UNIVERSITY OF TARTU
Institute of Computer Science
Software Engineering Curriculum

Andreas Ellervee

A Reference Model for Blockchain-Based Distributed Ledger Technology

Masters's Thesis (30 ECTS)

Supervisor: Raimundas Matulevičius, PhD
Supervisor: Nicolas Mayer, PhD

Tartu 2017

A Reference Model for the Blockchain-Based Distributed Ledger Technology

Abstract:

Blockchain is a distributed, transactional database that is shared across all the nodes participating in the network. This is the main technical innovation of Bitcoin and it acts as a public ledger for the transactions. Every node in the system has a full copy of the current chain, which contains every transaction ever executed. Every block contains a hash of the previous block, linking these two together. The linked blocks become a blockchain. However, this technology lacks standardisation and uniform understanding. This is due to a few studies, that would provide a comprehensive model of the blockchain and the distributed ledger technology. In this thesis we compare four blockchain technology platforms and focus on their business level properties including actors and roles, services, processes and data model. Our comparison results in a reference model, which could potentially guide the business and system analysts and software developers when developing new blockchain platforms or their supported implementations. Accuracy of the proposed reference model is validated by considering it against selected blockchain platforms. The reference model is also validated via application, showing its usefulness for a risk-related blockchain security assessment.

Keywords: Blockchain technology, Reference model, Distributed ledger, Bitcoin, Etherum

CERCS: T120 - Systems engineering, computer technology

Standardmudel Plokiahelal Põhinevale Hajusale Raamatupidamistehnoloogiale

Lühikokkuvõte:

Plokiahel on hajus, transaktsioonidel põhinev andmebaas, mis on jagatud kõigi kasutajate vahel. See on Bitcoin'i põhiline tehnoloogiline innovatsioon ning plokiahela roll on olla hajus pearaamat kõikidele transaktsioonidele. Igal kasutajal (sõlm, ingl.k. 'node') on terve koopia kõige hilisemast ahelast, mis hoiab endas igat transaktsiooni, mis kunagi tehtud. Iga plokk sisaldab endas eelmise ploki räsiväärtust (ingl.k. 'hash'), mis seob kaks plokki omavahel. Ühendatud plokid moodustavadki plokiahela. Paraku sellisel tehnoloogial puudub standard ja ühtne arusaam. Selle põhjuseks on vähesed akadeemilised uurimustööd, mis käsitleksid seda tehnoloogiat ja kirjeldaksid standardiseeritud mudelit. Käesoleva lõputöö eesmärgiks on võrrelda nelja plokiahela tehnoloogiat ning keskenduda nende äritaseme atribuutidele - tehnoloogias osalejad ja nende rollid, teenused, äriprotsessid ja andmemudelid. Analüüsi ja võrdluse tulemusena valmib standardiseeritud mudel, mis võib potentsiaalselt abistada analüütikuid ja arendajaid plokiahela tehnoloogiaga töötamisel. Loodud mudeli täpsuse valideerimiseks võrreldakse seda erinevate plokiahela tehnoloogiatega. Lisaks, valideerime standardmudelit kasutades seda riskipõhise analüüsi teostamisel.

Märksõnad: Plokiahela tehnoloogia, standardmudel, hajus raamatupidamine, Bitcoin, Etherum

CERCS: T120 - Süsteemitehnoloogia, arvutitehnoloogia

Contents

1 Introduction

The blockchain technology was introduced in 2008 and the first implementation, i.e. Bitcoin, was introduced a year later, in 2009, published in the paper "Bitcoin: A Peer-to-Peer Electronic Cash System" under the alias Satoshi Nakamoto [21]. Since Bitcoin's release, the popularity of the cryptocurrency has only kept growing, because customers have started value the convenience and security of digital currencies[1], enabled by the blockchain technology. In the traditional banking systems, the ledger is a centralised party (e.g., the bank), which stores all the transactions. Blockchain, which serves as the decentralized public ledger for bitcoin, can also be applied to other fields, such as healthcare, insurance, data verification and others.

Different businesses have developed various implementations using the blockchains, however, this technology lacks standardization [23]. Only limited analysis [11] exists on the conceptual explanation and understanding of the blockchain technology.

This thesis is focused on unifying this understanding and proposes a comprehensive reference model to characterise the blockchain technology. The main research question is "How to unify the understanding of the technology, through a reference model?". Our proposed model will be developed by investigating and comparing existing blockchain implementations. Being presented in ArchiMate, BPMN and UML modelling languages, the proposed reference model could potentially guide business analysts, system analysts and software developers when engineering applications using blockchain technology, developing new blockchain technology platforms, analysing and comparing existing blockchain solutions.

1.1 Research Questions

The main research question (**MRQ**) is:

MRQ - How to unify the understanding of the technology, through a comprehensive reference model? This question is broken down into several sub-research-questions (**SRQ**):

SRQ1 - What is the current state of the blockchain technology and how to model the representations? We will investigate different blockchain technologies and decide which properties we will consider for analysis. Information will be gathered and modeled using known modeling notations.

[1] http://www.newsbtc.com/2016/02/27/bitcoin-continues-become-increasingly-popular/

SRQ2 - How to build the reference model? We will use the information gathered in **SRQ1** and analyze the similarities and differences between the technologies. Based on that analysis we are going to build the reference model.

SRQ3 - What are the means of validating the model? We will focus on validating the model's accuracy by comparing it to a selection of blockchain technologies. Additionally, we will use the model and the research to see if it is possible to do a security assessment on the technology.

1.2 Research Method

The following research method is applied to provide a sufficient and detailed answer to the main research question (see MRQ in Section 1.1):

1. State of the art - Discover and research the existing technologies and represent them using known modeling languages. For a top-level presentation of the technology, from an enterprise viewpoint, we have chosen ArchiMate[2] modeling language, which allows us to capture different layers of the technology and see what are the relationships between the entities. ArchiMate language also supports process and domain models, but because they can not be modeled in detail, we will use BPMN[3] to capture the processes and UML[4] to present the domain with class diagrams. Figure 1 presents a map that show our approach to modeling the technology.

2. The current solutions with their representations are analysed and compared to see what are the differences and similarities between the technologies.

3. Conceive a reference model, that would present the domain of the blockchain technology. Show how the reference model covers different use-cases (permissioned and unpermissioned, transactions only and with smart-contracts).

4. Accuracy Validation - Compare the reference model to the four blockchain technologies, that were used as the basis for building the model, and conceive a Delta (Δ) metric that would represent the difference between the reference model and the original implementation. Then choose another four blockchain technologies, compare and conceive the Delta (Δ). Compare the Deltas to see how well the reference model performs compared to existing blockchain technologies.

[2]http://www.opengroup.org/subjectareas/enterprise/archimate-overview
[3]Business Process Model and Notation - http://www.bpmn.org/
[4]Unified Modeling Language - http://www.uml.org/

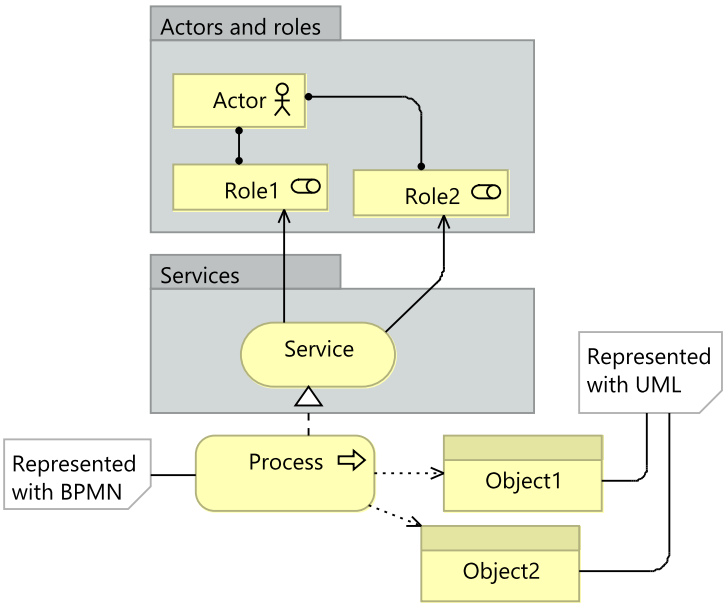

Figure 1: Map showing our approach to modeling the technology

5. Validation through application (Security assessment supported by the reference model) - We will research a set of known risks and vulnerabilities and present them using ISSRM[5]. Then link the risks and the reference model by ISSRM alignment with ArchiMate and BPMN. The goal is to provide a security assessment, supported by the model, to see which actors and components are affected by the attack.

Chapter 2 presents the research protocol and gives a state of the art overview. This is followed by the analysis of the selected blockchain technologies. Chapter 3 describes the contribution - comparison of the technologies and conceiving the reference model. Chapter 4 and 5 present our approaches to validating the model - firstly validating the accuracy of the model by considering actors, services, processes and data of the existing blockchain platforms; secondly validating the model via application by applying it to risk-related security assessment of the blockchain technology. Finally, chapter 6 gives the concluding remarks and presents future work.

[5]Information Systems Security Risk Management

2 State Of The Art

This chapter introduces the state of the art for blockchain technology and provides an answer to "What is the current state of the blockchain technology and how to model the representations?" (**SRQ1** in Section 1.1). To better answer this question, we break it down into three sub-questions: 1) What are the considered properties that we are looking for in the blockchain technology? 2) What blockchain technology implementations to select? 3) What modeling languages to use to model the representations? We will begin by explaining the blockchain technology in more detail. Research protocol is presented to show what information is expected from different implementations and which implementations were selected and how they are different from each other. After this, each implementation is looked at in more detail and for each, a conceptual representation model is provided.

2.1 Background

In the traditional banking systems, the ledger is a centralized party (the bank) which stores all the transactions. This means that all of the trust is put into that one party and there is no-one to verify that the central party can be trusted.

Blockchain acts as the distributed public ledger. It is a digital record of transactions and ownership, that is replicated among all of the participants of the peer-to-peer network and a consensus algorithm ensures that each node a the same copy of the ledger as the other nodes. Technically, it is a back-linked ordered list of blocks, where each block contains transactions [5]. Each time a transaction is made, it is broadcasted to the network and if it is valid, it will be added to a block. When a new block is published to the network, all participants (nodes) in the network will run algorithm to evaluate and verify the block. Majority of the nodes have to agree that the new block is valid and if so, it will be part of the main blockchain. Once a block of data is recorded on the blockchain ledger, data becomes *more* secure as the blockchain grows [22].

There are two main types of blockchains. Bitcoin has a *public ledger*, i.e. *public blockchain*, where anyone is allowed to read and write [24]. There is no need for a third authority to grant any permissions. On the other hand, a *private blockchain* is a network where all the participants are known and trusted [10] and the consensus process is managed by pre-selected set of participants [6].

The first generation of blockchain was all about cryptocurrency and its exchange possibilities. The 2.0 generation is about contracts. Bitcoin introduced a very basic, Turing-incomplete scripting language, that allowed some form of contractual complexity, but Ethereum came out with a full Turing-complete scripting language and a new blockchain, that focuses heavily on the use of smart contracts.

The third generation of blockchain is believed to support applications beyond currency and finance in different areas of government, health, science and culture.

2.2 Research Protocol

Research protocol presents which properties of the technology we consider for our research, which implementations of the technologies were selected and what are the limitations regarding our research.

2.2.1 Considered Properties

This research is done from a business perspective and thus we are considering the following properties:

- **Platforms** - we are considering implementations of the blockchain technology that introduce different approaches to privacy and smart contracts.
- **Actors** - we want to know who the actors are and what roles they play in the given blockchain technology.
- **Services** - what services are provided by the blockchain platform? Who interacts with the services?
- **Processes** - what are the underlying processes to services? How do network, transaction and mining/consensus processes work?
- **Data models** - what are the entities that hold information? What are the relationships between them?

2.2.2 Selected Blockchain Platforms

Blockchain technology platforms can be separated into four groups [3] [17] as illustrated in Table 1. For our study we have selected one blockchain platform of each group. They can be characterised as follows:

Table 1: Overview of chosen blockchain technologies

	Permissionless	**Permissioned**
With Smart Contracts	Ethereum	Chain Core
Transactions only	Bitcoin	MultiChain

- **Permissionless** - Fully public blockchains, where anyone can read and write.
- **Permissioned blockchain** allows to define different permissions on different users on the network. There can be different permissions for different operations on the blockchain.

- **Blockchains with Smart Contracts** enable “smart contract" like capabilities and allow building business logic and business process mechanism into the chain.
- **Blockchains with transactions only** are built for transaction capabilities. They support transferring value from one account to another.

2.2.3 Scope

Our scope includes the selected platforms and we will be modeling them based on the considered properties mentioned in the previous sub-sections. ArchiMate modeling language supports three layers (Business, Application and Infrastructure layer), but we will only focus on the Business layer. The business processes in the Business layer are expanded with BPMN and the data objects are expanded by UML, to provide a more detailed overview.

2.2.4 Limitations

Even though the technology has been around for a while now, there are limitations when it comes to the literature in hand. As mentioned in the introduction, there are very few academic studies available regarding blockchain and its representation. Most of the available non-academic papers about blockchain technology are whitepapers, documentations or technical documents; some of which are still being updated and may be incomplete [14] [2]. Due to this, some information is missing or incomplete, so the following review of the technologies is based on the literature available to the author (any absence of information is noted in the discussion).

2.3 Bitcoin

2.3.1 Introduction to Bitcoin

Bitcoin came to life with the publishing of a paper called “Bitcoin: A Peer-to-Peer Electronic Cash System” by an anonymous author or group called Satoshi Nakamoto [21]. Bitcoin[6] is a collection of concepts and technologies that form the basis of a digital currency ecosystem [5]. It is a distributed and decentralised network, where users communicate with each other via peer-to-peer. This means there is no central authority to handle all the transactions between users. In the Bitcoin network, bitcoins are the units of currency, that are used to store value among participants. The Bitcoin protocol is open-source and can run on

[6]Bitcoin (with upper B) stands for protocol, the software and community, bitcoin (with lower b) stands for a unit of currency

laptops, smartphones, tablets and others. Having the possibility to adopt so many platforms is making the technology very convenient for the consumers.

Users can use bitcoins to do anything that they would normally do with a more conventional currency, meaning that it can be used for buying and selling products, sending and receiving payments. There are several special currency exchange ATMs (Automated Teller Machines) that accept bitcoins[7] as a form of currency and there users can buy, sell or exchange bitcoins.

2.3.2 Business Layer

This section will give an overview of the business layer for Bitcoin, presented in Figure 2. It consists of six major components - Actors and roles, Services and four processes: Network Discovery process, Transaction Creation process, Block validation process and Mining process.

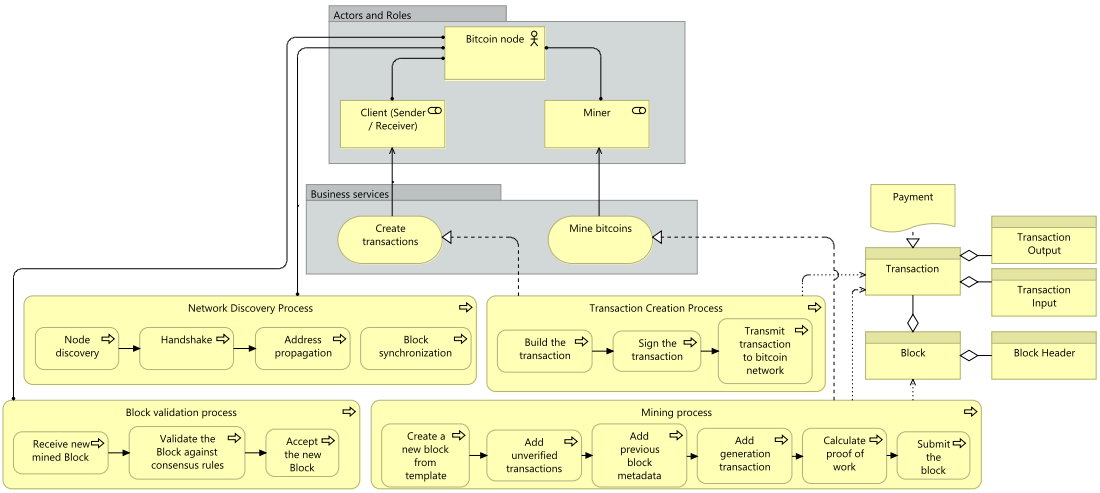

Figure 2: Archimate business layer for Bitcoin

Actors and roles: In the blockchain context, each node is an actor in the network, but actors perform different operations. Based on that, two roles are defined:

- Client - user of the technology that wants to send or receive bitcoins. To do so, a transaction is created, signed and broadcasted to the network.
- Miner is an actor who mines bitcoins. This means validation of transactions, generating blocks and submitting them to the network, to be included in the

[7]http://www.coindesk.com/bitcoin-atm-map/

blockchain. When submitting a valid block, miner is rewarded with bitcoins (for the work done on the proof-of-work algorithm).

Services: There are two main services displayed on Figure 2. Transaction creation service allows the client to create a new transaction and send it to the Bitcoin network (to transfer bitcoins to another user). The Miner will use Bitcoin's Mining services to be able to mine bitcoins.

Processes: There are 4 sets of processes displayed on Figure 2: **Network discovery process**, **Transaction Creation process**, **Block validation process** and **Mining process**.

The first process that every node in the network interacts with in the beginning is the **Network discovery process**, seen on Figure 3. This process is split into 4 sub-processes:

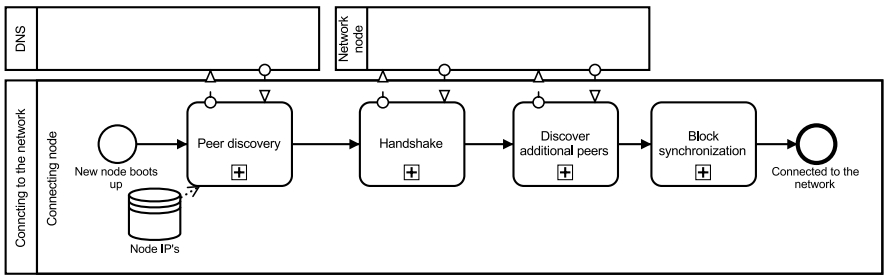

Figure 3: Network discovery process

- Finding known Bitcoin peers (Figure 4) - how a new node can find known peers IP addresses to connect to the network. New node will either get a known IP from unknown 3rd source, another person for example, or it can query the DNS seed list for known IPs. Once node has IPs, it can connect to a network.
- Handshake process (Figure 5) - Once connection with the known IP address is established, the new node has to transmit a version message to the existing node, in order to verify that they are running the same version of the software. The existing node can either choose to respond to the request or not. In the positive case, existing node will reply with its version message and a verack message. Verack message is sent to acknowledge that the peer is willing to connect [5].
- Discover additional peers - Once the new node receives the confirmation, it sends its IP address for other nodes to be propagated around the network

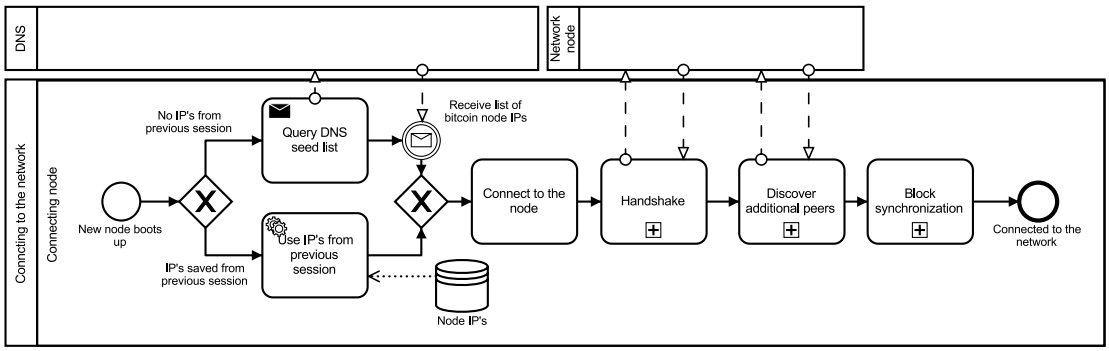

Figure 4: Expanded peer finding process in the Network discovery process

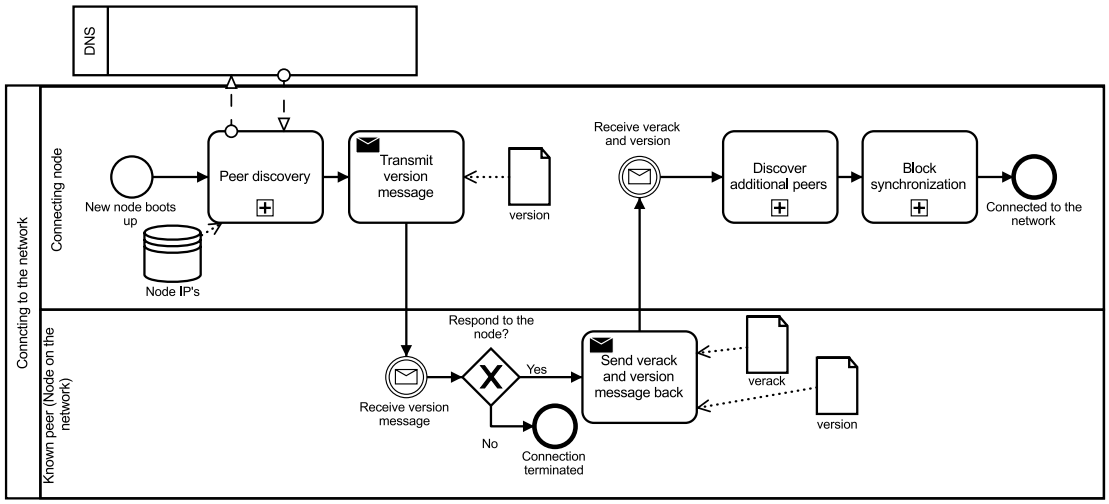

Figure 5: Expanded handshake process in the Network discovery process

and/or it can ask the neighbouring node for IP addresses of other nodes, to establish further connections.

