the blockchain handoff never taking place.

The question is of course raised what the ultimate justification for participating in a system like the one in this thesis is. This is ultimately a subjective judgement for the business leaders. Sections 5.1.1 and 5.1.2 contain some justifications that could be used to motivate such a decision.

The proof of concept suffers from definite technical issues, including those associated with Fabric itself, see Section 5.3.

About PIN-handling: there is still the possibility of the recipient using their private key to manually decrypt the PIN code before they receive the package. They would then be able to spread the PIN code similarly to the current system, compromising the TDL security solution. This could potentially be solved with some sort of threshold scheme to decrypt the PIN code only once the package is delivered.

6.2.1. Historical vs documental truth

To make sure the data points are attributed to the correct transport chain actor, the handoff procedure must be correctly performed at every single physical handoff of TDL packages from one actor to another.

Even if done correctly, the TDL data summary uploaded as part of the handoff does not necessarily completely overlap with the raw TDL data. The limited capacity violation counters being the main reason. On-device data could still be manipulated between each handoff given enough technical expertise. Including a signature for the complete data in addition to the summary could alleviate some of these problems.

Even if all TDL data points are accurately recorded and stored on the blockchain, the package could still be completely destroyed/ruined along the way. A package could, for example, be opened and the contents stolen, or the package could be damaged in the form of cutting. There is simply not a complete overlap between the sensor values recorded by the TDL and all adverse effects that could affect the package. This is not really evaluated as part of the thesis, but still important enough to at least acknowledge. A handoff procedure including some sort of visual inspection could help alleviate some of these problems.

It is not specifically adressed in the proof of concept, but how to ultimately handle the complete log data once the package is delivered is important. It would likely require code changes all the way from the TDL device to the chaincode but uploading the complete data as part of the final handoff could be implemented. Other ways of handling it is discussed in Section 6.2.2. The larger question is the one of storing certain data on- or off-chain, see Section 3.5. Storing it on the blockchain would increase storage demands and increase the connection between the blockchain system and the raw TDL data. Storing it off-chain, perhaps in some other database, with the blockchain only containing the summary and its signature along with the signature of the complete raw data would decrease required blockchain network node storage. Uploading a megabyte of complete log data is not necessarily easy either. Mobile data cannot be assumed to be fast, or available, everywhere TDL-equipped packages would need to be handed off between actors.

6.2.2. Other blockchain system solutions

If the TDL device were to be equipped with internet connectivity, the possibility to use it as a blockchain network client and/or node opens. As the TDL baseline used in this thesis does not include internet connectivity the discussion of this ‘future TDL’ is mostly speculative.

The requirements for a peer are in the case of Hyperledger Fabric quite heavy, at least in terms of network and storage usage. A peer needs to store the complete channel blockchain, a big ask of a device currently only equipped with 1MB of log data storage. The firmware storage is even smaller. The entire Hyperledger Fabric network peer code, including its dependencies, would have to be ported to the C language used by the TDL firmware. A task that even just by itself is quite monumental. Restricting the use of the TDL to that of a Fabric client would mean not needing to store the blockchain. The network traffic associated with driving the transaction process is still considerable compared to a simple data transfer. For the sake of a frame of reference for code storage requirements, the linux-amd64 Fabric peer binary and the Java SDK both used in this thesis are roughly 20MB in size each. The blockchain data would, admittedly with a different but related system, require orders of magnitude more storage than currently available on the TDL [75].

Considering a “local” blockchain is pointless. A blockchain that is not network-distributed offers no more ‘real’ protection of its data than that of signatures. Anyone with read/write access could easily delete it. Blockchain data immutability comes from both network distribution and the data structure, not from either or.

On the question of uploading the entire log data versus just the summary there are a few tradeoffs. Uploading all data would decrease battery life as more network traffic would be required. The TDL would of course also have to be updated to allow full log data downloads without PIN access.