- Block synchronization - When first connecting to the blockchain and before the new node can start validating transactions and newly mined blocks, it has to download and validate the entire blockchain from the very first block. The new node will download the longest chain based on what its neighbours have (in the handshake process, nodes exchange information about the latest block). When connecting to the network next time, node will synchronise all the blocks that have been added since last connection [2].

Transaction Creation process (Figure 7) describes creating transactions and transmitting them to the Bitcoin network. Process begins with the Client

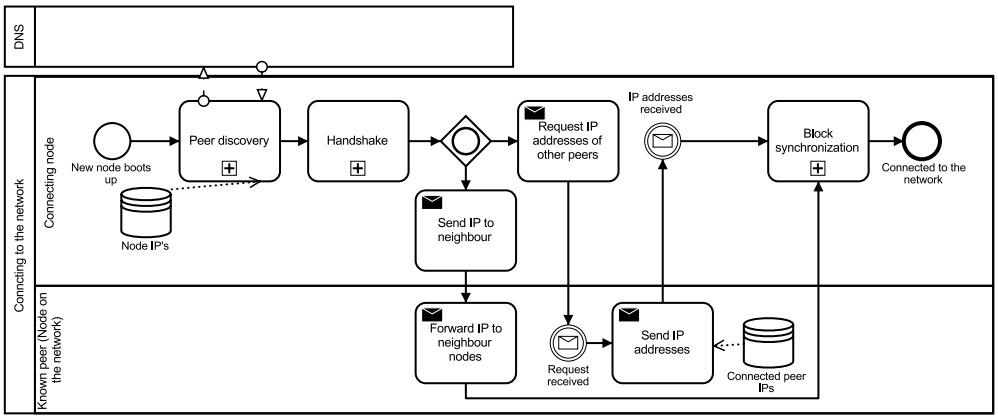

Figure 6: Expanded additional peer discovery process in the Network discovery process

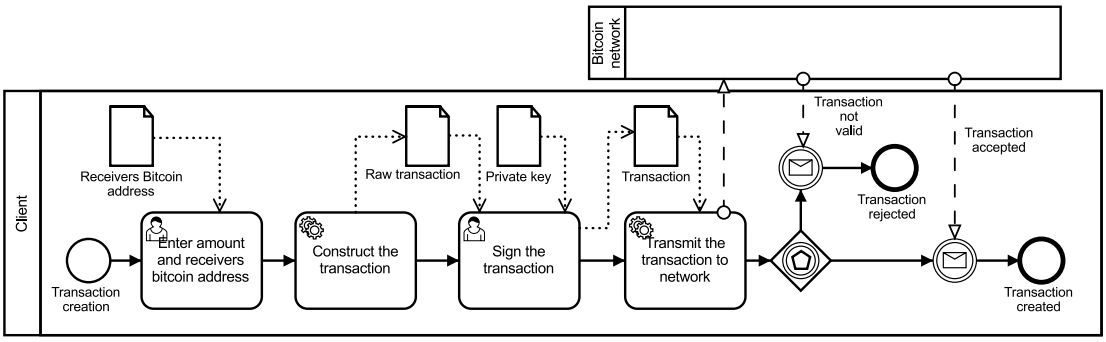

Figure 7: Process of creating a transaction in Bitcoin

specifying the amount of bitcoins to send and the receivers Bitcoin address. Then the transaction is constructed and the client will sign the transaction with a private key. After signing, the transaction is transmitted to the neighbouring nodes, who will validate the transaction [5]. If the transaction is valid, they will propagate the transaction to other neighbouring nodes who repeat the validation process. Additionally, node will keep the transaction in the memory (called unverified transaction pool). Otherwise, if the transaction is invalid, the neighbouring node will not propagate it forward.

Mining process on Figure 8 explains the generation of a new block and submitting it to the network to be included in the global blockchain. New block is triggered when the previous block has been mined, broadcasted to the network and valid. Miner first creates a new block from a template, collects unverified transactions, adds the hash value from the previous block and a generation trans-

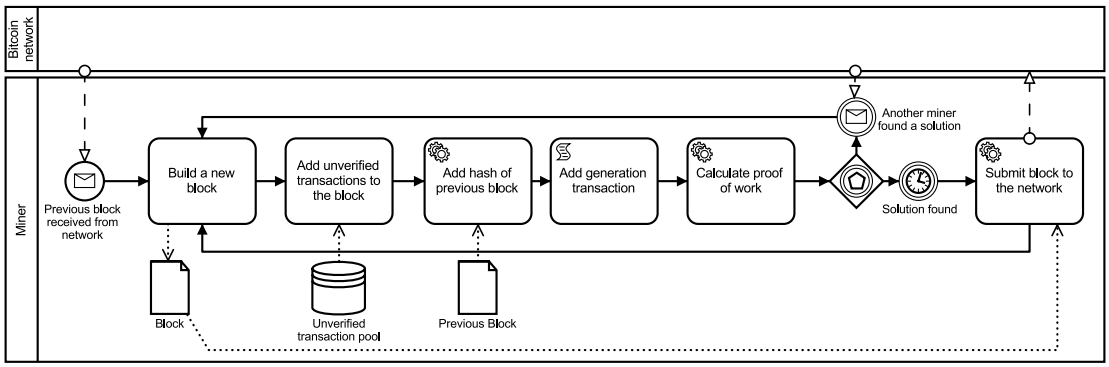

Figure 8: Process of mining bitcoins - creating a block, validating transactions and providing the proof for the block

action, which will reward the miner on creating a successful block. Then the race for calculating the proof-of-work starts. Whichever miner is the first to find the solution and broadcast the new block to the network (to be added to the global blockchain), will receive the reward for that block. Once a new block is broadcasted on the network, other miners will stop mining the current block and begin working on a new one.

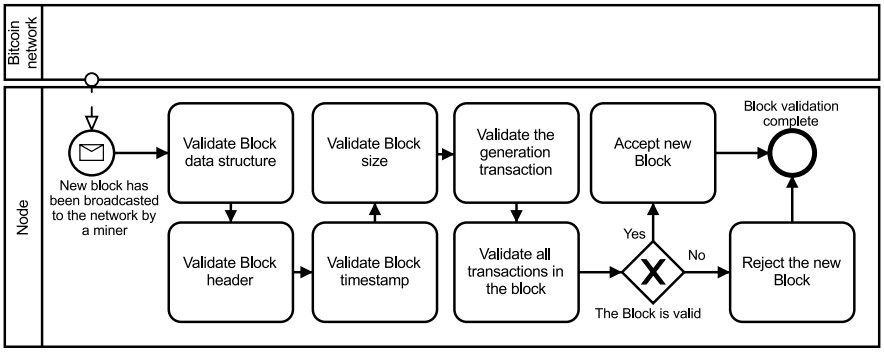

Figure 9: Block validation process

When the new block is submitted to the network by the miner, it is not added to the blockchain right away. Nodes in the network will know when a new block has been mined, and then each of them will validate the block according to the consensus rules (see Figure 9). If the block is valid, they will add the new block to their local blockchain. This independent validation ensures the consensus in the network, because everyone is validating based on the same rules. The consensus rules in Bitcoin [5]:

1. Block data structure validation
2. Block Header validation
3. Block timestamp validation
4. Block size validation
5. First (and only the first) transactions is a generation transaction, that will reward the miner
6. Validation of all the transactions included in the block

Data models: Bitcoin implementation has two main data objects: Transaction and Block (see Figure 10).

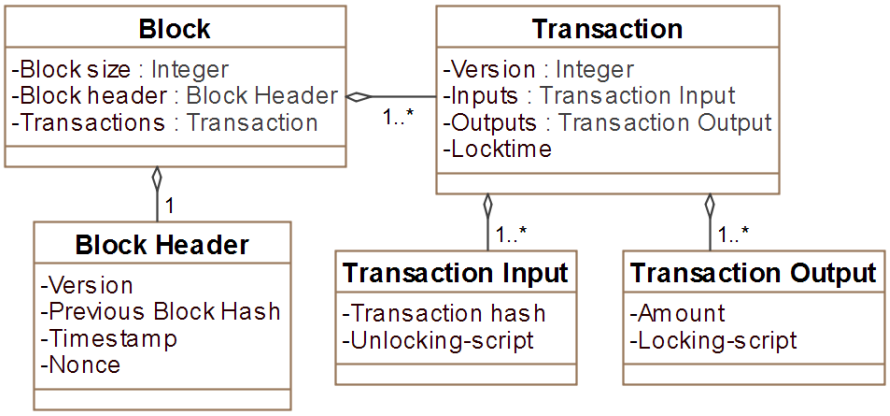

Figure 10: Bitcoin domain model

Block is a container data structure that aggregates the transactions that are to be included in the blockchain. The block consists of a header, containing metadata (a reference to a previous block hash, which connects the given block to the previous block in the blockchain, the difficulty of the block, timestamp, nonce) and a list of transactions [5].

Transaction is a representation of a payment in the system. Transaction contains a transfer of value from a source of funds, called a transaction input, to a destination, called a transaction output [5].

Every bitcoin transaction creates outputs, which are recorded on the blockchain as unspent transaction outputs (UTXO) and are recognized by the whole network. Sending bitcoin means creating an UTXO registered to receivers address and available for them to spend [5].

Transaction inputs are pointers to UTXO. They point to a specific UTXO by reference to the transaction hash and sequence number where the UTXO is recorded in the blockchain. To spend UTXO, the receiver has to prove ownership

of the Bitcoin address[8], which means only the person who was meant to receive the bitcoins, can spend them.

2.4 MultiChain

2.4.1 Introduction to MultiChain

MultiChain is a platform that allows creation and deployment of private blockchains. MultiChain introduced a number of shortcomings with Bitcoin's solution [14]:

- Scalability and cost:
  - Limited capacity - Bitcoin blockchain supports roughly the same amount of transaction per day, as big financial services, like Visa, process in couple of minutes. Maximum block size in Bitcoin can be increased, which will increase the number of transactions per day, but also the costs, because miners are required to do more work.
  - Transaction costs - Transaction fees are gathered by miners, but Bitcoin uses prioritization, where transactions with greater fees get processes faster, meaning delays for smaller transactions.
  - Irrelevant data - Full Bitcoin node has to download the entire blockchain and verify it from the beginning. This takes time and space and might be irrelevant to certain institutions.
- Privacy and security:
  - Mining risks - Bitcoin's proof of work is sufficient for general purpose, but for institutional use the unpredictable delay in the transactions or the potential of 51% attack is considered as a risk.
  - Lack of privacy and openness - Bitcoin's blockchain is completely open and viewable from the internet. Every transaction can be traced back to the first one and linked to a certain Bitcoin address. In addition, it is claimed that Bitcoin is an attractive network for illegal transactions, since Know Your Customer can not be forced on the network level.

MultiChain aims to solve theses issues via integrated management of user permissions. The core aim is following: 1) blockchain's activity is only visible to known participants, 2) introduce controls to permit transactions, 3) secure and inexpensive mining.

2.4.2 Business Layer

Business layer for MultiChain is presented in Figure 11. Consists of five major components discovered from the literature: Actors and Roles, Services, Network Discovery process, Transaction creation process and Mining process.

[8]Unlock the Locking-script using the private key corresponding to Bitcoin address

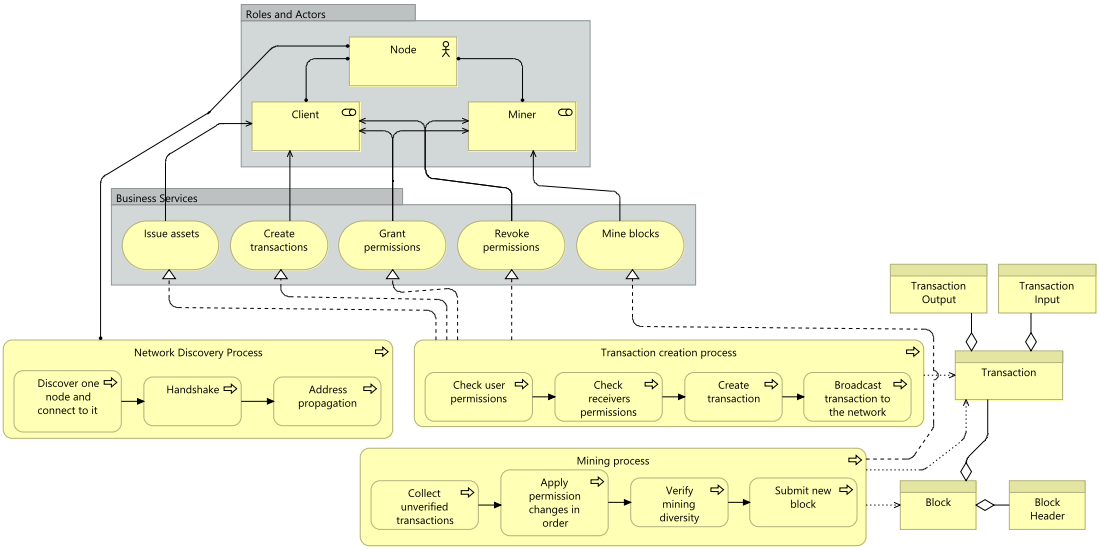

Figure 11: MultiChain business layer

Actors: We have identified 2 main actors for MultiChain:

- Client - can issue assets and send a quantity of assets to another user, create streams and publish data to stream or grant and revoke permissions if the client is an administrator on the chain. Client can perform all of these operations via transactions.
- Miner - can mine blocks. Miner of the first block in the blockchain, known as the "genesis" block, will be granted administrative privileges. From there, miner can also grant access and privileges to users.

Services: There are total of 5 identified main services:

1. Issue assets - create new assets
2. Create transactions - create transactions to transfers assets, create streams[9], privilege management and other.
3. Grant privileges - there are total of 8 permissions that can be granted: Connect, send, receive, issue, create, mine, activate and admin.
4. Revoke privileges - remove privileges from users
5. Mine blocks - perform block mining

Processes: MultiChain presents three processes that were discovered from the documentation[10] [14]: **Network Discovery Process**, **Transaction creation**

[9]http://www.multichain.com/developers/data-streams/
[10]http://www.multichain.com/developers/

process and **Mining process**.

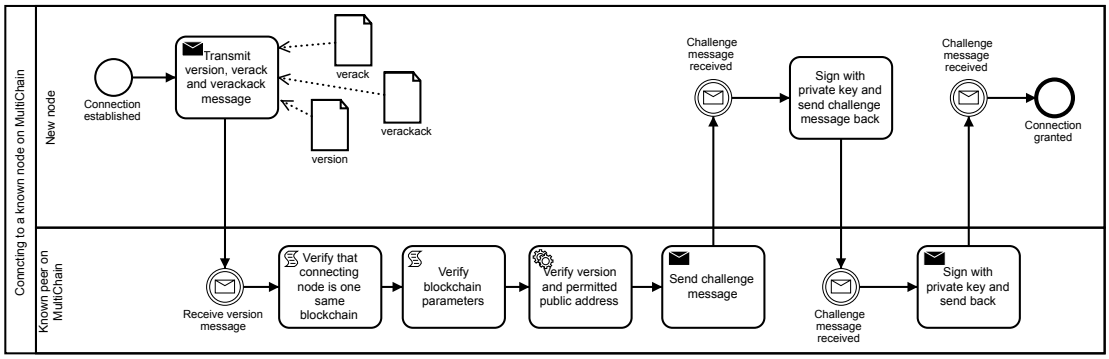

Figure 12: Handshake process in MultiChain

Network discovery process is similar to the one provided by Bitcoin, but since MultiChain provides permissions, the "handshake" process is different with the inclusion of a *verackack* message (see on Figure 12). The new MultiChain node has to get a name, IP and port number of the blockchain it wants to connect to (finding known peers). First step in the handshake process is to verify that two connecting nodes are on a blockchain with the same name and using same blockchain parameters. Next, each node presents its identity as a public address, that has permissions to connect to the blockchain, and the corresponding private key. If the node does not have permissions to connect, node's address should be presented to the administrator who can grant him the permission to connect. Then each node sends a challenge message to the other party and each node sends back a signature of the challenge message, proving their ownership of the private key corresponding to the public address they presented.

Issuing and sending quantities of assets, create streams and publishing data to streams, granting and revoking privileges - all of these can be performed by creating a transaction, that contains specific metadata. **Transaction creation process** is described in detail on Figure 13.

When creating a new blockchain, the miner of the first "genesis" block automatically receives all privileges, including administrator rights to manage the privileges of other users. Admin can then grant privileges to other users by sending transactions, which contain users addresses and metadata, which lists granted permissions. Permissions can be granted permanently or temporarily, by specifying the start block and end block.

When creating a regular transaction, that transfers funds from one account to another, the transaction is valid if the senders address has a 'send' permission and the receivers address has 'receive' permission. If one or the other is missing

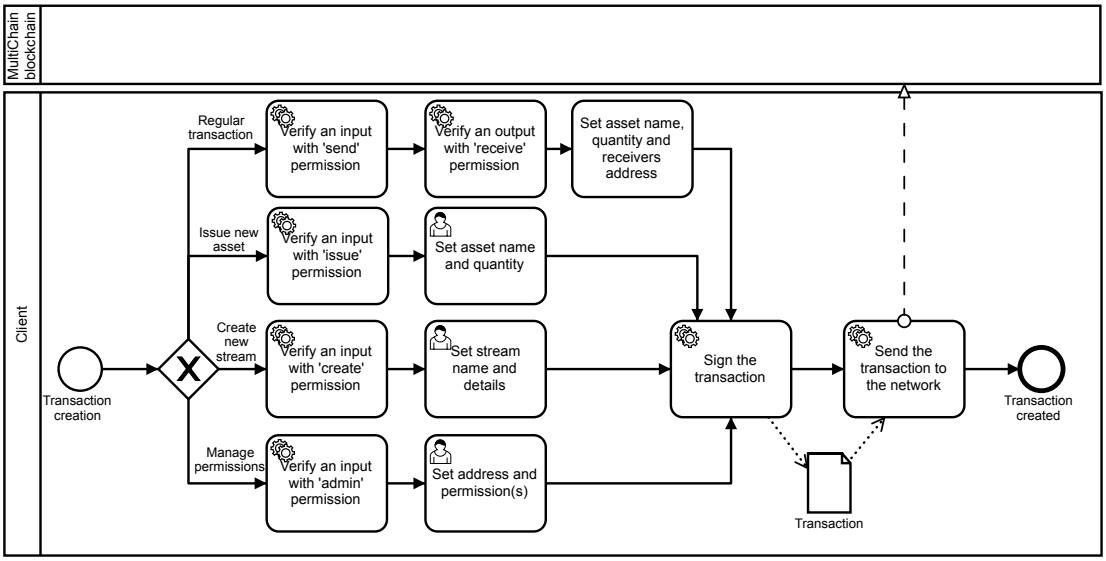

Figure 13: Creating a transaction in MultiChain

a permission, the transactions is invalid. When creating a new asset or a new stream, transaction must include an input that has been signed with an address with 'issue' or 'create' permission respectively.