Suppose the TDL uploaded its data to the blockchain automatically, directly from the device. This would likely offer good data protection, assuming secure connection procedures and well-enforced TDL digital identities and data signatures. Loss of internet access or otherwise sparse data uploads, deliberate or otherwise, could cause a gap between the TDL's data as recorded on the blockchain and on the device. This would lead to similar data attribution difficulties as with the current TDL system, having to use timestamps to arbitrate between which data existed before or after a certain handoff took place. This means the handoff procedure would still have to include a sync between the on-device data and the blockchain to accurately be able to attribute exactly which data points occurred in relation to what actor was responsible for the TDL at a certain point in time.

Using a relatively limited device, common in Internet of Things (IoT), as a proof-of-work blockchain network node would quite likely not be worthwhile as such a system requires a large amount of CPU-cycles, electricity and available storage. Instead the device would have to talk to some sort of intermediary server. This would require complete trust in that server to perform the correct tasks with that data. A catch-22 scenario can be envisioned where a device would need an intermediary server to connect to the blockchain network in its place but to independently verify that this is properly done the device would have to completely implement the network node functionality rendering the intermediary server pointless.

7. Future work

In a permissioned blockchain system mapping real-world power structures to a few dozen network nodes would take considerable work. Making sure participating organizations do not become de-facto Sybil identities would require mapping complicated corporate structures and ownership schemes and accurately reflecting them in the network power distribution. As existing participating organizations close, are acquired and new ones emerge and need to be included being able to maintain the network power structures in relation to those of the real world changing would again be a source of potential research.

An area that is of great importance, especially for consideration of using internet-of-things devices as peers, is the one of blockchains having an ever-increasing storage footprint. It is important to research long-term solutions to this problem. For more information regarding this issue, see [75].

Other research into the consequences of running blockchain networks long-term is needed. Permissionless systems explored in this thesis depend on the distribution, and at times re-distribution, of some sort of blockchain-based token to incentivize network participation. Research exploring the links between the real-world token value, its intra- and extra-blockchain use and the overall health of the blockchain network could yield intriguing results.

Researching the link between on-blockchain data and the real world is an area of great potential. Making sure the on-blockchain data accurately reflects the real world requires rigorous blockchain system verifications. Finding the best ways of performing these, with or without the need for trusted third parties, would have an impact on what use cases for blockchain technology can be considered viable. That includes the TDL blockchain system of this thesis, where the gap between historical and documental truth is a source of much potential headache.

There is still a large amount of research and development needed to bring the proof of concept developed as part of this thesis into a fully developed product, if one wanted to do so. For example, several features of Fabric needed to make the product work are simply not yet developed, most importantly the sort of chaincode level access control needed to verify the connection between the owner of a TDL and who is invoking the owner-changing chaincode.

As the Transport Data Logger examined in the thesis is a snapshot in time of a product in active development the blockchain solution could of course change as the capabilities of the Transport Data Logger device becomes more

capable. For instance, once the TDL device has network access the possibilities of directly using it as a blockchain network node emerges.

8. Summary

The field of blockchain is still in its infancy and there is a lot of work that can and should be done in the field. There are huge hurdles to overcome and its adoption depends on if the advantage of more decentralized and trustable data storage and distribution is worth getting over the hurdles for.

Because the hype and promise of blockchain technology is so massive finding and pointing to the problems associated with it, as has been done in this thesis, feels somewhat like raining on its parade.

Blockchain technology should be considered when wanting to keep data safe from manipulation, but it is important to recognize that blockchains are only capable of protecting the data provided to them. If a blockchain system is fed data that cannot already be considered trustworthy then the system will not make that data trustworthy, only immutable. And even that is with some important asterisks such as distribution of blockchain network power and putting trust in blockchain system developers.

This thesis is very much a product of its time. I fully anticipate similar future theses on blockchain technology to yield different and likely more impressive results because of the volatile, nascent, and fast-moving nature of the field.

8.1.1. Question 1 – Permissioned vs permissionless blockchains

What are the different strengths and weaknesses of permissioned and permissionless blockchains?

Permissionless systems scale better to many thousands of nodes, whereas permissioned systems only support twenty or so nodes. They both support thousands of clients. Permissionless systems can be a lot more decentralized thanks to supporting an enormous number of nodes. This comes at the cost of poor transaction performance with high latencies and low transaction bandwidth.

Permissionless systems using proof-of-work use (or arguably waste) an enormous amount of electricity. Permissioned systems require rigorous use of public key infrastructure and a huge amount of inter-organizational cooperation.

8.1.2. Question 2 – Data reliability

How can blockchain technology be used to increase the reliability of the Transport Data Logger data?

The data can be trusted to be stored immutably, which does technically increase reliability, but blockchain technology does not itself make the data more trustworthy. In other words, the data itself is only as trustworthy as it is outside the blockchain system. This means that all the usual practices of data signing are still just as required to provide a trusted connection between the real-world and the on-blockchain data.

Periodically uploading (signed) data automatically could potentially increase reliability. This would require internet connectivity and is not possible with the current TDL device.

8.1.3. Question 3 – Data attribution

How can blockchain technology be used to attribute specific data points from the Transport Data Logger to specific transport chain actors?

A permissioned blockchain system requiring rigorous use of inter-organizational PKI can be used to attribute specific data points to individual organizations and, if needed, individuals within those organizations. Uploading data points as they have been accumulated up to the precise time of the package handoff means that a direct connection between the log data and the responsible actor is established.

Other solutions that would potentially increase data reliability such as automatic uploads have the potential to weaken the link between the two unless the handoff upload is somehow still part of the process.

8.1.4. Recommendation on blockchain technology adoption

For organizations who want to get in on the ground floor and help develop the underlying systems and standards this is the time to do so. For those not interested in doing that or who just wants to use finished software the blockchain technology sector is not yet ready for them. It will likely be years before it is.

References

[1] A. Jeppsson and O. Olsson, "Blockchains as a solution for traceability and transparency," MSE thesis, Dept. Design Sci., Lund University, Lund, Sweden, 2017.

[2] O. Petersen and F. Jansson, "Blockchain Technology in Supply Chain Traceability Systems," MSE thesis, Dept. Ind. Manage. and Logistics, Lund University, Lund, Sweden, 2017.

[3] J. Yli-Huumo, D. Ko, S. Choi, S. Park and K. Smolander, "Where Is Current Research on Blockchain Technology?—A Systematic Review," *PLoS ONE*, vol. 11, no. 10, pp. 1-27, ISSN: 1932-6203, October 2016.

[4] M. Scherer, "Performance and Scalability of Blockchain Networks and Smart Contracts," MSE thesis, Dept. Computing Sci., Umeå University, Umeå, Sweden, 2017.

[5] Q. Lu and X. Xu, "Adaptable Blockchain-Based Systems: A Case Study for Product Traceability," *IEEE Software*, vol. 34, no. 6, pp. 21-27, ISSN: 07407459, 2017.

[6] IBM, "AOS and IBM developing logistics and transportation solution built on IBM Blockchain and Watson IoT," IBM, 21 June 2017. [Online]. Available: https://www.ibm.com/press/us/en/pressrelease/52665.wss. [Accessed 27 November 2017].

[7] IBM, Maersk, "Maersk and IBM Unveil First Industry-Wide Cross-Border Supply Chain Solution on Blockchain," IBM, 5 March 2017. [Online]. Available: http://www-03.ibm.com/press/us/en/pressrelease/51712.wss. [Accessed 27 November 2017].

[8] M. Hell, "EIT060 Computer Security 2016/2017 Vt1 Lectures - Lect2-3," 18-19 January 2017. [Online]. Available: http://www.eit.lth.se/fileadmin/eit/courses/eit060/lect/Lect2-3.pdf. [Accessed 25 September 2017].

[9] H. C. A. van Tilborg, Encyclopedia of Cryptography and Security, Boston, MA: Springer US, ISBN: 9781441959065, 2011.