In the **mining process** (Figure 14), 'mine' permissions are required to mine blocks and in order to receive a fee (if one is set), miner must also have 'receive' permission.

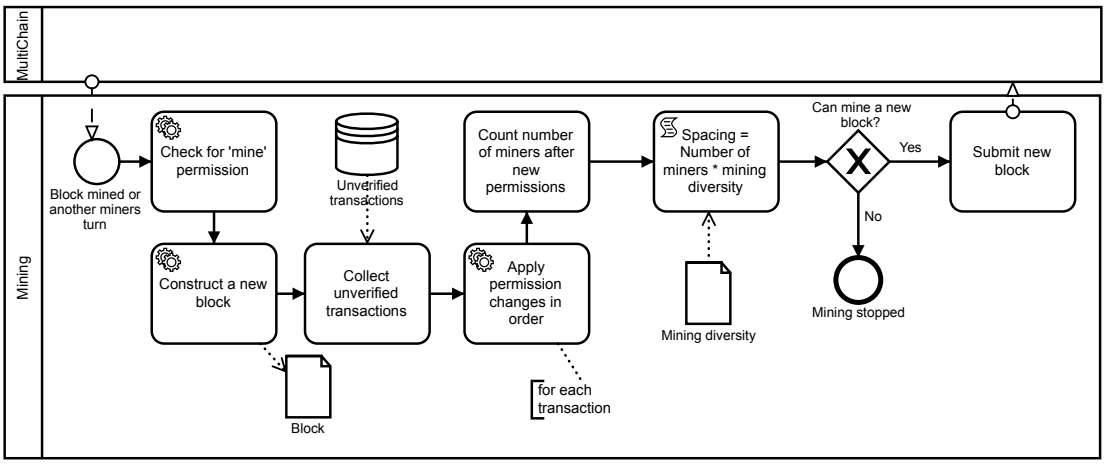

Figure 14: MultiChain mining process

MultiChain forces a mining scheme, where permitted miners must mine new blocks in rotation. Depending on the *mining diversity* parameter, strictness of mining can be configured. This helps to ensure that no single miner will dominate mining on the given blockchain.

Validity of a transaction is confirmed if the total quantities of all the assets in the outputs are strictly equal to the total in transaction inputs and depending on the type of transaction, an input has to exist that is signed with an address with certain permissions, also shown on Figure 13.

The validity of the block is verified in four steps [14]:

1. Apply all the permissions changes defined by transactions in the block in order.
2. Count the number of permitted miners who are defined after applying those changes.
3. Multiply miners by mining diversity, rounding up to get spacing.
4. If the miner of this block mined one of the previous spacing-1 blocks, the block is invalid.

Data models: Documentation for MultiChain lacks concrete descriptions for main entities, but based on [14], MultiChain uses the same Block and Transaction model described in Bitcoin. This model is also depicted on the business layer Figure 11.

2.5 Ethereum

2.5.1 Introduction to Ethereum

Ethereum is a project which attempts to build the generalised technology - technology, on which all transaction based state machine concepts may be built [27]. It is a Turing-complete contract processing and execution platform based on a blockchain ledger. It is a completely independent design and implementation compared to Bitcoin [5]. This system can be said to be a very specialised version of a cryptographically secure, transaction-based state machine [27].

Ethereum does this by building what is essentially the ultimate abstract foundational layer: a blockchain with a built-in Turing-complete programming language, allowing anyone to write smart contracts and decentralized applications where they can create their own arbitrary rules for ownership, transaction formats and state transition functions [12].

The Ethereum blockchain can be alternately described as a blockchain with a built-in programming language, or as a consensus-based globally executed virtual machine. The part of the protocol that actually handles internal state and computation is referred to as the Ethereum Virtual Machine (EVM). From a practical

standpoint, the EVM can be thought of as a large decentralized computer containing millions of objects, called “accounts”, which have the ability to maintain an internal database, execute code and talk to each other.

2.5.2 Business Layer

The business layer for Ethereum is presented on Figure 15. Consists of six major components - Actors and Roles, Services, Network discovery process, Transaction creation, Block validation and Mining process.

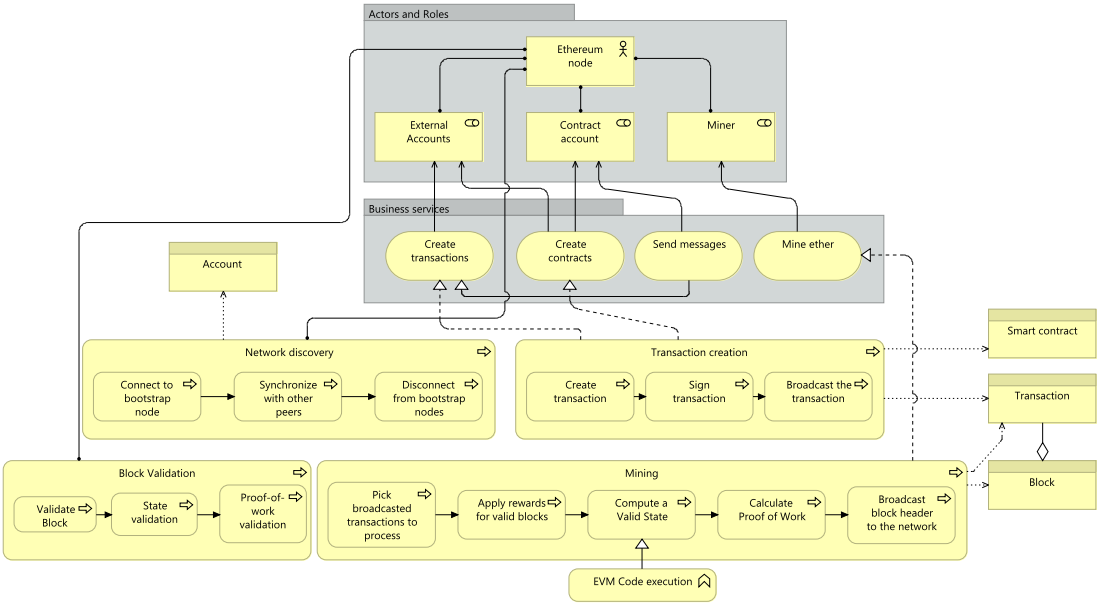

Figure 15: Ethereum business layer

Actors: There are total of three actors in Ethereum [12] presented in Figure 15:

- Externally owned user accounts (EOA), which are controlled by private keys. This actor can create transactions to transfer value, create smart-contracts or call contract functions.
- Contract accounts (CA), which are controlled by their code. Every time it receives a message, its code executes, allowing it to read and write to internal storage and send messages to other contracts or create contracts in return.
- Miners validate transactions and also the smart-contract code execution. The transactions are wrapped in a block and proof-of-work will be provided for the block.

Services: Figure 15 shows four services Ethereum actors interact with: 1) Create transactions 2) Create contracts 3) Send messages 4) Mine ether.

Processes: Figure 15 shows four processes. The processes, **Block validation**, for validating blocks, **Network discovery**, which is necessary for a new node to join the network, **Transaction creation**, which allows user to create transaction or contracts and allows contracts to create transactions and messages, and **Mining**, which describes the mining process and broadcasting a new block to the network.

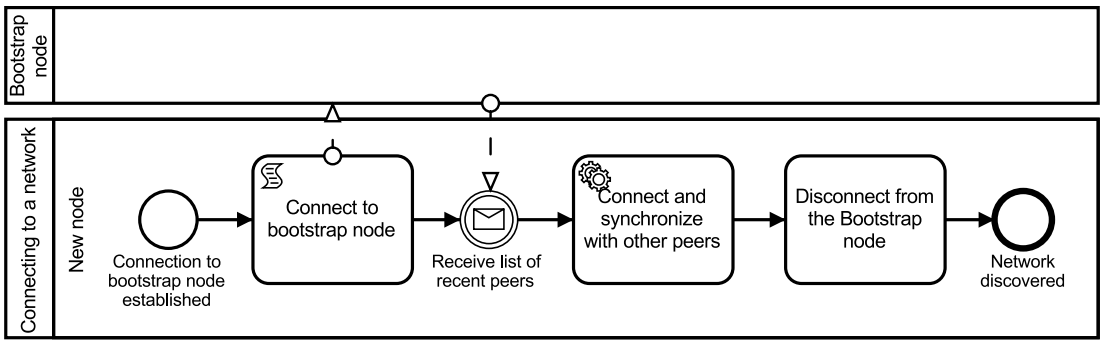

Figure 16: Network discovery in Ethereum

The **Network discovery process**, seen on Figure 16, begins by establishing a connection to a bootstrap node (a node which IP comes from the source code, is assumed to be always online and is connected to other regular nodes). Once connected, the bootstrap node will share IPs of peers connected to it and the new node will synchronize with the other nodes. If the node is connected to other nodes as well, there is a possibility to prune the bootstrap node, leaving only connection to other regular nodes.

Transaction creation process (Figure 17) is complex in Ethereum, since it supports smart contracts. Initially, EOA has 2 choices (because contract functions can not be called if there are no contracts), either to send a standard transaction to another EOA (transfer ether) or create a new contract. When a contract has been created, EOA can create new transaction that can call the functions of the smart contract, described in the data of the transaction.

Contract accounts can send messages. Messages are essentially like transactions, difference being that it is produces by a CA. CA-s can either send ether to an address that is stored in the internal storage of the contract, or create and send messages to other contracts.

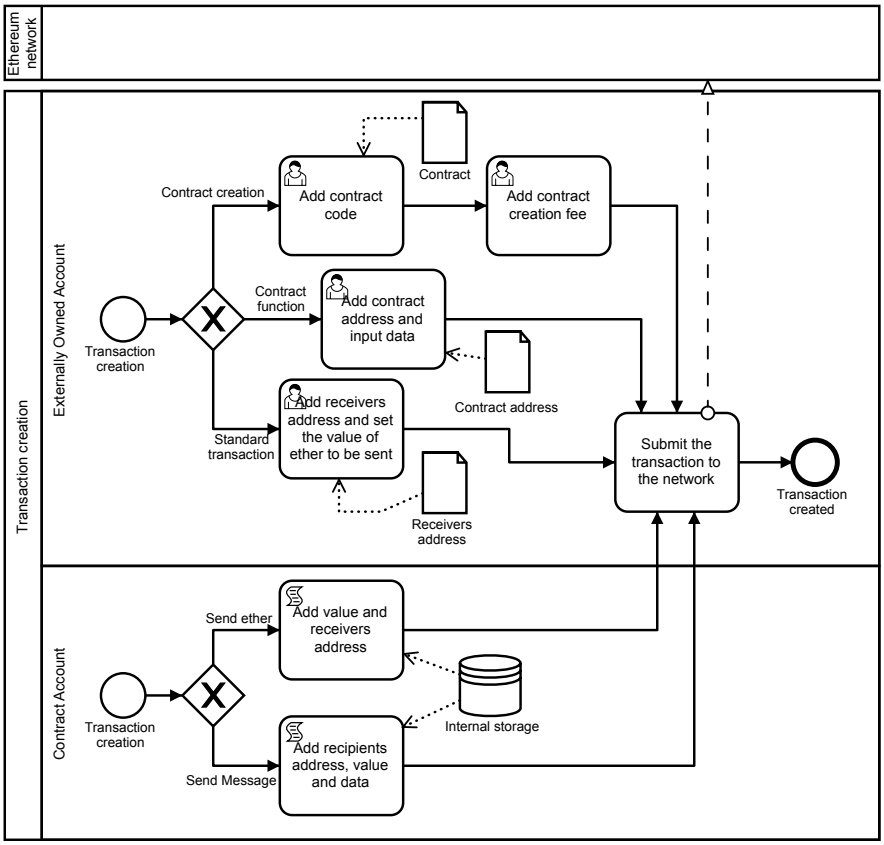

Figure 17: Creating a transaction in ehtereum

Before taking a look at mining in Ethereum, we need to understand the concept of state transition (see Figure 18). Ethereum keeps a state and transaction alters the state. If there are no errors in altering the state, it becomes the new state in the next block. The state transition process consists of following steps:

1. Verify the structure of the transaction (signature and nonce)
2. Verify that the sender has enough balance to cover the transaction fee
3. Initialize GAS with the given input amount of START GAS
4. Transfer the value from sender to receiver. If receiver EOA does not exist, one is created. If the target was CA, code is executed until finished or until it runs out of gas. Contracts can Read or Write to its internal storage, read the message data or create a new message that is sent to another contract or to an EOA.
5. If transfer fails because sender EOA did not have sufficient balance or code execution runs out of GAS, all changes are reverted, except transaction fees,

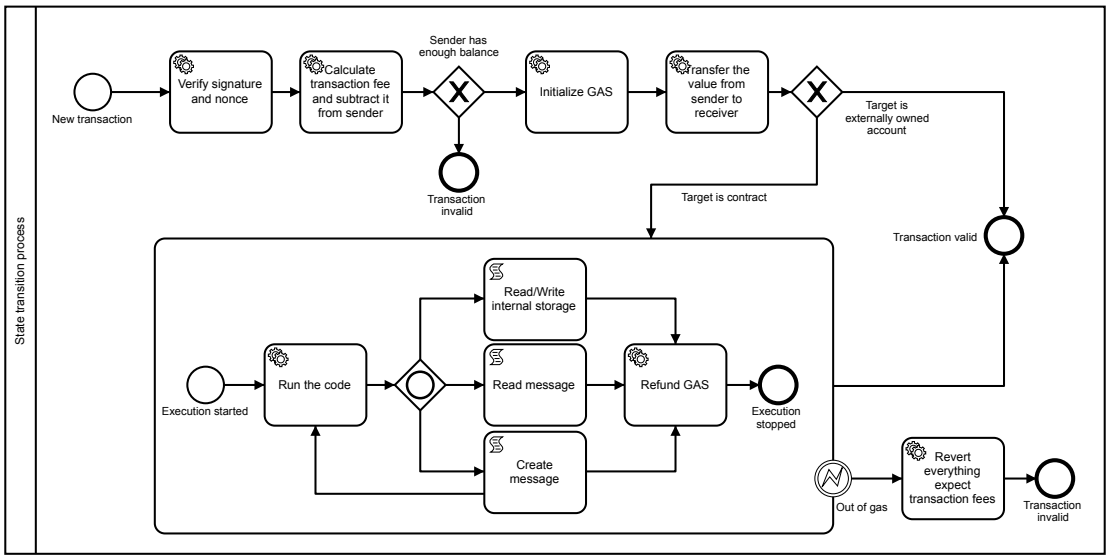

Figure 18: Ethereum state transition process

which are transferred to the miner.

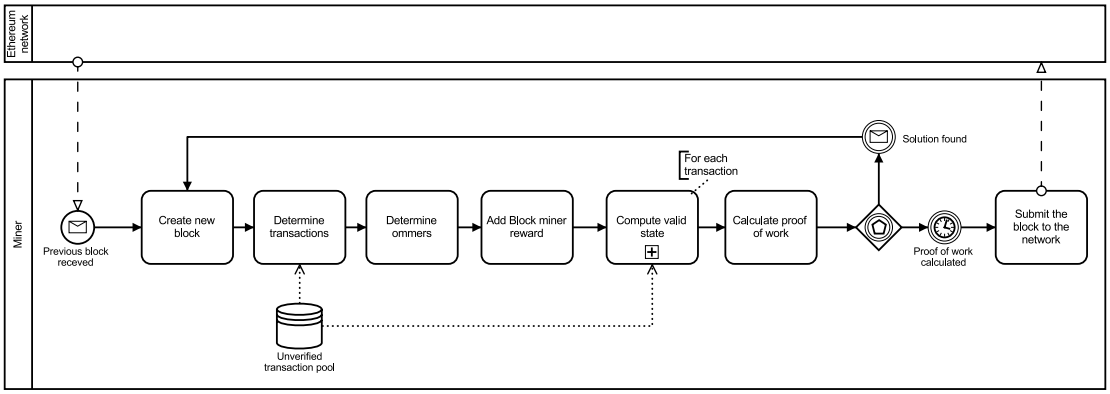

Figure 19: Ethereum mining process

The **mining process** on Figure 19 begins when a miner detects that new transactions are broadcasted to the network. Then the miner will find ommers, which are valid blocks, but not part of the main chain, and include them into the block header. Then the miner will compute a valid state, which means validating the transactions and messages, executing code and checking that the execution does not run out of gas (expanded process can be seen on Figure 18). If a valid state is computed, miner also has to provide the proof-of-work. After all this,

miner can broadcast the new block to the network. If a new block was already broadcasted, miner will move on to creating the next block.

Block validation process is described in Figure 20. All of the block's components are validated, including the proof-of-work and state validation (by executing the transactions). This is done by every full node in the network before they add the new block to their blockchain.

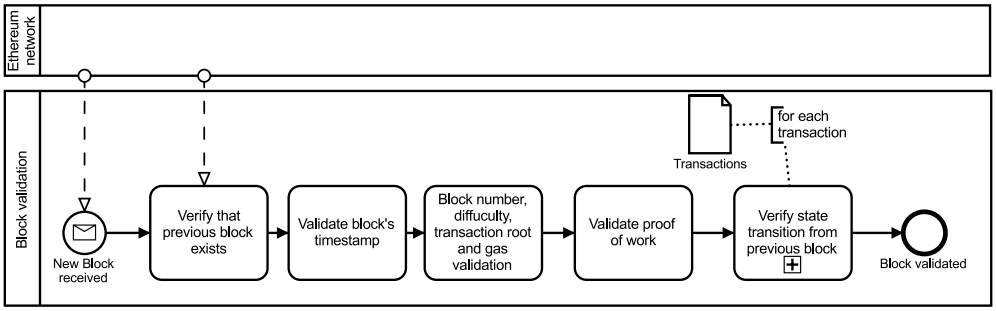

Figure 20: Block validation in Ethereum

Data models: The Data model is presented in Figure 21. Account contains a nonce, which is equal to the number of transactions sent from given account [27] and the balance of ether. If the account is a CA, there is also the contract code and storage for the contract.

Transaction in Ethereum is a signed package of data, that stores a message that is to be sent from an EOA. The receiver can either be an EOA or CA. Simplified, the transaction contains the address of the recipient, signature of the sender, from which the address can be extracted, amount of ether, data (in case creating a contract or calling a function) and start gas and gas price.

Block contains a list of transactions. In addition, block holds a block header, which contains the most information - hash of the parent block, block number, gas limit, gas used for the given block, timestamp and nonce.

Receipt is an entity that keeps information about the post transaction state, the total gas used creating the block and logs of the transactions. Contracts have the ability to write to logs and they can be observed later if needed.

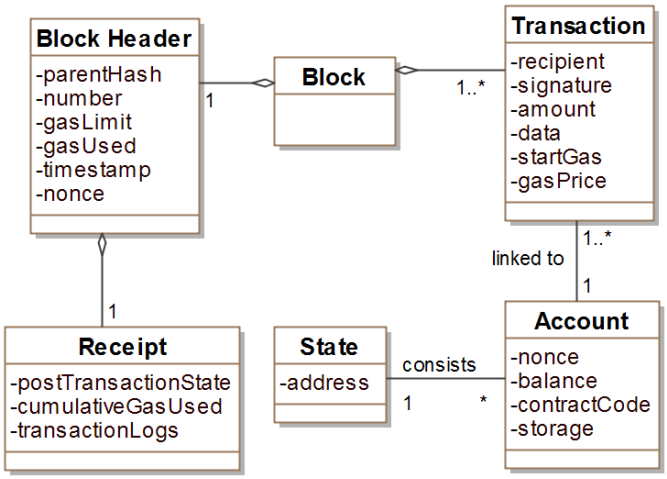

Figure 21: Ethereum domain model

2.6 Chain Core (Chain Protocol)

2.6.1 Introduction to Chain Core

Chain Protocol [8] is a design for a shared, multi-asset, cryptographic ledger. It supports the coexistence and interoperability of multiple independent networks, with different operators, sharing a common format and capabilities.

The Chain Protocol allows any network participant to define and issue assets by writing custom "issuance programs". Once issued, units of an asset are controlled by "control programs". These programs are expressed in a flexible and Turing-complete programming language that can be used to build sophisticated smart contracts.

Each network is secured by a federation of "block signers". The system is secure against forks as long as a quorum of block signers follows the protocol. For efficiency, block creation is delegated to a single "block generator". Any node on the network can validate blocks and submit transactions to the network.

Chain Core is an enterprise software product that implements the Chain Protocol and is used as the basis for modeling.

2.6.2 Business Layer

The business layer for Chain Core is presented in Figure 22. It features five major components: Actors and Roles, Services, Transaction process, Consensus process and Network discovery process.

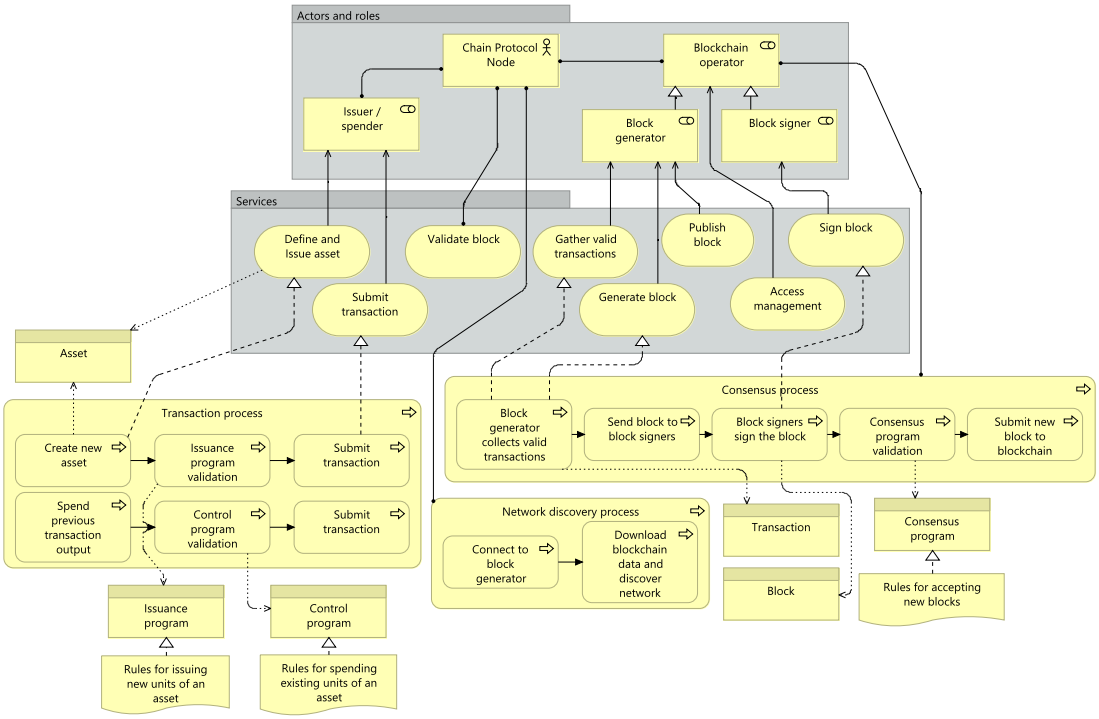

Figure 22: Chain business layer.

Actors: Clients/Users in Chain are issuer and spender. Issuers can issue assets (they write an "issuance program"), spenders can spend transaction outputs (if "control programs" allow them to). Blockchain operators are split into two roles as well - Block generator (one of the operators will be the designated block generator) and block signers. Block operators perform four basic tasks - Access management, Gather valid transactions from participants, Generate and sign blocks of valid transactions, Distribute blocks to participants

Services: Define and Issue new assets. Submit transactions. Gather valid transactions. Generate block. Access management. Publish block. Sign block.

Processes: **Network discovery process** (Figure 23): When a new blockchain is created, the Block Generator has created a network token which will be distributed to participants that want to connect to the blockchain. Participant has to also provide block generator URL and blockchain ID. If Block Generator receives a request to connect with a specified network token, access is granted and the participant will begin downloading the latest blockchain data from the Block

Generator.

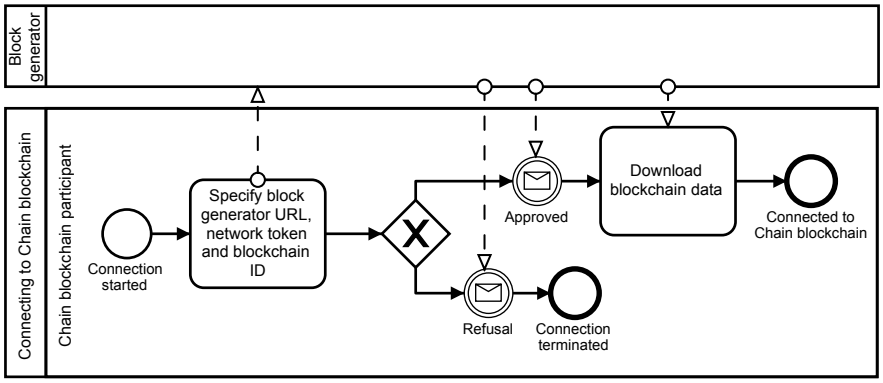

Figure 23: Connecting to Chain blockchain.

Transaction process (Figure 22) - Creating a transactions means to either issue a new asset or spend an existing asset. When issuing a new asset, it must comply with the issuance program defined on the blockchain. If valid, transaction can be submitted to the network. When spending an existing asset, it must comply with the control program, which specifies what is required to spend an asset. For example, spending an asset X might require a signature from the asset issuer.

Consensus process (seen on Figure 24) is carried out by two roles:

- Block Generator
  1. Accept transactions from participants in the network and collect them
  2. Periodically validate the transactions and generate a Block of valid transactions
  3. Sign the Block
  4. Send the Block to Block Signers for signatures
  5. Distribute the Block to the network
- Block Signer
  1. Accept a Block from Block Generator
  2. Validate the Block
  3. Validate each Transaction in the Block
  4. Sign the Block
  5. Send the signed Block back to Block Generator

Data models: The Data model is presented in Figure 25. The purpose of a Chain blockchain network is to manage issuance, ownership, and control of digital assets [8]. Assets are issued, transferred, and exchanged by posting transactions

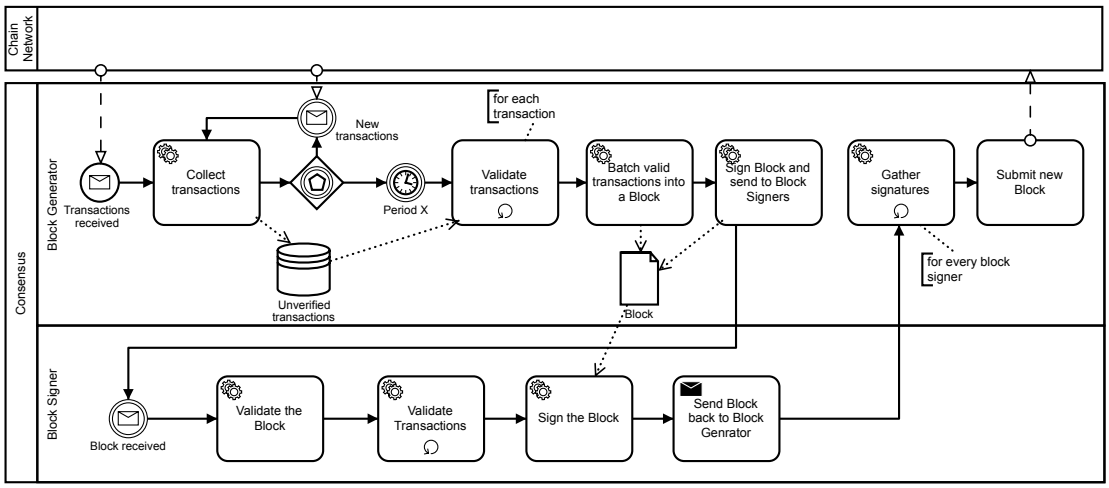

Figure 24: Chain consensus process.

to the network. These transactions are ordered and batched into blocks, which together form an immutable blockchain.

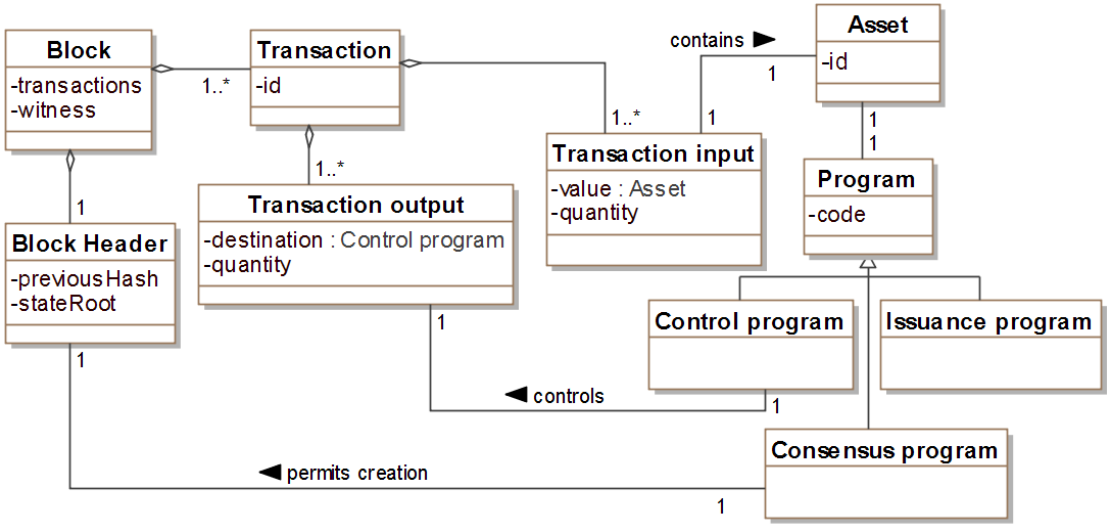

Figure 25: Chain domain model

Multiple types of assets are supported by a single chain [8]. Each asset has an unique asset ID, which corresponds to a issuance program, that defines the rules for issuing units of the given asset. Once units have been issued, the rules for spending them are determined by control programs.

Transaction has inputs and outputs [8]. Input specifies a value - either issuance of new units or output from previous transactions. Output specifies a destination - a control program that defines the rules of spending them in the future. Each input and output specifies a quantity of a single asset ID. Each input must satisfy an issuance program or a control program. The issuer or spender may pass arguments to the program via the witness field. Each transaction, and each individual input and output, includes a reference data field for arbitrary application-level uses.

Each block header contains the hash of the previous block. A block contains the hash of all its transactions and the hash of the current state - the set of current unspent outputs. To prevent unauthorized participants from creating new blocks, each new block must satisfy a consensus program, which is specified in the header of the previous block [8].

2.7 Answers to Research Questions

In this chapter we defined the research protocol and reviewed the state of the art to answer the research question "What is the current state of the blockchain technology and how to model the representations?" **SRQ1** in Section 1.1. For a more convenient answer, this question was broken down into three sub-questions.

What are the considered properties that we are looking for in the blockchain technology? - We defined four properties: Actors, Services, Processes and Data models.

What blockchain technology implementations to select? - We reviewed four distinct platforms: Bitcoin, MultiChain, Ethereum and Chain core. These platforms were chosen because each of them brings something new to the blockchain technology. Bitcoin and Ethereum are examples of public blockchains, while Bitcoin being transaction based and Ethereum being smart contract optimised blockchain. Opposed to public blockchains, MultiChain and Chain Core provide private blockchains, where participants are known and they are controlled by admin or a group of admins on the blockchain, so at all times it is known who has access to the blockchain and what operations they can perform.

What modeling languages to use to model the representations? - These four platforms were modeled, according to the properties, with three different modeling languages: ArchiMate for top-level view of the technology, BPMN for detailed process models and UML for detailed class diagrams.

Having such a variety of public / private and transaction based / smart contract based platforms, we can see what are the differences in services and processes, that allow for such kind of solutions for the blockchain technology. The next section will provide comparison of the proeprties and aims to build a conceptual reference model to represent the blockchain technology.

3 Contribution

This chapter describes my contribution to this thesis and provides answers for research question "How to build the reference model?" (**SRQ2** in Section 1.1). To help us answer this questions, we have broken it down into following sub-questions: 1) What are the differences and similarities between the considered properties? 2) What are the criteria for including an entity in the reference model? First, a comparison between the technologies is presented (based on the state of the art review done in Chapter 2), followed by a discussion on what components and entities are included in the reference model.

3.1 Technology Comparison

In the research protocol (see Section 2.2.1), four key elements were defined that were looked into regarding the selected technologies: Platforms, Actors, Services and Process. The chosen platforms are described in previous section: Bitcoin, MultiChain, Ethereum and Chain Core. For this section, the aim is to analyse Actors, Services, Processes and Data models.

3.1.1 Actors

As mentioned, Blockchain technology relies on a decentralised network of individual nodes - but nodes have different purposes and different roles regarding the ecosystem and from these specialities actors are defined. Table 2 shows overview of actors for all of the platforms.

Table 2: Overview of actors from different platforms

Platform	**Actors**
Bitcoin	Client, Miner
MultiChain	Client, Miner
Ethereum	Externally Owned Account, Contract Account, Miner
Chain Core	Client, Blockchain operator

For each platform, except Ethereum, there exists a notion of a Client and a Miner or someone who builds and agrees upon which transactions are included in a block. Client generally being the one who interacts with the blockchain (create and broadcast transactions). Ethereum introduces Externally Owned Account (EOA), that can be considered as the physical actor, and additional Contract Accounts (CA), that can be thought as a system user, a virtual account which acts

upon requested by an EOA or by another CA. Because CA is created by an EOA and as Ethereum Whitepaper mentions, that they are autonomous agents living inside the execution environments [12], we do not consider CA as a separate actor. Miners deal with validating transactions, state changes and blocks. In Chain Core, since it is a private blockchain and there is no need for anonymity, has Blockchain operators, who are either block generators or signers. Fundamentally, a "blockchain operator" is a miner, because miner's tasks in a blockchain environment are to create new blocks, sign them, validate them and submit them to the blockchain.

To conclude, we can say that a blockchain ecosystem has two actors from business perspective:

- Human actor who interacts with the blockchain by adding content by creating transactions. This actor will be called "User".
- Human or system actor responsible for verification and validation of transactions, building new blocks, signing new blocks and publishing new blocks to the blockchain. This actor will be called "Block generator", because the main purpose is building blocks.

3.1.2 Services

This section will give a comparison between the business level services provided by the reviewed platforms. The services from different platforms are presented in Table 3.

Table 3: Overview of services from different platforms

Platform	**Services**
Bitcoin	Create transactions, Mine bitcoins
MultiChain	Create assets, Create transactions, Grant permissions, Revoke permissions, Mine blocks
Ethereum	Create transactions, Create contracts, (Send messages), Mine blocks
Chain Core	Define and Issue assets, Submit transaction, Validate block, Gather valid transactions, Generate block, Publish block, Sign block, Determine who can participate in the network.

Firstly, every platform provides a service to create/submit transactions to the network. This is essential because transactions dictate the state of the blockchain and enable addition of new data to the blockchain.

MultiChain and Chain Core both have the notion of assets, which is a type of value, that is issued on the blockchain. While Bitcoin and Ethereum both have

their native currencies, bitcoin and ether respectively, MultiChain and Chain core allow creation of different assets.

While Bitcoin and MultiChain are purely based on transactions, Ethereum and Chain core also rely on state and smart contracts. Ethereum provides a service to create a new contract, that can be submitted to the network. Chain Core allows the use of smart contract while issuing assets because it has to define business rules for issuing new units of given assets and also rules for spending the assets. For more complex rules, smart contracts (or "issuance programs" and "control programs" as they are called in Chain) provide a good solution.

Since MultiChain and Chain core are both designed to support private blockchains, both support services to manage privileges. MultiChain provides services for granting and revoking permissions to and from specific users. Chain Core, as we know from actors comparison, defines Blockchain Operators, who have the control over who can access the blockchain.

When it comes to mining, all platforms except Chain Core have a specific service for mining new blocks. Mining in Bitcoin and Ethereum is publically available for anyone, while in MultiChain user needs to have 'mine' permission to perform mining. In Chain Core, traditional miners job is split into 2. Block generator will use services like gathering valid transactions, generate a block and publish it. Block signers, who validate and sign the block, use block validation services and block signing services.

In conclusion, common services among the technologies are *creating transactions*, *validating blocks* and *mining / creating blocks*. Additionally, permissioned blockchains provide services to manage permissions and access. Overall, it depends on the features blockchain offers - with Bitcoin being the most generic blockchain, the number of provided services is different compared to Chain or MultiChain. Features like assets, smart contracts and permissions add additional services to the commonly offered ones.

3.1.3 Processes

Table 4 provides overview of the processes from different platforms and this section will give a comparison between them. Many of the platforms have similar general processes, but looking at the details we can see that one is more complex then the other.

Table 4: Overview of processes from different platforms

Platform	**Processes**
Bitcoin	Network discovery process, Transaction creation process, Mining process, Block validation process
MultiChain	Handshake process (network discovery), Transactions creation process, Mining process
Ethereum	Network discovery process, Transaction creation process, Mining process, Block validation process
Chain Core	Network discovery process, Transaction process, Chain consensus process,

Every platform has a **network discovery process**, which allows new nodes to connect to existing peers and discover the network. In the case of Bitcoin and Ethereum the process is simpler compared to private blockchains. This usually involves a new node connecting to a known peer (either knowing IPs from previously connecting to the network or allowing the software to query IPs from outside source or read hard-coded values embedded into the system), verifying certain parameters, such as same version of the software blockchain properties and the current longest chain on either nodes. If there are any differences, the nodes will synchronize blockchain data (meaning the new node will match the longest chain on other nodes). The handshake process for MultiChain and Chain core are different from Ethereum and Bitcoin, due to the privacy of the blockchain.

MultiChain expands to Network discovery process introduced in Bitcoin. In addition to verifying that node is connecting to the same blockchain and using the same version of the software, the known peer must also verify that the connecting node's public address is on the permitted list. If the node is allowed to connect to the blockchain, both nodes still have to prove that they own the public address (key) by signing it with their private key. Once proven, the new node is granted access to the network and it's address is propagated to the network, making sure all the existing nodes know that a new one has connected.

In Chain core, connecting node has to specify block generator's URL, provide network token and blockchain ID. It is assumed, that all of the previous is handed out to the connecting node by the block generator itself, because in the private blockchain, the participants are known users.

Another general process that all platforms have in common is the process related to **creating and submitting transactions** to the network. Creating the transaction in Bitcoin requires user to enter amount and the receivers Bitcoin address. The process will check if the user has enough unspent outputs to make the transaction. If so, the transaction is constructed, signed and transmitted to

the network. Since Bitcoin is meant for transfer of value, MultiChain adds certain metadata to the transaction to understand what the transaction is for and also checks the user for privileges (for example, is the user allowed to submit this kind of transaction). MultiChain uses similar Input-Output style as Bitcoin, so in order to execute a similar regular transaction in MultiChain, the sender has to have 'send' permission and receive has to have 'receive' permission. Also, since MultiChain supports multiple assets, in addition to amount and receivers address, user also has to specify the asset name. To issue new assets, create new streams or manage permissions user must also have the specific privileges. Once the raw transaction is constructed, it is signed and transmitted to the network.