[10] dr jimbob, "cryptography - Why shouldn't we roll our own - Information Security Stack Exchange," Stack Exchange Inc, 6 August 2012. [Online]. Available: https://security.stackexchange.com/questions/18197/why-shouldnt-we-roll-our-own/18198. [Accessed 25 September 2017].

[11] N. Narula, "Cryptographic vulnerabilities in IOTA," 7 September 2017. [Online]. Available: https://medium.com/@neha/cryptographic-vulnerabilities-in-iota-9a6a9ddc4367. [Accessed 20 November 2017].

[12] Cornell University, "RSA Signing is Not RSA Decryption," Cornell University, 2015. [Online]. Available: https://www.cs.cornell.edu/courses/cs5430/2015sp/notes/rsa_sign_vs_dec.php. [Accessed 15 November 2017].

[13] I. Eyal and E. G. Sirer, "Majority Is Not Enough: Bitcoin Mining Is Vulnerable," in *Financial Cryptography and Data Security 18th International Conference*, DOI: 10.1007/978-3-662-45472-5_28, Christ Church, Barbados, 2014, pp. 436-54.

[14] M. Vukolic, "The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication," in *Int. Workshop on Open Problems in Network Security*, ISBN: 9783319390277, Zurich, Switzerland, 2015.

[15] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 31 October 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf. [Accessed 1 September 2017].

[16] D. Ichikawa, M. Kashiyama and T. Ueno, "Tamper-Resistant Mobile Health Using Blockchain Technology," *JMIR Mhealth Uhealth,* vol. 5, no. 7, p. e111, DOI: 10.2196/mhealth.7938, 2017.

[17] C. Cachin and M. Vukolic, "Blockchain Consensus Protocols in the Wild," arXiv:1707.01873v2 [cs.DC], 2017.

[18] F. B. Schneider, "Implementing Fault-Tolerant Services Using the State Machine Approach: A Tutorial.," *ACM Computing Surveys,* vol. 22, no. 4, pp. 299-319, ISSN: 15577341, 1990.

[19] Bitcoin Project, "51% Attack, Majority Hash Rate Attack," Bitcoin Project, 24 April 2015. [Online]. Available: https://bitcoin.org/en/glossary/51-percent-attack. [Accessed 23 November 2017].

[20] L. Lamport, R. Shostak and M. Pease, "The Byzantine generals problem," *ACM Transactions on Programming Languages and Systems,* vol. 4, no. 3, pp. 382-401, ISSN: 01640925, 1982.

[21] J. R. Douceur, "The Sybil Attack," in *Peer-to-Peer Systems - 1st International Workshop, IPTPS 2002, Revised Papers*, ISSN: 16113349, 2002.

[22] E. Mik, "Smart contracts: terminology, technical limitations and real world complexity," *Law, Innovation & Technology,* vol. 9, no. 2, pp. 269-300, ISSN: 1757-9961, 2017.

[23] Hyperledger, "Glossary," Hyperledger, 17 September 2017. [Online]. Available: http://hyperledger-fabric.readthedocs.io/en/latest/glossary.html. [Accessed 21 November 2017].

[24] E. Heilman, N. Narula, T. Dryja and M. Viza, "IOTA Vulnerability Report: Cryptanalysis of the Curl Hash Function Enabling Practical Signature Forgery Attacks on the IOTA Cryptocurrency," 7 September 2017. [Online]. Available: https://github.com/mit-dci/tangled-curl/blob/master/vuln-iota.md. [Accessed 20 November 2017].

[25] N. Atzei, M. Bartoletti and T. Cimoli, "A survey of attacks on Ethereum smart contracts (SoK)," in *Principles of Security and Trust - 6th International Conference, POST 2017 Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Proceedings*, ISSN: 16113349, 2017.

[26] R. Waters, "Automated company raises equivalent of $120M in digital currency," CNBC, 17 May 2013. [Online]. Available: https://www.cnbc.com/2016/05/17/automated-company-raises-equivalent-of-120-million-in-digital-currency.html. [Accessed 20 November 2017].