Transactions also work on the Input-Output principle in Chain, similar to Bitcoin and MultiChain. In Chain, transactions allow to issue new assets or spend existing units of assets. When issuing new assets, it has to comply with the rules defined in the issuance program. When spending assets, similar to Bitcoin, the amount of unspent transaction outputs must cover the value of the new transaction and in addition, the transaction must satisfy the conditions set by the Control program, which defines the rules for spending assets. Compares to MultiChain, Chain core does not have specific permissions to do certain operations, but instead have issuance and control programs.

Ethereum also supports regular value transactions, but does not use the Input-Output verification. Since Ethereum always maintains the state of the blockchain, the only requirement is that the current state has enough ether that will cover the value of the transaction. The input parameters for the transaction are similar to Bitcoin and MultiChain - amount to be transferred and address of the receiver. In addition to regular transactions, Ethereum allows to create contracts and call contract functions. In contrast to MultiChain, Ethereum is public blockchain and does not require any privilege checks.

Another common process among Bitcoin, MultiChain and Ethereum is the **Mining process** (Block generation). Bitcoin's mining process is based on the proof of work. A miner will build a new block, adds unverified transactions, adds metadata, calculates the proof of work and if he is the first one to find the proof, he can submit the block to the network. Ethereum's mining process is also based on the proof of work, but since Ethereum does not have the Input-Output transaction style, instead tracking the state of the blockchain, the mining process also requires a state transition process. In the state transition process transactions are verified and in the case of contract creation / calling contract function, code execution is also performed. Once transactions are verified by the state transition process, the miner in Ethereum will also calculate the proof of work and if he is the first one, can also submit the block to the network. MultiChain, being a private blockchain, claims that since all of the participants are known, there is no need

for the exhaustive proof of work. Also, in addition to permissions, the miner has to have 'mine' permission and it is important that the transactions are executed in order, for it to correctly record the grant and revoke operations of permissions. MultiChain also introduces a round-robin style mining to prevent one miner doing all the work - Mining diversity is responsible that all the miners will get a chance to mine new blocks.

Compared to Bitcoin, MultiChain and Ethereum, Chain core introduces a "Chain consensus process" instead of mining. When a new transaction is submitted, it will be transferred to the Block generator who will add it to the new block. After certain periods, Block generator will send the block over to block signers, who will verify the block and sign the block, after which it is sent back to the Block generator. The Block generator can only submit the block if the required block signers have signed the block. Also to note, since Chain is a private blockchain, the blockchain operators are known at all time.

Bitcoin and Ethereum also explicitly introduce the **block validation (consensus) process**, that each node will perform once the miner has submitted a new block. Since these are public blockchains and the miners are anonymous, there has to be a guarantee that the miner has indeed produced a valid block. This is what the block validation (consensus) process is for. In Bitcoin, block data structure, header, timestamp, size and all of the transactions are verified by all of the nodes in the network. In Ethereum, each node will verify that the previous block exists, validates the timestamp of the new block, number, difficulty, transaction roots, uncle roots and gas limits. Also, each node will verify the proof of work and run through the state transition process. In Chain Core, the consensus is provided by block signers, who do the validation.

To conclude, the four platforms each provide similar general processes: Network discovery, transaction creation, Block generation and submission (Mining and Chain consensus process) and Block validation process, but there are differences when it comes to what tasks are performed in each process:

- Similarities:
  - Network Discovery - New nodes need information about where to connect to. In the case of Bitcoin and Ethereum you need to know the IP of a peer to connect to. In MultiChain you need to know the name/IP of the blockchain. In Chain, User needs to know the url of the Block generator. When connected, there is a check to see if both nodes are running the same version of the software, are on the same blockchain and both nodes have the latest and longest chain of blocks. Also, once connected and everything is verified, new node's IP is propagated to the network so existing nodes will know that a new node has connected to the network.

  - Transactions - Regular transactions are similar for all platforms. Construct a transaction, add metadata (the metadata or data that is transacted is different per platform), sign the transaction and submit the transaction to the network.
  - Mining (Block generation) - Public blockchains Bitcoin and Ethereum are similar in a sense that they require proof of work for mining and also miners add a special generation transaction to the block that will reward the miner for mining the block. Private blockchains do not use proof of work because all of the participants are known in the network. The general idea of mining stays the same for all platforms - build the block and fill block header with metadata, add unverified transactions, validate the transactions and submit the block.
  - Block validation (Consensus) - Both Bitcoin and Ethereum have provided block validation processes, that each node will perform once receiving a new block. The processes are similar in terms that the node has to verify the metadata of the new block and all of the transactions (for Bitcoin make sure transactions are allowed based on unspent outputs and for Ethereum make sure the new state is valid).
- Differences:
  - Network Discovery - Main differences are with the private blockchains since they require some form of authentication. MultiChain checks that the connecting node's IP is in the permitted list and that the node has the private key corresponding to the public key. Chain requires that the connecting node has been given a network token that is generated by the Block generator.
  - Transactions - The main differences come in the form of metadata that is provided for the transaction. Since Ethereum allows to create contracts and call contract functions, MultiChain allows to issue assets, create streams and grant/revoke privileges, Chain allows the creation of assets, assigning assets and others - all of these provide different metadata to the transaction. Another difference comes from MultiChain, where there is a requirement for the sender to have 'send' permission and the receiver to have 'receive' permission.
  - Mining (Block generation) - The biggest differences in mining come from the private blockchains - MultiChain and Chain. MultiChain introduces a round-robin style mining where miners take turns mining blocks so that there is no single dominant miner. Chain introduces Block generators and Block signers, where Block generators build the block and fill it with unverified transactions and Block signers sign the block.

Finally, for Block validation (Consensus), the difference for Bitcoin and Ethereum comes in the form of the state transition function. As mentioned, Bitcoin does not keep track of the blockchain state, instead verifies transactions based on previous unspent outputs. In Ethereum, transactions modify the state, so that each node has to re-verify the state transition process for each transaction.

3.1.4 Data Models

Notable similarity between the four platforms is that each of the data models have a Block, a Block header and Transactions. Bitcoin introduces the Input-Output transactions, that is also used in MultiChain and Chain core. Ethereum on the other hand relies on the state replication, where each new block's state is the outcome of the transactions that were included in the block. Ethereum has chosen not to use the Input-Output transaction method, because it does not allow multi-stage contracts or scripts that could keep an internal state [12].

Ethereum also has the notion of Accounts, which are either user accounts or virtual contract accounts, that hold the balance, contract code and internal storage. In Ethereum's case, all of this is stored on the blockchain.

With the addition of Assets, Chain core keep an Asset entity containing only the ID for the asset. The assets are tied with certain programs, either Issuance program (for issuing new assets) or Control program (for spending assets). The Consensus program is used by Block generator to verify that a block is ready to be submitted to the network.

To conclude, the main set of entities for a blockchain technology are the Block, Block Header and Transactions. The Input-Output transactions are de facto Bitcoin solution to prevent double-spending, but there are several arguments about the use of UTXOs and their scalability [7], so in the end it depends on the kind of blockchain. In case of Ethereum, to enable the multi-stage smart contracts and let them have a knowledge of the state, UTXO's can not be used for this matter.

3.2 Reference Model

This section describes how the reference model is built using the information gather about the state of the art and the comparison of actors, services, processes and data models discussed in the previous section. The reference model is presented in Figure 26. It consists of six major components: Actors and Roles, Services, Network discovery process, Transaction process, Block validation process and Block generation process. Components represented in darker color are part of private/permissioned blockchains, whereas components in lighter color are specific to public blockchains.

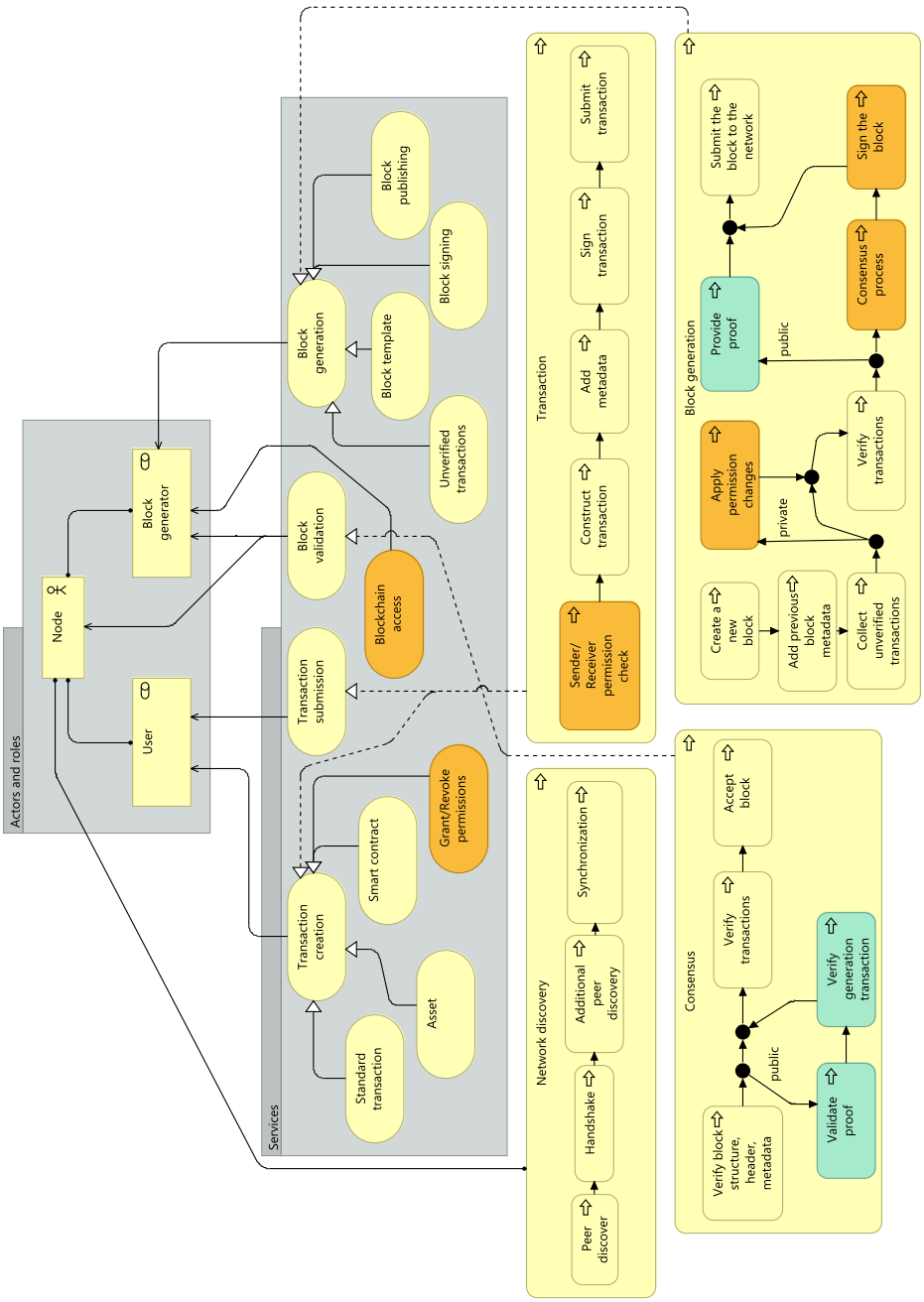

Figure 26: Complete business layer of the reference model

3.2.1 Actors

For the reference model, actors are chosen based on the discussion in Section 3.1.1. From the conclusion, the two main roles for the blockchain technology are the User and Block generator, displayed on Figure 27.

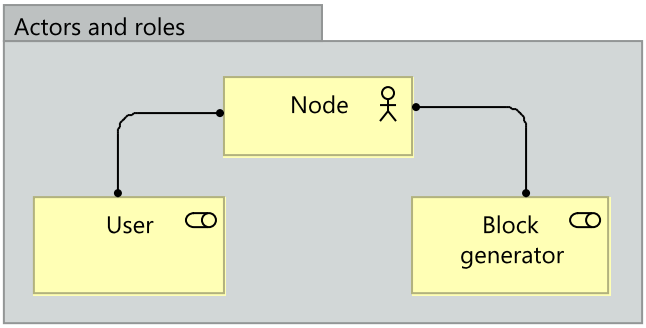

Figure 27: Actors of reference model

3.2.2 Services

The services for reference model are chosen based on the comparison done in Section 3.1.2 and are presented in Figure 28.

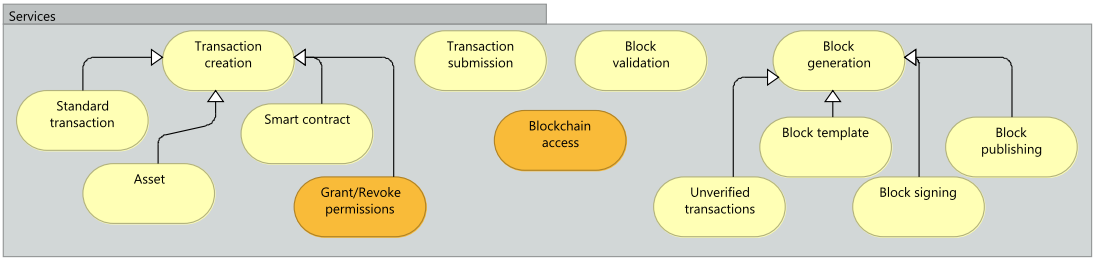

Figure 28: Services of reference model. Dark color represents services specific to private blockchains

Transactions allow *Users* to append information to the blockchain. This information can be almost anything, dependant on the implementation. Transactions can be used to create assets, spend assets, create smart contracts, call functions on smart contracts, grant/revoke permission and others. Once transaction is created, it can be broadcasted via Transaction submission service, which will also allow

signing the transaction. Blockchain access is specific to private and permissioned blockchains, where the *Block generators* or administrative users, who are known identities, will grant access to known parties. Block validation is a general service used by all the nodes to validate and agree on the newest blocks that have been added to the blockchain.

"Mining" or Block generation has been made as generic as possible, taking into account the differences between public (proof-of-work, proof-of-stake) and private blockchains. Block generation is broken down into smaller services that can be used by specific roles in the blockchain. Public blockchains with proof-of-work will let one miner use all the services, but for example on Chain, Block signing is specific to Block Signer role.

3.2.3 Processes

The processes for the reference model are composed based on the discussion in the comparison of processes (see Section 3.1.3). The processes from a top-level view are presented on Figure 26. The detailed process models are presented alongside discussion.

The four main processes groups are: **Network discovery**, **Transaction creation**, **Block generation** and **Consensus**. The processes also have a distinct color scheme, where tasks in darker color are performed in private blockchains and tasks in lighter color are performed in public blockchains.

Network discovery process (displayed Figure 29) begins by acquiring a known peers or blockchain IP. Once connected, a handshake is performed, which could be either 1) check if the user is allowed to connect via its public IP or a network token; 2) version verification 3) checking the difference between latest blocks. After handshake, if the node has successfully connected, its IP will be propagated to the network - letting other peers know that a new node has connected and IPs of the existing peers will be shared to the new node. Once the network is discovered and if there were any differences in terms of the latest block, the connecting node will synchronise its blockchain.

Transaction process (Figure 30) begins by constructing a transaction and adding relevant metadata, dependant on the type of transaction (standard transfer of funds, creating assets, deploying smart contract or other). In the case of private blockchains, there will be an additional permissions check - 1) is the specific user allowed to create the transaction; 2) is the specific receiver allowed to receive funds; 3) does the network allow transfer of specific funds or creation of a specific type of transactions. If the transaction creation is permitted, *User* will sign the transaction and it will be broadcasted to the network, to the neighbouring nodes. What the nodes do with the received transactions is up to the implementation, there could be a validity check or it could just be propagated forward to a *Block*

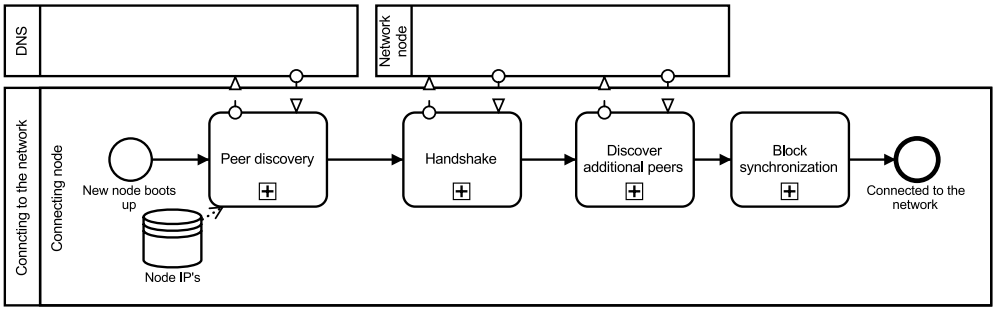

Figure 29: Network discovery process

generator.

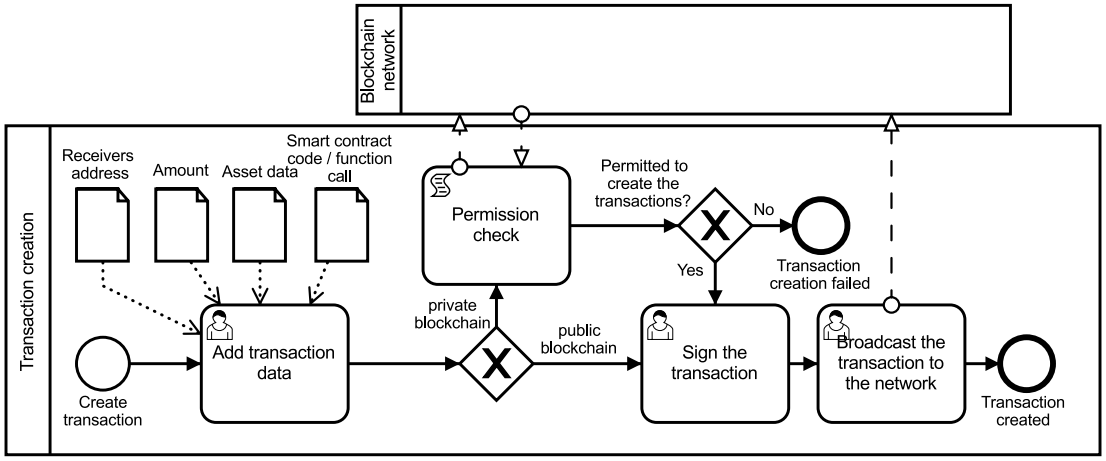

Figure 30: Transaction creation process

First step in the **Block generation process** (Figure 31) is creating a new block and previous block metadata is added. Unverified transactions are collected from the pool and they will be validated. In permissioned/private blockchains permission changes have to be applied first and in order to avoid permissionless users performing unwanted actions. All the transaction will also be validated against the consensus rules on the blockchain. In public blockchains, the block generators (miners) will provide proof for the work (proof-of-work or proof-of-stake or others) and will be rewarded for their work. In private blockchains, the consensus process might be part of the block generation process. Once the block is constructed, it is submitted to the network and propagated to all the nodes.

Consensus process (seen on Figure 32) is performed by each node when

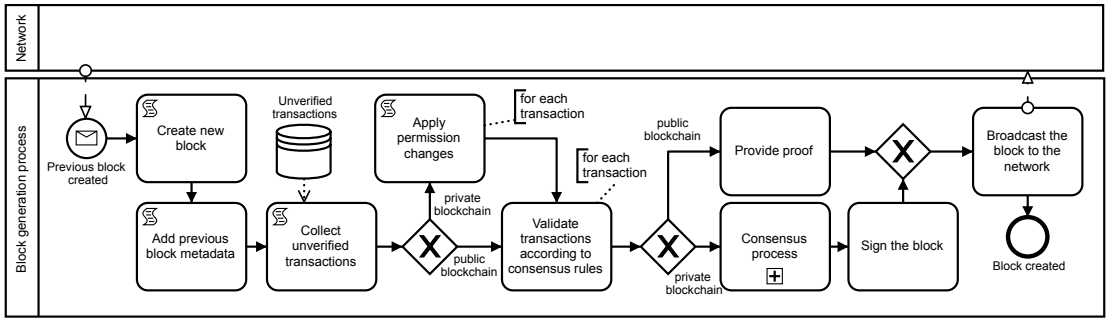

Figure 31: Block generation process

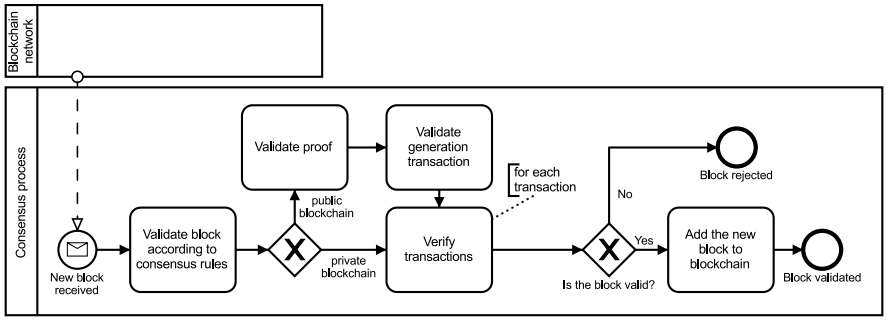

Figure 32: Consensus process

a new block has been broadcasted. Each blockchain defines its own "Consensus rules", according to which blocks and transactions are validated. Additionally, in public blockchains, proof of the block generators work as well as the reward are validated as well. If the block and the transactions it contains are all valid, nodes will append the block to the blockchain. If the block is not valid, it is rejected and not added to the blockchain.

3.2.4 Data model

The data model is presented in Figure 33. This model is mainly inspired from the domain model of Ethereum (see Figure 21).

Keeping the state of the blockchain is preferred opposed to choosing a Input-Output type transaction logic. Since UTXO's are stateless [7], they are better used for issuing assets or performing standard transfer of value (assets or cryptocurrency). However, since smart contracts are powerful, keeping the state of the blockchain allows for more complex logic.

State contains accounts, which have balance and address. In case of contracts,

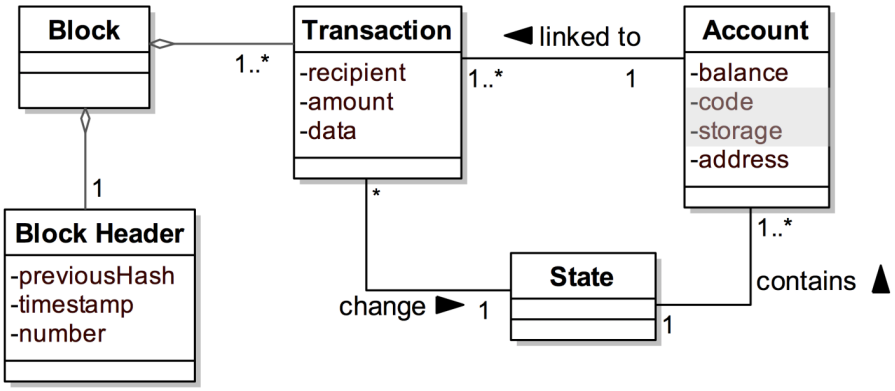

Figure 33: Data model

they also have to keep the executable code and storage specific to given account. Accounts are linked to transactions - each transaction changes the state of the blockchain (balance changes, contracts function calls and others).

As standard for every analysed platform, Block and Block Header are essential to blockchain technology. Block contains transactions, while the metadata (previous block's hash, timestamp, number) of the block is kept in the Block Header.

3.3 Answers to Research Questions

This chapter described my contribution and how the reference model for blockchain technology was conceived. The research question for this chapter was "How to build the reference model?" (See **SRQ2** in Section 1.1). This question was broken down into sub-questions.

What are the differences and similarities between the considered properties? - The reference model is the result of analysing individual platforms of the blockchain technology, extracting relevant information from the sources (actors, services, processes, data models) and comparing the findings. The four properties were compared separately, taken into account the specifics of public/private blockchains. Each of the entities was laid out into a table, followed by a discussion of the differences and similarities.

What are the criteria for including an entity in the reference model? - Based on the discussion, generalised entities that shared the most similarities were chosen to be present on the reference model. Some entities, that were specific to public/private blockchains and smart contracts, were also added in order to make the model modular and be applicable to different requirements.

Now that the conceptual reference model has been built, the next two sections present our approach to validation.

4 Accuracy Validation

This chapter will provide the first answer to sub-research-question "What are the means of validating the model?" (See **SRQ3** in Section 1.1). For this chapter, the formulated sub-question is "How to assess the correctness of the reference model?". We will analyse the reference model in comparison to the four implementations used to build the model and additionally pick four new implementations and see how the reference model performs compared to them.

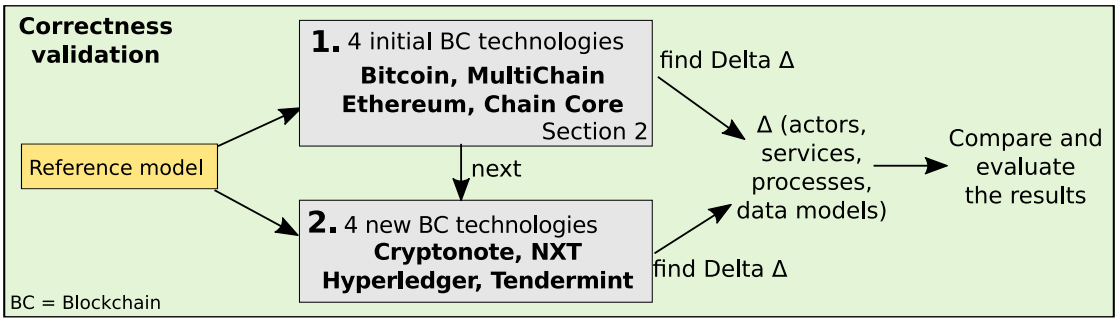

Figure 34: Validation of correctness of the reference model.

4.1 Platforms Used to Construct the Reference Model

Correctness is a form of validation where we assess the accuracy of the reference model. The validation is carried out by comparing the reference model to existing blockchain solutions. First, we will compare the reference model to the four platforms that were used to build the model itself. The models will be compared and based on that a Delta (Δ) will be calculated, which will represent the difference between the models and will also provide validation for correctness. Once the delta for the chosen implementations is found, we will pick 4 new implementations and calculate the delta value for them and compare the overall results. For the first part, our goal is to show that the reference model is sufficient to represent all four initially chosen blockchain technologies.

4.1.1 Delta Definition

For finding the overall delta Δ, which will represent the difference between models, we will compare actors (Δ_{actors}), services ($\Delta_{services}$), processes ($\Delta_{processes}$) and data models ($\Delta_{datamodels}$) separately. The overall Δ will be the sum of all deltas ($\Delta = \Delta_{actors} + \Delta_{services} + \Delta_{processes} + \Delta_{datamodels}$). $\Delta = 0$ means they have the same entities. $\Delta > 0$ means the reference model has more entities, for

example private / public / permissioned blockchain specific actors / services / processes. $\Delta < 0$ means that the reference model is missing some entities, that the comparable model has.

4.1.2 Delta Boundaries

Missing information will not be taken into account in the comparison. The goal is to compare what was present in the literature.

Δ_{actors} - We strictly compare roles on the model. Actors with same role and purpose will be counted as one.

$\Delta_{services}$ - We compare services or groups of services that have similar purpose, end goal. For example services might be broken down for proof-of-stake, but they are presented as single service for proof-of-work. Since they are both part of the consensus process, they are counted as one.

$\Delta_{processes}$ - For each of the major processes (Networking, Transactions, Block generation, Consensus), we compare the tasks for each process.

$\Delta_{datamodels}$ - We compare the classes presented on the models.

4.1.3 Bitcoin

Appendix A presents a side-by-side comparison table of Bitcoin and reference model. The following discussion explains how the Delta Δ values are derived.

Actors In terms of the reference model, Client is a User, who adds data to the blockchains via transactions - $\Delta_{actors} = 0$. Miner is specific to public blockchains, but in terms of the blockchain technology, miner is someone who adds trust to the system, someone who creates consensus by validating transactions, building blocks, calculating proof-of-work and publishing new blocks to the blockchain - $\Delta_{actors} = 0$. Overall $\Delta_{actors} = 0$.

Services Both of the models present Transaction creation service, but reference model has also defined transaction submission service separately. For Bitcoin, this functionality is included in "Create transactions" service - $\Delta_{services} = 0$. "Block validation" is not presented as a service in Bitcoin model, rather something that is done automatically by every node. "Blockchain access" is something that is specific to permissioned blockchains and is not present in the Bitcoin model - also due to this, it will not be taken into account for delta $\Delta_{actors} = 0$. Mining service is a representation of the Block generation service - $\Delta_{actors} = 0$. Overall $\Delta_{actors} = 0$.

Processes The Network discovery process differs from one aspect and that is permission check. This is not present in Bitcoin, because Bitcoin is public blockchain - thus we will not count this towards the delta $\Delta_{processes} = 0$.

Same with Transactions, Block generation and Consensus processes, which all introduce permission checks, but will not be taken into account for the public Bitcoin blockchain - $\Delta_{processes} = 0$. Overall $\Delta_{processes} = 0$.

Data models Bitcoin does not feature Accounts or State defined in the reference model, due to not having accounts and it's Input-Output transactions nature. Due to this, Bitcoin is not able to provide highly sophisticated smart contracts - Overall $\Delta_{datamodels} = 2$

4.1.4 MultiChain

Appendix B presents a side-by-side comparison table of MultiChain and reference model. The following discussion explains how the Delta Δ values are derived.

Actors Similarly to Bitcoin, MultiChain presents two actors. Client, who interacts with the blockchain by issuing assets, spending assets, granting / revoking permissions all via transactions, and a Miner, who is equivalent to Block generator on the reference model. Overall $\Delta_{actors} = 0$.

Services Creating assets, granting permissions and revoking permissions are all part of creating a transaction, each with different type of input data presented to the transaction that will be broadcasted to the network. Blockchain access in MultiChain is controlled by an administrator, that created the first genesis block. Overall - Multichain features all the same services as presented on reference model - $\Delta_{services} = 0$.

Processes Network Discovery process is similar to Bitcoin's, but with added security. MultiChain requires that the connecting node's public IP is on the permitted list and that the connecting node verifies itself by proving ownership of the private key. This is all part of the Handshake process in the network discovery and will not add anything to the delta.

Transactions and Block generation both begin with a permission check to make sure the user can perform these actions. MultiChain also introduces "spacing" which is a round-robin like mining, where one miner can mine specific number of blocks before the task is handed over to another miner in the network. Since there is no available information on Consensus process, but because this is a necessary step towards consensus on the blockchain, this process is assumed to be similar

to Bitcoin, based on the information that MultiChain is a fork of Bitcoin [14]. Overall $\Delta_{processes} = 0$.

Data models Due to limited literature, there is a lack of description regarding the Data model for MultiChain. From the whitepaper [14], we understand that the Transactions have similar Input-Output type as Bitcoin and Block and Block Header are a must on blockchain technology. Since there is no mention of keeping Accounts on the blockchain, or keeping the state - Overall $\Delta_{datamodels} = 2$.

4.1.5 Ethereum

Appendix C presents a side-by-side comparison table of Ethereum and reference model. The following discussion explains how the Delta Δ values are derived.

Actors Ethereum presents 3 actors, but as discussed in Section 3.1.1, we do not consider Contract Account as a separate actor for the reference model - Overall $\Delta_{actors} = 0$.

Services Creating contracts is also part of the Transaction creation. Similar to MultiChain's assets, streams and permissions, contract is a part of metadata that goes into the transaction. Also, sending messages is a part of communication done between CA's using transactions Ethereum is a public blockchain, so we do not take the Blockchain access service into account. Overall $\Delta_{services} = 0$.

Processes Network discovery process is simpler compared to MultiChain and Chain, but still features Peer discovery (connecting to bootstrap nodes), Handshake (verify versions), Network propagation (connecting to multiple peers) and Synchronization (downloading the latest block data when connecting to the network). Both transaction and consensus processes have permission controls, which is not taken into account due to non-private nature of the blockchain. Overall $\Delta_{processes} = 0$.

Data models The Data model presented on the reference model covers all the entities present in Ethereum. Overall $\Delta_{datamodels} = 0$.

4.1.6 Chain Core

Appendix D presents a side-by-side comparison table of Chain Core and reference model. The following discussion explains how the Delta Δ values are derived.

Actors Chain presents 2 main actors - Issuer / Spender of assets (essentially User) and Blockchain Operator (which is broken into Block generator and Block signer, due to the proof-of-stake type consensus model). Overall $\Delta_{actors} = 0$.

Services Defining and issuing assets is part of the creating a transaction. Chain has number of services that regard Block generation (Gathering transactions, generating block, signing block, publishing block) and due to this, they are also counted as one. Blockchain access is controlled via network tokens, that are handed out by the administrator. Overall $\Delta_{services} = 0$.

Processes Similar to MultiChain, a connecting user must know the IP / name of the blockchain and that user must have been granted a network token from the blockchain operator. Upon connecting, the blockchain is synchronized and the network discovered. Transactional process introduces "Issuance program validation" and "Control program validation" which control and issuance and spending of given assets. This is similar to Bitcoins general consensus rules, or can be controlled with Ethereums smart contracts for custom assets. Block generation is different from other platforms - Chain features proof-of-stake where the block is generated by a single person and M out of N block signers have to validate and sign the block in order for it to get submitted to the blockchain. This is still compliant with the model (private blockchains feature a consensus process as a sub-process in the block generation process) - Overall $\Delta_{processes} = 0$.

Data models The Data model features the standard Block, Block Header and Transaction. Chain introduces Assets and Programs separately, but lack accounts or state. This gives an Overall $\Delta_{datamodels} = 0$.

4.2 Platforms Not Used to Construct the Reference Model

Second part of the correctness validation consists of researching four new blockchain technologies, which are:

- Cryptonote [25] - a technology that aims to solve the issues of untraceability and unlinkability. With Bitcoin, transactions can be linked together via Outputs and Inputs and there is a possibility to trace the money's movement back to its origins.
- NXT [26] - a proof-of-stake based public blockchain, which also introduces accounts for permissioned operations.
- Hyperledger Fabric [15] - a modular blockchain solution for supporting different enterprise requirements and a unique approach to consensus by endorsing transactions before being included in a block.

- Tendermint [18] - blockchain with a state-replication machine (similar to Ethereum) for building smart-contracts and applications

Regarding these four platforms, we will be using the reference model as a check-list to analyse what is present on a given platform, that is also present on the reference model. Similarly, we will provide a Delta (Δ) value for each comparison, but with a modified Delta definition.

4.2.1 Validation Method

Compared to correctness validation in part 1 (Section 4.1), this section will have the Δ defined differently. Since the research and analysis is not as detailed and complete as for the first four technologies, the Δ value in this part will represent the existence of entities in the chosen implementations. $\Delta = 0$ evaluation is given when all of the entities provided on the reference model are present on the chosen technology. If there is a complete absence of the described entity in the documentation, it will be marked with $-$ (dash). $\Delta > 0$ means that reference model present entities that are not present on the chosen technology.

4.2.2 Cryptonote

The comparison against the reference model is done based on the whitepaper [25] and the Cryptonote standards[11]. From the comparison (presented in detail in Table 5 we can see that Cryptonote features same **Actors** as we have defined in the reference model - this gives a $\Delta_{actors} = 0$. In **Services**, Cryptonote does not feature a *Blockchain access* service, because it is a public blockchain. This is excluded in the Delta calculation and gives a $\Delta_{actors} = 0$. In **Processes**, Network discovery process is evaluated to $-$ due to the lack of detailed information in the documentation. Consensus process is also missing detailed information, but the general information provided is enough to see that the necessary steps, in terms of block validation, are taken. Transaction process and Block generation process both lack permission check, because of the unpermissioned environment. Overall $\Delta_{processes} = 0$. In **Data models**, Cryptonote is missing representations for *Account* and *State*, because of the technology being more similar to Bitcoin in terms of functions and transactions - this evaluates to $\Delta_{datamodels} = 2$.

[11] https://cryptonote.org/standards/

Table 5: Overview of Cryptonote and reference model actors, services and processes

Cryptonote model	**Is present?**	**Reference model**
Actors		
CryptoNote features the same actors as the reference model. There is an User that wants to send/receive value via transactions and miner, that mines blocks by solving proof-of-work.	✓	User
	✓	Block generator
Services		
User creates transactions similar to the reference model	✓	Transaction creation
Miners who comprise blocks and solve proof-of-work (different from one in Bitcoin or Ethereum)	✓	Block generation
Peers validate all the data and rely on proof of work to reach a consensus. [CryptoNode standards CNS009]	✓	Block validation
Public blockchain	-	Blockchain access
Network discovery		
No information available regarding description of network discovery for Cryptonote	-	Peer discovery
	-	Handshake
	-	Discover additional peers
	-	Block synchronization
Transactions		
Sender collects transaction inputs and specifies the amount of coins to be sent	✓	Add transaction data
Public blockchain	-	Permissions check
User generates a signature using a secret key and the signature is attached to the transaction	✓	Sign the transaction
Transaction is sent to peers in the network	✓	Broadcast the transaction to the network
Block generation		
Assumed to be present because the technology features a Block data structure and unverified transactions will be included in a Block. Adding previous Block's metadata is an essential step for any blockchain.	✓	Create new block
	✓	Collect unverified transactions
	✓	Add previous block metadata
Non-permissioned blockchain	-	Apply permission changes
In blockchain ecosystem, unverified transaction would not get added to the blockchain	✓	Verify transactions
	✓	Sign the block
Miner calculates the proof of work + receives a reward for the work	✓	Provide proof
Block is added to the network if a valid proof-of-work is found	✓	Submit block to the network
Consensus		
No detailed information available regarding description of Consensus process for Cryptonote. Cryptonote Standard CNS009 mentions "Peers validate all the data and rely on proof of work to reach a consensus."	✓	Validate transactions according to consensus rules
	✓	Validate proof
	✓	Validate generation transaction
	✓	Validate transactions
	✓	Add the new block to blockchain
Data models		
Block and Block Header are described in Cryptonote standards CNS003	✓	Block
	✓	Block Header
Transactions are described in CryptoNote standards CNS004	✓	Transaction
	-	Account
	-	State

4.2.3 NXT

The comparison is presented in detail in Table 6. NXT features same **actors** - one who contributes in terms of transactions and one who contributes to generating blocks - this evaluates to $\Delta_{actors} = 0$. In **Services**, similar to Cryptonote, it features the same services apart from *Blockchain access*, but since NXT is a public blockchain (does not feature access controls), this does not contribute towards delta - $\Delta_{services} = 0$. In **Processes** Network discovery process evaluates to $-$ because no information was available for the given process. The Transaction process does feature a permission check, due to the existence of accounts which can be assigned permissions, but for example, Block generation is not permissioned. Consensus process is well covered with all the steps present in the implementation. Overall $\Delta_{processes} = 0$. NXT is well covered in terms of **Data models**, with the only absence of state, because NXT is a transactions based and does not feature a state-replication machine - $\Delta_{datamodels} = 1$.