[27] C. Brennan and W. Lunn, "Blockchain: The Trust Disrupter," Credit-Suisse, 3 August 2016. [Online]. Available: https://plus.credit-suisse.com/rpc4/ravDocView?docid=NQDC92AF-WErFXS. [Accessed 19 January 2018].

[28] Ethereum Classic, "The Ethereum Classic Declaration of Independence," 20 July 2016. [Online]. Available: https://ethereumclassic.github.io/assets/ETC_Declaration_of_Independence.pdf. [Accessed 20 November 2017].

[29] P. B. Nichol, "If I only had 5 minutes to explain blockchain," IDG Contributor Network, 30 August 2016. [Online]. Available: https://www.cio.com/article/3112773/it-industry/if-i-only-had-5-minutes-to-explain-blockchain.html. [Accessed 27 November 2017].

[30] A. Madeira, "What are Mining Rewards in Ethereum?," CryptoCompare, 28 September 2017. [Online]. Available:

https://www.cryptocompare.com/mining/guides/what-are-mining-rewards-in-ethereum/. [Accessed 27 November 2017].

[31] Trustnodes, "Ethereum May Reduce Mining Reward and Inflation by 40%," Trustnodes, 15 July 2017. [Online]. Available: http://www.trustnodes.com/2017/07/15/ethereum-may-reduce-mining-reward-inflation-40. [Accessed 27 November 2017].

[32] Digiconomist, "Bitcoin Energy Consumption Index - Digiconomist," Digiconomist, 22 November 2017. [Online]. Available: https://digiconomist.net/bitcoin-energy-consumption. [Accessed 23 November 2017].

[33] POWERCOMPARE.CO.UK, "Bitcoin Mining Now Consuming More Electricity Than 159 Countries Including Ireland & Most Countries In Africa," POWERCOMPARE.CO.UK, 20 November 2017. [Online]. Available: https://powercompare.co.uk/bitcoin/. [Accessed 23 November 2017].

[34] V. A. Red, "Practical comparison of distributed ledger technologies for IoT," in *Proc. SPIE*, DOI: 10.1117/12.2262793, Anaheim, CA, USA, 2017.

[35] R. Burn-Callander, "LXC Coin crowdfunds in challenge to Bitcoin," The Telegraph, 16 September 2014. [Online]. Available: http://www.telegraph.co.uk/finance/businessclub/money/11100354/LXC-Coin-crowdfunds-in-challenge-to-Bitcoin.html. [Accessed 16 November 2017].

[36] W. Li, S. Andreina, J.-M. Bohli and G. Karame, "Securing Proof-of-Stake Blockchain Protocols," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology - ESORICS 2017 International Workshops, DPM 2017 and CBT 2017, Proceedings*, ISSN: 16113349, 2017.

[37] Ethereum community, "Mining - Ethereum Homestead Documentation," Ethereum Community, 18 August 2016. [Online]. Available: http://ethdocs.org/en/latest/mining.html. [Accessed 14 September 2017].

[38] V. Zamfir, "Introducing Casper “the Friendly Ghost” - Ethereum Blog," 1 August 2015. [Online]. Available: https://blog.ethereum.org/2015/08/01/introducing-casper-friendly-ghost/. [Accessed 14 September 2017].

[39] The Linux Foundation, "Projects," The Linux Foundation, 23 November 2017. [Online]. Available: https://www.hyperledger.org/projects. [Accessed 23 Novemebr 2017].

[40] The Linux Foundation, "Members," The Linux Foundation, 23 November 2017. [Online]. Available: https://www.hyperledger.org/members. [Accessed 23 November 2017].

[41] B. Patrick Eha, "Blockchain moves beyond its 'moonshot' phase," *American Banker*, p. 1, ISSN: 0002-7561, 17 November 2017.

[42] The Linux Foundation, "Hyperledger Fabric," The Linux Foundation, 23 November 2017. [Online]. Available: https://www.hyperledger.org/projects/fabric. [Accessed 23 November 2017].