Table 6: Overview of NXT and reference model actors, services and processes

NXT model	**Is present?**	**Reference model**
Actors		
A node on the Nxt network is any device that is contributing transaction or block data to the network [26].	✓	User
	✓	Block generator
Services		
Each node on the Nxt network has the ability to process and broadcast transactions [26].	✓	Transaction creation
Known as "Block forging" on NXT	✓	Block generation
Blocks are validated as they are received from other nodes [26].	✓	Block validation
Non-permissioned blockchain	-	Blockchain access
Network discovery		
No information available regarding description of network discovery for NXT	-	Peer discovery
	-	Handshake
	-	Discover additional peers
	-	Block synchronization
Transactions		
NXT features accounts, which can be embedded with permissions for creating certain transactions	✓	Permission check
Several types of transactions are possible + there are number of required parameters	✓	Add transaction metadata
The transaction is signed using the sending account's private key [26]	✓	Sign the transaction
The encrypted transaction data is placed within a message instructing network peers to process the transaction and the transaction is broadcast to all peers on the network [26]	✓	Submit the transaction to the network
Block generation		
	✓	Create new block
Up to 255 unverified transactions will be collected and added	✓	Collect unverified transactions
Blocks contain hash of the previous block and number of Transactions stored	✓	Add previous block metadata
Non-permissioned	-	Apply permission changes
All block parameters are validated, including transactions	✓	Verify transactions

Generating account's ID, block's signature and signature for entire block is added	✓	Sign the block
NXT has proof-of-stake algorithm and onsensus is rewarded with transaction fees	✓	Provide proof
	✓	Submit block to the network
Consensus		
All possible block parameters are verified, including the effective balance of the block generators account [26].	✓	Validate block according to consensus rules
	✓	Validate proof
	✓	Validate generation transaction
	✓	Validate transactions
If block validation fails, nodes are blacklisted to prevent the propagation of invalid blocks	✓	Add the new block to blockchain
Data models		
The ledger of Nxt transactions is built and stored in a linked series of blocks [26]	✓	Block
Prefaced by a block header that contains identifying parameters [26]	✓	Block Header
Transactions are the only means Nxt accounts have of altering their state or balance [26].	✓	Transaction
NXT provides accounts, stored on the network and represented by a 64-bit address that is also the account address	✓	Account
	-	State

4.2.4 Hyperledger Fabric

The detailed comparison for Hyperledger Fabric is presented in Table 7. The Fabric features both *User* and *Block generator*, but the latter is defined in a more specific way "Ordering-service-node" - **Actors** evaluate to $\Delta_{actors} = 0$. Hyperledger presents all the **Services** that are present in the reference model - $\Delta_{services} = 0$. In **Processes**, both Network discovery and Transaction creation is well covered by the reference model. In Block generation, the transactions are "endorsed" before reaching the *Block generator* (the ordering node), which means they are not unverified when added to a block (evaluates to $\Delta = 1$). In Block generation *Apply permission changes* does not contribute toward the Delta, because the permission management is done after transaction is submitted to the network, by the endorsing peers, who "endorse" the transaction before it reached the Block generator. Consensus process is generally described in the documentation and covers the necessary steps in the reference model. Overall $\Delta_{processes} = 1$. In the **Data models**, Hyperledger is the only technology represented in this paper that does not feature a *Block header*. All of the information is kept in the Block - this evaluates to $\Delta_{datamodels} = 1$.

Table 7: Overview of Hyperledger Fabric and reference model actors, services and processes

Hyperledger Fabric model	**Is present?**	**Reference model**
Actors		
The client represents the entity that acts on behalf of an end-user. Clients create and thereby invoke transactions [15].	✓	User
Ordering-service-node or orderer: a node running the communication service that implements a delivery guarantee, such as atomic or total order broadcast [15].	✓	Consensus operator
Services		
Clients create and thereby invoke transactions.	✓	Transaction creation
Ordering services creates blocks of transactions specific to a channel	✓	Block generation
The blocks of transactions are "delivered" to all peers on the channel. The transactions within the block are validated to ensure endorsement policy is fulfilled and to ensure that there have been no changes to ledger state for read set variables since the read set was generated by the transaction execution.	✓	Block validation
Hyperledger Fabric underpins a transactional network where all participants have known identities.	✓	Blockchain access
Network discovery		
Client connects to a peer for communicating with the blockchain. The client may connect to any peer of its choice.	✓	Peer discovery
Access control lists are implemented on all hierarchical layers of the network and payloads are repeatedly signed, verified and authenticated.	✓	Handshake
Each Member on a channel has an anchor peer (or multiple anchor peers to prevent single point of failure), allowing for peers belonging to different Members to discover all existing peers on a channel.	✓	Discover additional peers
If the block height received upon *DISC_HELLO* is higher than the current block height of the peer, it immediately initiates the synchronization protocol to catch up with the network [9].	✓	Block synchronization
Transactions		
Building a PROPOSE message in Fabric ecosystem and specifying if it is a Invoke or Deploy transaction	✓	Add transaction data
Access control lists are implemented on all hierarchical layers of the network and payloads are repeatedly signed, verified and authenticated.	✓	Permissions check
Client signature is included in the transaction	✓	Sign the transaction
If client receives enough messages and signatures on the transaction, then it means that the proposal is endorsed and transaction can be submitted to the network (through ordering service)	✓	Submit the transaction to the network
Block generation		
Ordering Service creates a block	✓	Create new block
Transactions are endorsed and validated beforehand	-	Collect unverified transactions
Block header contains previous block's hash value	✓	Add previous block metadata
Transactions and their regarding permissions are managed before they reach the Block generator	-	Apply permission changes
Transactions are re-verified	✓	Verify transactions
No information about signing blocks or blocks containing signatures	-	Sign the block
Private blockchain	-	Provide proof
Blocks are delivered to peers	✓	Submit block to the network
Consensus		
No information available for detailed block validation. In fabric, transactions are validated and if all are valid, block is considered valid and will be accepted.	✓	Validate block according to consensus rules

Private / permissioned blockchain	-	Validate proof
Private / permissioned blockchain	-	Validate generation transaction
	✓	Validate transactions
	✓	Accept the new block
Data models		
	✓	Block
Block header is not present [9]	-	Block Header
Transactions are operations invoked on the chaincode. Deploy transactions / Invoke transactions	✓	Transaction
	✓	Account
The latest state of the blockchain (or, simply, state) is modeled as a versioned key/value store (KVS), where keys are names and values are arbitrary blobs.	✓	State

4.2.5 Tendermint

Detailed comparsion for Tendermint is presented in Table 8. This technology features the same **Actors** that are present in the reference model (*Block generator* is described as a "validator") - this evaluates to $\Delta_{actors} = 0$. In **Services**, Tendermint is missing the *Blockchain access* service, but since it is not permissioned or private blockchain, this does not contribute towards Delta - $\Delta_{services} = 0$. In **Processes**, all the entities are covered well, with the only abundance of permission checks. Since it is not a permissioned blockchain, this does not contribute towards the Delta - processes evaluate to $\Delta_{processes} = 0$. In **Data models**, *Account* is the only missing entity in comparison to the reference model - this evaluates to $\Delta_{datamodels} = 1$.

Table 8: Overview of Tendermint and reference model actors, services and processes

Tendermint model	**Is present?**	**Reference model**
Actors		
Users are present. They have accounts, that hold coins, that can be used by transactions	✓	User
Tendermint features validators, that participate in the consensus process [18]	✓	Consensus operator
Services		
Users create transactions to exchange coins	✓	Transaction creation
Byzantine proposer creates a block	✓	Block generation
Blocks are validated by peers receiving the block	✓	Block validation
Non-permissioned	-	Blockchain access
Network discovery		
Peers specified manually, Tendermint Core currently uses the Query connection to filter peers upon connecting, according to IP address or public key. [18]	✓	Peer discovery
Info about the connection is required	✓	Handshake
Peer-exchange protocol can be enabled, that will make peers gossip about known peers and form a more resilient network [1].	✓	Discover additional peers

Peers need to sync to common height	✓	Block synchronization
Transactions		
Value transactions, bond/unbond transaction, evidence transactions	✓	Add transaction metadata
	-	Permission check
Transactions hold cryptographic credentials	✓	Sign the transaction
DeliverTx message to deliver a transactions	✓	Submit the transaction to the network
Block generation		
New block is created by Byzantine proposer	✓	Create new block
Transactions are collected from mempool [1]	✓	Collect unverified transactions
New Block Header contains information about the previous Block	✓	Add previous block metadata
	-	Apply permission changes
Transactions are verified in the delivery process	✓	Verify transactions
All validators who vote for the block, provide their signature	✓	Sign the block
Byzantine consensus and the ransactions fees are divided among validators	-	Provide proof
Transactions are broadcasted to the network	✓	Broadcast the block to the network
Block validation		
A block is said to be valid if all the transactions in the block are valid and sufficient signatures are included in the validation [18].	✓	Validate block according to consensus rules
	-	Validate proof
	-	Validate generation transaction
	✓	Validate transactions
	✓	Add the new block to blockchain
Data models		
Block has header, data and signatures from the previous block	✓	Block
Block header contains most of the information regarding the block - chain id, height, block time, last block id, hash of data, hash of validation	✓	Block Header
	✓	Transaction
	-	Account
State is represented by AppHash which keeps the current state of the blockchain	✓	State

4.3 Results

Results for the Δ values are presented in Table 9. The goal of accuracy validation was to get a Δ value as close to 0 as possible for initial four technologies as well as for four new technologies that were not used as a basis of building the model.

From the results, we can see that the initial four technologies are covered almost accurately by the reference model, with subtle differences in Data models (e.g., four differences - Bitcoin and MultiChain do not support Accounts and State). We consider this result acceptable, because of the differences that smart contracts introduce to the data model (see Figure 33).

As for the four technologies that were not part of the initial building of the model, we found differences in the **Data models** (e.g., five entity differences - CryptoNote and NXT do not keep the blockchain state, Hyperledger Fabric does not have Block Header, Tendermint is missing account representation) and **Con-**

Table 9: Results of the delta Δ for the initially chosen Blockchain platforms

	Actors	Services	Data model	Networking	Transactions	Block generation	Consensus
Blockchain platforms used to construct reference model							
Bitcoin	0	0	2	0	0	0	0
MultiChain	0	0	2	0	0	0	0
Ethereum	0	0	0	0	0	0	0
Chain Core	0	0	0	0	0	0	0
Blockchain platforms *not* used to construct reference model							
Cryptonote	0	0	2	–	0	0	0
NXT	0	0	1	–	0	0	0
Hyperledger	0	0	1	0	0	1	0
Tendermint	0	0	1	0	0	0	0

sensus process (e.g., one entity differences - Transactions in Hyperledger Fabric are verified and endorsed before reaching the block generator). The information regarding **Network discovery process** was not present in the documentation for Cryptonote and NXT. Overall this result is also considered acceptable, due to the fact that the new blockchain technologies are different from the initial four technologies, but still perform well on the defined blockchain properties.

Threats to validity. The accuracy validation was done by the first author of this paper with the technologies in hand. Thus, another validation approach might conceive different results, either by defining the Delta differently or by selecting different implementations of the technology. We did not construct conceptual models for platforms which were not used to create the reference model. These were assessed following their documentations. Potentially we could miss some concepts from the comparison and some sources might be updated later on [25] [26] with more detailed information. However, this is less likely as the majority of the entities were in fact captured as shown in Table 9.

4.4 Answers to Research Questions

In this chapter we described the first approach towards validating our reference model. The main research question for this chapter was "What are the means of validating the model?". A sub-question was formulated to give a first answer to this question:

How to assess the correctness of the reference model? - The reference model was used in comparison to the four initial platforms, that were used to build the model (Bitcoin, MultiChain, Ethereum, Chain Core) and also to four new

platforms (Cryptonote, NXT, Hyperledger Fabric and Tendermint). We defined a Delta (Δ) metric, that represents the difference between a given platform in regards to our reference model.

The results reflected a good overall coverage when comparing to different technologies. This could potentially mean, that the conceptual reference model can be considered as a baseline for the technology, from which new implementation can build upon. This being said, the comparison was done from the authors perspective and may yield different results when comparing to other implementations, but the risk is minimal since the model is performing good on the researched technologies.

5 Security Assessment

First part of the validation (see Chapter 4) was about the correctness of the reference model - how accurately does it compare to other technologies. This chapter will provide an additional answer to "What are the means of validating the model?" (See **SRQ3** in Section 1.1). The sub-question is formulated as "How to assess the utility of the model?". The model will be used for security assessment (see Figure 35) - we will gather a set of known security risks that are already identified for blockchain technology and represent using the reference model, with additional examples from the modeled platforms. The goal is to see if the reference model can be used for such evaluation and can we find out, what assets or components are targeted or affected by certain attacks or risks. To present the researched risks, we will use Information System Security Risk Management (ISSRM) alignment with Archimate [13] and BPMN [4].

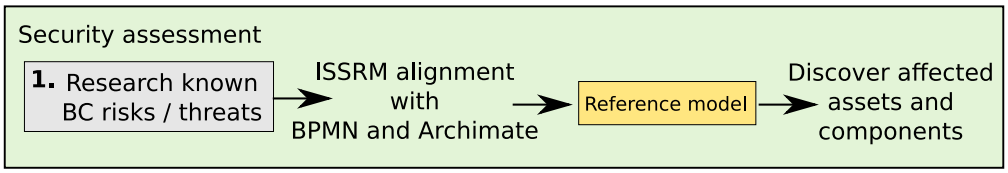

Figure 35: Security assessment using the reference model

5.1 Security Risks

Table 10 illustrates security risks which we have identified in the literature [19] [16]. We define them using ISSRM domain model [20].

Risk is a combination of threat and vulnerabilities, that lead to a negative impact harming assets [20]. In *DNS Seeds man-in-the-middle attack* (see Table 10), the attack happens when DNS query is intercepted by the malicious seed (*Threat agent*). The intercepted query will return IPs to compromised nodes, that can be part of another network. This means that the DNS Seed query is *vulnerable*, because it can be intercepted and additionally, the query results are not authenticated. This leads to a loss of DNS query integrity and reliability, which *impacts* the broadcasted transactions and received blocks. All of this combines into a *risk*, where a malicious seed will intercept the DNS query and return IPs that are not authenticated. This leads to the honest node connecting to a compromised network, where transactions and blocks are controller by the network.

The same type of analysis is done for the *Sybil attack* and *Selfish mining (51% attack)*, which is presented in Table 10. Next, we will present the ISSRM-ArchiMate alignment for the security risks.

Table 10: Blockchain related risks decomposed using the ISSRM Risk-related concepts.

ISSRM	DNS Seeds man-in-the-middle attack	Sybil attack	Selfish mining (51% attack)
Risk	Malicious seed intercepts the DNS query and returns compromised node IP addresses that are not authenticated. The honest node will connect to a compromised node/network	Attacker fills the network with finite number of nodes, trying to increase the chance of a connecting node to be surrounded by compromised node. Then compromised nodes can control what is broadcasted.	Majority of the miners control what is included in the blocks and create new blocks faster then honest miners [16].
Impact	Loss of DNS query integrity and reliability. Unsafe node IPs are returned. Unreliable network. The transactions broadcasted and blocks received are controller by the compromised network.	Loss of network reliability. Blocks / Transactions are not broadcasted honestly	Loss of network reliability. Broadcasted blocks may not be built honestly.
Event	Malicious seed can intercept the DNS query and return IPs of compromised nodes, because the results are not authenticated	Attacker fills the network with finite number of nodes in order to increase the possibility of the honest node connecting to surrounding compromised nodes	Majority of the miners control what is included in the block and submit the block to the network faster then honest miners.
Vulnerability	DNS query interception. DNS query results are unauthenticated. In public blockchains, nodes do not authenticate with peers.	Possibility to create and control finite number of nodes.	Block generation can be dominated by majority with lots of computing power. In public blockchains, nodes not not verify miners, because they are anonymous.
Threat	Malicious seed intercepts the peer IP query and returns IP of compromised peer [2]	Attacker has filled the network with compromised nodes	More then 50% of the miners in the network are compromised
Threat agent	Malicious seed intercepting the DNS query	Attacker who wants a honest client to connect to nodes controlled by him	Group of miners working together to monopolize the network
Attack method	1. Intercept the DNS query 2. Return compromised node IP	1. Create and connect finite number of nodes to the network	1. Form a group of miners 2. Start listening to the network and mine new blocks 3. Control what is included in the block and outperform honest nodes

5.2 Security Risks in ISSRM-Aligned ArchiMate

This section presents the alignment between ISSRM and Archimate [13]. The legend is displayed on Figure 36, mapping specifics are explained in Table 11 and the ISSRM concepts are described in Table 10.

Risk is linked to both the cause (*Loss event*) and mitigation (*Control objective*). *Control objective* is a concept used to implement a control for the risk and *Security requirement* is a realization of that concept. *Loss event* is a loss or damage to

an asset, which is triggered by the *Threat event*. It is an event, performed by the *Threat agent*, that impacts an asset through *Vulnerability*. Components from reference model are linked to *Vulnerability* and *Risk* to show what components are vulnerable and what components could be a part of a risk.

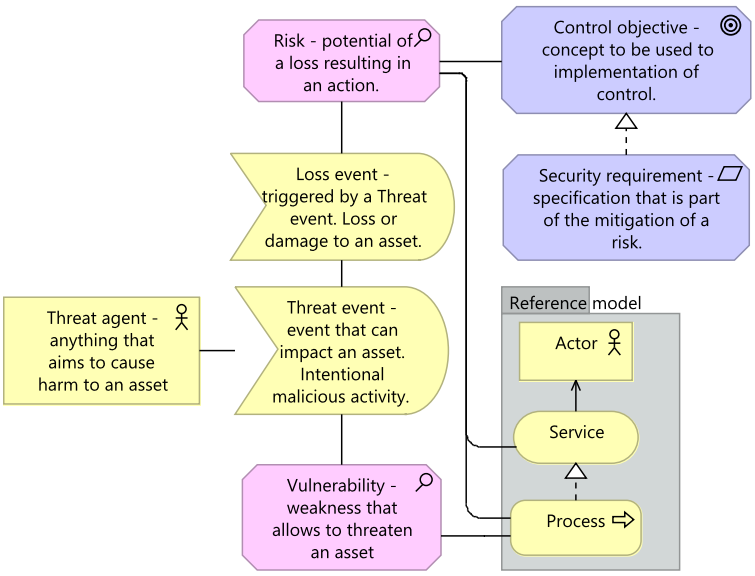

Figure 36: Legend for ISSRM-ArchiMate aligment

Table 11: Mapping between ISSRM and Archimate

ISSRM concept	**Corresponding Archimate concept**
Attack method	Threat event - event that can impact the asset
Impact	Loss event - event that is triggered by threat event Risk - Quantification of threat. Shows what risk a specific component has, that can cause the loss event.

5.2.1 DNS Seeds man-in-the-middle-attack

ArchiMate mapping for the vulnerability is displayed on Figure 37. Network discovery process is vulnerable if the blockchain technology is using DNS seed

query to help new users connect to the network. This query can be intercepted and a *malicious agent* can return the IP of a compromised node, who can be part of a compromised network. The query results are not authenticated, so the new node will accept the received IP and will connect. Once connected, services like Transaction creation become vulnerable because the created transactions will be broadcasted to the compromised network. One of the ways to prevent this type of attack would be to acquire node IP's from a trusted source, to prevent DNS seed query. Another way would be to implement DNS Seed query results authentication so that the results could be trusted.

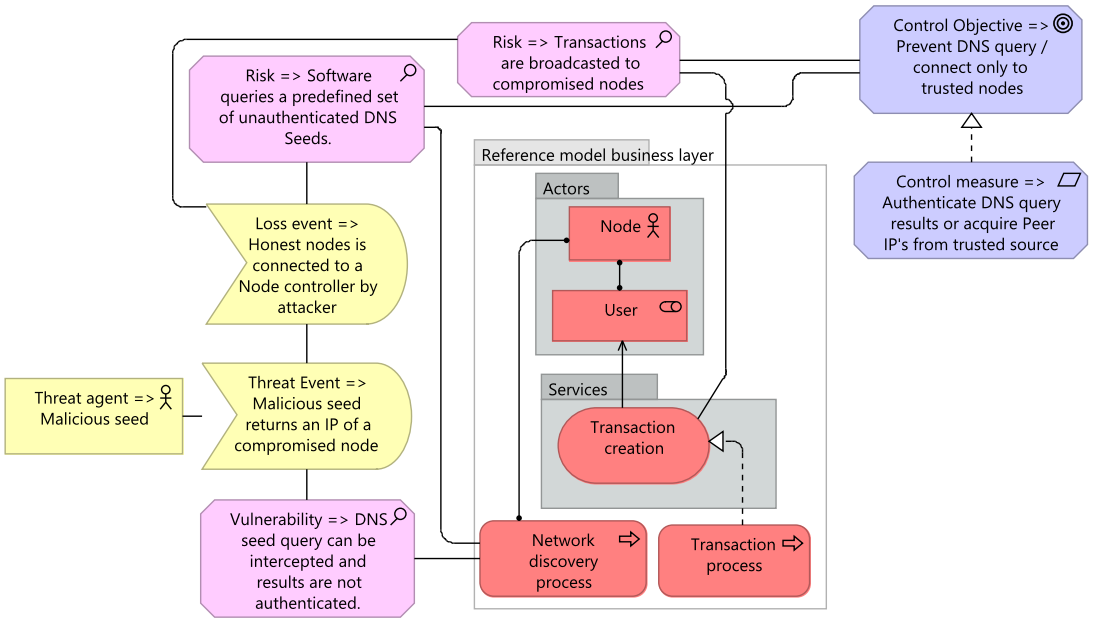

Figure 37: DNS Seeds query man-in-the-middle attack ISSRM mapping with Archimate.