[43] Intel Corporation, "Introduction," Intel Corporation, 2 November 2017. [Online]. Available: https://sawtooth.hyperledger.org/docs/core/releases/latest/introduction.html. [Accessed 23 November 2017].

[44] Hyperledger Project, "Start," Hyperledger Project, 2 November 2017. [Online]. Available: https://wiki.hyperledger.org/start. [Accessed 23 November 2017].

[45] Intel Corporation, "Intel Q3’17 ME 11.x, SPS 4.0, and TXE 3.0 Security Review Cumulative Update," Intel Corporation, 22 November 2017. [Online]. Available: https://security-center.intel.com/advisory.aspx?intelid=INTEL-SA-00086&languageid=en-fr. [Accessed 24 November 2017].

[46] M. Brandenburger, C. Cachin, M. Lorenz and R. Kapitza, "Rollback and Forking Detection for Trusted Execution Environments Using Lightweight Collective Memory," in *Dependable Systems and Networks (DSN), 2017 47th Annual IEEE/IFIP International Conference on*, ISSN: 2158-3927, Denver, CO, 2017.

[47] M. Schwarz, S. Weiser, D. Gruss, C. Maurice and S. Manguard, "Malware Guard Extension: Using SGX to Conceal Cache Attacks," in *Detection of Intrusions and Malware, and Vulnerability Assessment - 14th International Conference, DIMVA 2017*, ISSN: 16113349, 2017.

[48] I. Eyal, "Blockchain Technology: Transforming Libertarian Cryptocurrency Dreams to Finance and Banking Realities," *Computer*, vol. 50, no. 9, pp. 38-49, DOI: 10.1109/MC.2017.3571042, 2017.

[49] M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance and Proactive Recovery," *ACM Transactions on Computer Systems*, vol. 20, no. 4, pp. 398-461, ISSN: 07342071, 2002.

[50] A. Bessani, J. Sousa and E. Alchieri, "State Machine Replication for the Masses with BFT-SMART," in *Dependable Systems and Networks (DSN), 2014 44th Annual IEEE/IFIP International Conference on*, ISBN 978-1-4799-2233-8, Atlanta, GA, 2017.

[51] J. Sousa and A. Bessani, "Separating the WHEAT from the Chaff: An Empirical Design for Geo-Replicated State Machines," in *Reliable Distributed Systems (SRDS), 2015 IEEE 34th Symposium on*, ISSN: 10609857, Montreal, Canada, 2015.

[52] J. Cowling, "HQ replication," MSE thesis, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 2007.

[53] Hyperledger, "Hyperledger Fabric Model," Hyperledger, 31 October 2017. [Online]. Available: http://hyperledger-fabric.readthedocs.io/en/latest/fabric_model.html. [Accessed 23 November 2017].

[54] Hyperledger , "Architecture Explained," Hyperledger , 12 October 2017. [Online]. Available: http://hyperledger-fabric.readthedocs.io/en/latest/arch-deep-dive.html. [Accessed 23 November 2017].

[55] Hyperledger, "Transaction Flow," Hyperledger, 17 September 2017. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/latest/txflow.html. [Accessed 27 November 2017].

[56] Hyperledger, "Membership Service Providers (MSP)," Hyperledger, 19 September 2017. [Online]. Available: http://hyperledger-fabric.readthedocs.io/en/latest/msp.html. [Accessed 28 November 2017].

[57] Hyperledger, "Hyperledger Fabric Capabilities," Hyperledger, 4 July 2017. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/latest/capabilities.html. [Accessed 27 November 2017].

[58] A. Sorniotti, "[FAB-3057] Chaincode-level access control," Hyperledger, 13 April 2017. [Online]. Available: https://jira.hyperledger.org/browse/FAB-3057. [Accessed 28 November 2017].

[59] S. Muralidharan, "[FAB-1219] Support access control policies for chaincode (creation, termination, invocation - in terms of chaincode

library)," Hyperledger, 4 April 2017. [Online]. Available: https://jira.hyperledger.org/browse/FAB-1219. [Accessed 28 November 2017].

[60] Bosch Connected Devices and Solutions GmbH, "Transport Data Logger | TDL110 Data Sheet," 19 May 2017. [Online]. Available: http://ae-bcds.resource.bosch.com/media/_tech/2017_05_19_Datasheet_TDL.pdf. [Accessed 20 November 2017].

[61] Bosch Connected Devices and Solutions GmbH, "Transport Data Logger TDL 110 Operating Instructions," September 2017. [Online]. Available: http://ae-bcds.resource.bosch.com/media/_tech/BCDS-TDL110-0001_2017-09_EN.pdf. [Accessed 20 November 2017].

[62] Bosch Connected Devices and Solutions, "Transport Data Logger - TDL 110," Bosch Connected Devices and Solutions, 24 November 2017. [Online]. Available: http://www.bosch-connectivity.com/en/what_we_offer/products_and_solutions_1/tdl/connected_devices_1. [Accessed 24 November 2017].

[63] B. Grawemeyer and H. Johnson, "Using and managing multiple passwords: A week to a view," *Interacting with Computers*, vol. 23, no. 3, pp. 256-267, ISSN: 0953-5438, 2011.

[64] G. B. Duggan, H. Johnson and B. Grawemeyer, "Rational security: Modelling everyday password use," *International Journal of Human - Computer Studies*, vol. 70, no. 6, pp. 415-431, DOI: 10.1016/j.ijhcs.2012.02.008, 2012.

[65] M. Ryan, "Bluetooth: With Low Energy Comes Low Security," in *7th USENIX Workshop on Offensive Technologies*, Washington, D.C., 2013.

[66] S. S. Hassan, S. D. Bibon, M. S. Hossain and M. Atiquzzaman, "Security threats in Bluetooth technology," *Computers and Security*, Article in Press, ISSN: 0167-4048,2017.

[67] J. W. Oh, "Reverse Engineering Flash Memory for Fun," in *Black Hat USA*, Las Vegas, 2014.

[68] M. Wilson, "Retailers and producers turn to IBM Blockchain to improve food safety," IBM, 24 August 2017. [Online]. Available: https://www.ibm.com/blogs/cloud-computing/2017/08/blockchain-food-safety/. [Accessed 28 November 2017].

[69] T. Leek, "opensource - Trust Issues Relative to Open Source - Information Security Stack Exchange," Stack Exchange, 3 September 2013. [Online]. Available: https://security.stackexchange.com/questions/41734/trust-issues-relative-to-open-source. [Accessed 29 November 2017].

[70] K. Thompson, "Reflections on Trusting Trust," *Communications of the ACM*, vol. 27, no. 8, pp. 761-763, ISSN: 0001-0782, 1984.

[71] M. Walport, "Distributed ledger technology: beyond block chain," Government Office for Science, London, UK, 2016.

[72] A. Derbakova, "[FAB-156] Create a REST wrapper around SDK," Hyperledger, 17 August 2017. [Online]. Available: https://jira.hyperledger.org/browse/FAB-156. [Accessed 28 November 2017].

[73] R. Browne, "A little-known digital currency surges 70% after teaming up with firms like Microsoft," CNBC, 4 December 2017. [Online]. Available: https://www.cnbc.com/2017/12/04/cryptocurrency-iota-rallies-after-launch-of-data-marketplace.html. [Accessed 4 December 2017].

[74] M. Juden and M. Pisa, "Blockchain and Economic Development: Hype vs. Reality," Center for Global Development, Washington, DC, 2017.

[75] E. Palm, "Implications and Impact of Blockchain Transaction Pruning," MSE thesis, Department of Computer Science, Electrical and Space Engineering, Luleå University of Technology, Luleå, Sweden, 2017.

Printed by Tryckeriet i E-huset, Lund 2018

LUND
UNIVERSITY

Series of Master's theses
Department of Electrical and Information Technology
LU/LTH-EIT 2018-613

http://www.eit.lth.se