5.2.2 Sybil attack

ArchiMate mapping for the vulnerability is displayed on Figure 38. Similarly to DNS Seed query vulnerability, in Sybil attack, the Network discovery process is vulnerable because: 1) Attacker can connect a finite number of nodes to the network 2) Honest node will connect to the network and will be surrounded by compromised nodes. This means that all the operations the honest node does, are relayed to compromised nodes and they may ignore it and not relay it forward to the real network. Also, compromised nodes may broadcast blocks that are not part

of the main chain, for the honest node to validate and include in its blockchain. To counter this, Sybil resistance [19] means that honest node has to be connected to a single node from the main network of nodes, to resist the Sybil attack. Then the honest node will have knowledge of the longest (and one with most cumulative difficulty) and latest blockchain - which will make the honest node ignore all blocks relayed from Sybil nodes.

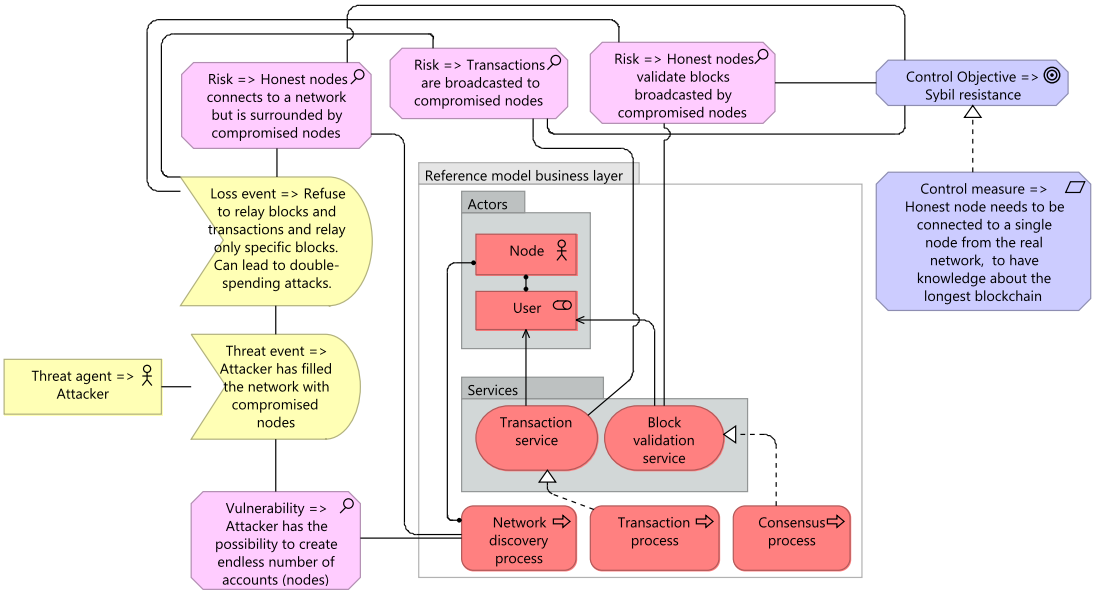

Figure 38: Sybil attack ISSRM mapping with Archimate.

5.2.3 Selfish mining (51% attack)

ArchiMate mapping for the vulnerability is displayed on Figure 39. 51% attack is where the majority of the miners control the blocks that are generated in the network. Block generation process is vulnerable because the majority, with their overwhelming amount of computing power, can mine blocks faster than honest miners. This creates a situation where compromised miners generate blocks and gather the reward for blocks. They could also control which transactions are included in the blocks. This keeps the honest miners busy trying to mine blocks, but in reality, they can not compete with the majority. Usually, private blockchains do not present problems like these because the miners are known and trusted. Another solution could be similar to MultiChain's approach, where *Mining diversity* requires different miners to generate blocks.

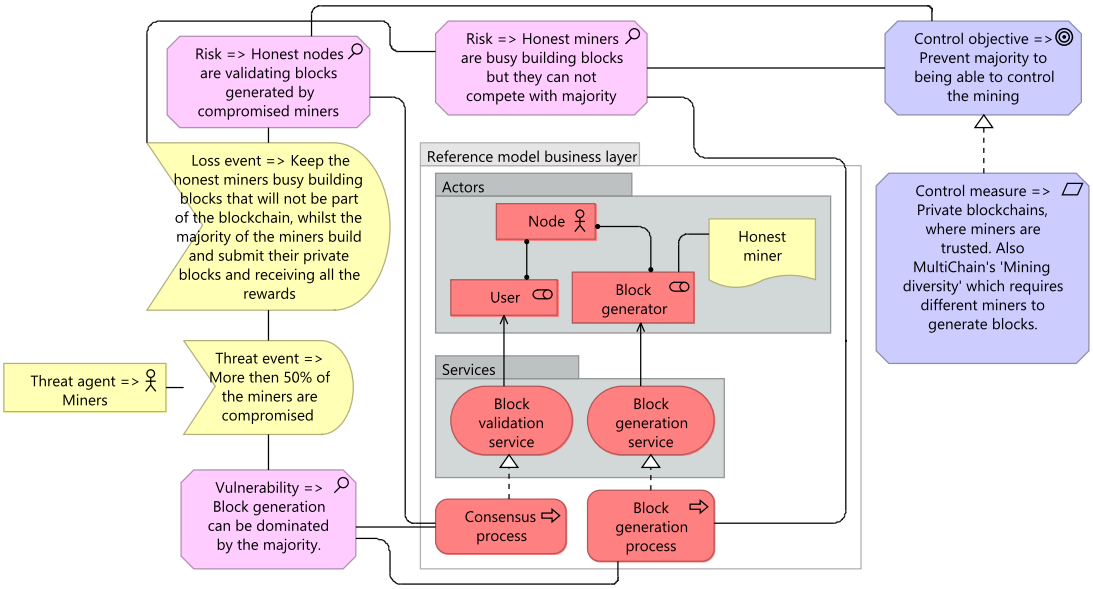

Figure 39: Selfish mining (51 % attack) ISSRM mapping with Archimate.

5.3 Security Risks in Security Risk-Oriented BPMN

This section presents the alignment between ISSRM and BPMN, with the syntax adopted from [4]. The ISSRM concepts are described in Table 10.

5.3.1 DNS Seeds Man-in-the-middle Attack

From the ArchiMate alignment (Figure 37) we can see that the **Actors** interact directly with the **Network discovery process**, that is linked with a vulnerability. The *vulnerable* component in this case is the Peer discovery process in **Network Discovery process**, seen on Figure 40. This type of vulnerability depends on the implementation and mainly affects implementations that feature a DNS query to acquire IPs. If this vulnerability is present, **Transaction creation** is under threat. Figure 41 shows, how *broadcasting transactions* becomes harmful to assets (Transactions), because they are broadcasted to a compromised network.

On Figure 42, we have expanded the reference model's *Peer discovery* sub-process (see Figure 42) of the **Network discovery process**, with the Bitcoin implementation of *Peer discovery* (see Section 2.3.2). From the BPMN we can see that *Query DNS Seed list* is the vulnerable task in the process. The results are unauthenticated and thus *Connecting to the node* becomes a threat, because the new node does not actually know where it is connecting.

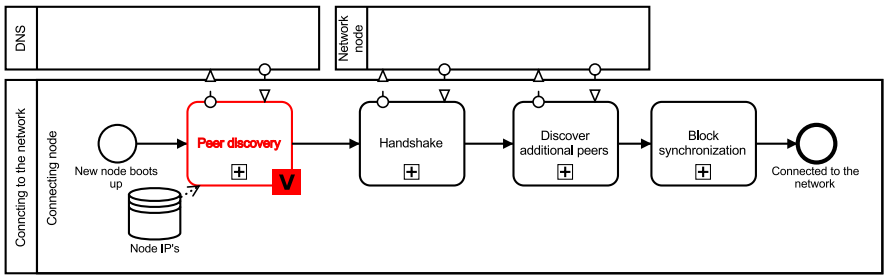

Figure 40: Vulnerable component in the Network Discovery process

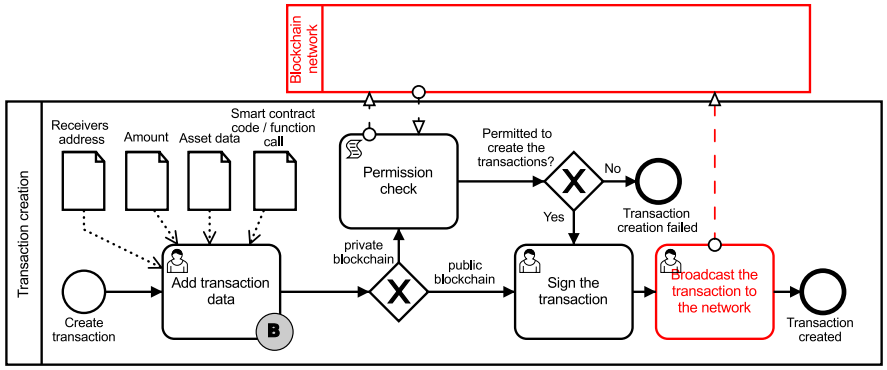

Figure 41: DNS Seeds query man-in-the-middle attack ISSRM mapping with Transaction creation BPMN

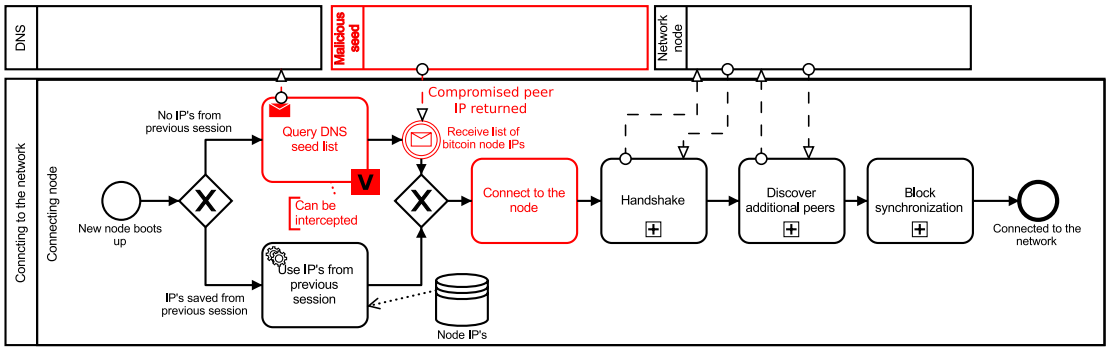

Figure 42: DNS Seeds query man-in-the-middle attack ISSRM mapping with Network discovery BPMN

On the contrary, Chain Core's **Network discovery process** (see Figure 23) does not feature a DNS Seed query. The access is managed by an administrator

who will distribute the IPs and access tokens for the participants. This mitigates the risk of executing an external query, that can be intercepted. Then again, this would make the network permissioned - only those who are given the access, can connect - and the blockchain would not be fully public anymore. This is the trade-off of having a private network - administrator can manage access control and can tell, if a certain user is connected to the network or not. For public blockchain DNS query is a required solution because new nodes need to get the initial IP of a peer to connect to. As we can see, the implementation of this sub-process is up to the developer.

5.3.2 Other Security Risks

Selfish mining BPMN mapping for the vulnerability (displayed on Figure 43). From the Block generation process, we can see that when a new block is created, the process resets and Block generators will begin working on new blocks. The *Business assets*, Transactions, are under threat because the *System asset*, *Collecting unverified transactions*, is controlled by the generator and certain transactions could be kept out of the block. Providing proof is also considered vulnerable because honest generators can not keep up with the computing power the majority has and thus they will not be able to publish new and honest blocks - they will just be busy trying to find the proof.

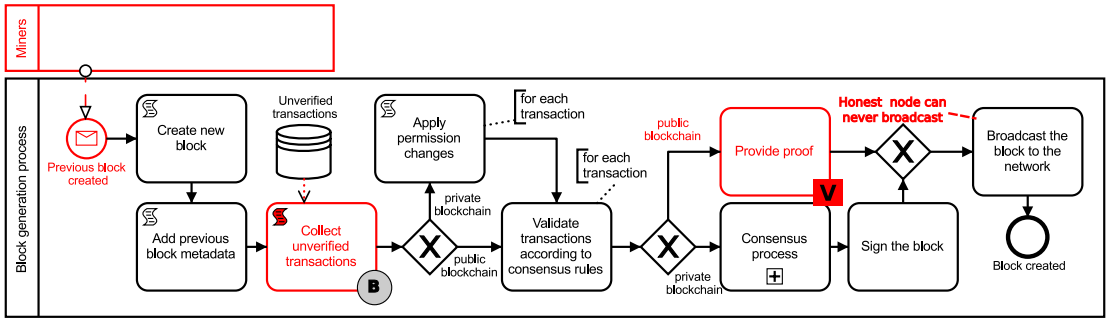

Figure 43: 51% Attack ISSRM mapping with BPMN. Block generation process.

This mainly affects public blockchains - because the block generators are anonymous. In private blockchains, all the generators are known and if there would be a group working together, it would be easily identifiable. Although, this is unlikely because private blockchains have abandoned the exhaustive proof mechanism (because generators are known) in favour fast transactions processing. Still, platforms like MultiChain have implemented "mining diversity", which restricts single generators from producing multiple blocks in succession and all the generators in the

network have to take turns in generating the blocks.

Sybil attack BPMN mapping for the vulnerability. On Figure 44, the reference model's Network Discovery process is vulnerable in two sub-processes. Firstly *Peer discovery*, because the new node can connect to a Sybil node, and secondly *Discover additional peers*, in which case the Sybil node will share IPs of other Sybil nodes. This will lead the honest node connecting to a network, where it is surrounded by Sybil nodes, which also leads to a vulnerability in Transaction creation process (Figure 45). The honest node will try to broadcast a transaction (a *Business asset*) to the network, but the Sybil nodes around him may not relay it forward and it might not be included in the next block.

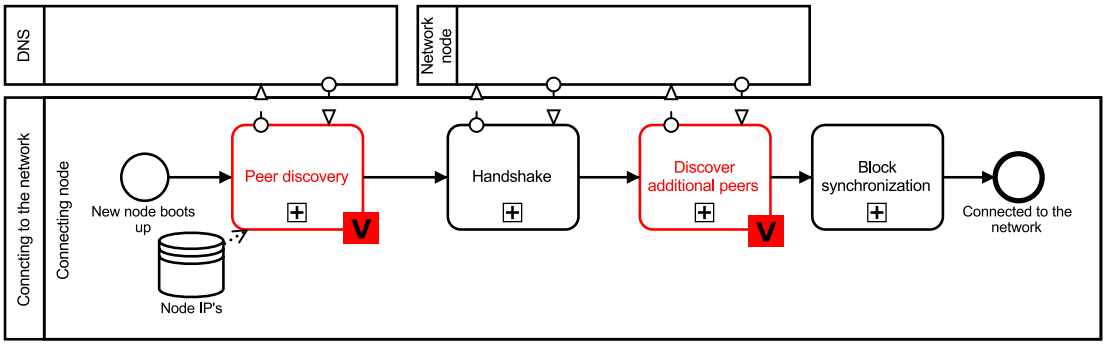

Figure 44: Sybil attack ISSRM mapping with BPMN on the Network discovery process

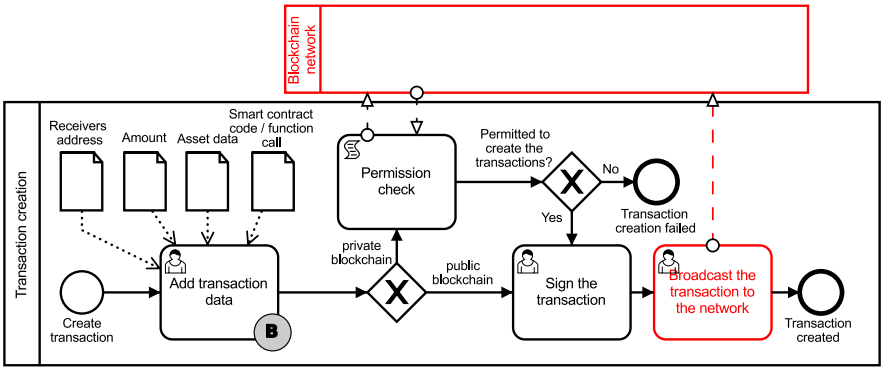

Figure 45: Sybil attack ISSRM mapping with BPMN on the Transaction creation process

Additionally, the Sybil nodes can relay blocks, that have been generated by the compromised nodes - this leaves the honest node open for double-spending attacks. In Bitcoin, such risk is mitigated when the honest node connects to another honest node. In this case, the honest node will learn about the longest chain with the highest difficulty [19] and takes that as the basis for including new blocks.

5.4 Discussion

We took the reference model, conceived in Section 3.2, and wanted to see if the model can be used to apply security-risk assessment to blockchain technology. We used ISSRM risk-related concepts and aligned them with Archimate and BPMN in order to see, which actors, assets and components are affected by different attacks on the blockchain technology. The alignment with Archimate presented a top-level view of the affected actors, services, processes and assets, while BPMN alignment allowed us to dive deep into the process and see exactly, which parts of the process are vulnerable. From the result in Sections 5.2 and 5.3 we can see which entities are affected and how. We also compared implementations from different technologies, to show where the vulnerability is present, where it is not present and what are the possible mitigations for the given threat. This also lead to a discussion about the trade-off between public and private blockchains. This type of security assessment on the blockchain would not be possible without the help of the reference model. Since the reference model is generalised, the implementation of the process is up to the developer. This security assessment should help developers in terms of understanding which threats they might be dealing with in certain steps of a process.

5.5 Answers to Research Questions

This chapter provided an alternative answer to our third research question "What are the means of validating the model?" (See **SRQ3** in Section 1.1). Specifically for this chapter, the sub-question was "**How to assess the utility of the model?**". We assessed the utility by using the reference model to perform a security assessment on the blockchain technology. We researched known security risks and presented them in ISSRM. From there, we provided ISSRM mapping with ArchiMate and BPM notations to show the link between actors and the vulnerable component.

6 Concluding Remarks

This thesis gives an overview of the disruptive blockchain technology. Bitcoin, MultiChain, Ethereum and Chain Core each presented a different approach to networking, transactions, block generation, consensus, security and permissions. The comparison resulted in the reference model, that aims to represent the domain of the blockchain technology. The model contributes to the explicit understanding of the technology and its processes. Additionally, the thesis presents two approaches to validation, that were performed on the model, with the corresponding results. The first approach being accuracy validation, where we compared the reference model to a total of eight platforms - initial four that were used to build the model and four new platforms - Cryptonote, NXT, Hyperledger and Tendermint. The second approach was validation via application - by applying risk-related security assessment on the blockchain technology using the reference model. Known risks were researched and ISSRM alignment with ArchiMate and BPMN was used to capture components affected by the threats.

6.1 Limitations

Even though the technology has been around since 2009, we still encountered limitations considering the literature in hand. As mentioned in the introduction, there are very few academic studies available regarding blockchain and its representation. Most of the available non-academic papers about blockchain technology are whitepapers, documentations or technical documents; some of which are still being updated and may be incomplete. Due to this, the research presented in this paper was performed based on the available literature.

The second form of limitation comes in form of threats to the validity of the accuracy validation. The validation was done by the first author of this paper with the technologies in hand and the available literature. Due to this, another validation approach might conceive different results, either by defining the comparison metric differently, by selecting different implementations of the technology or because the sources have been updated with the latest information and the missing gaps have been filled.

6.2 Answers to Research Questions

In the introduction we presented our main research question "*How to unify the understanding of the technology, through a comprehensive reference model?*". We divided this question in three sub-questions.

What is the current state of the blockchain technology and how to model the representations? - We investigated different blockchain technologies and presented our research method for the state of the art. The technologies were Bitcoin, MultiChain, Ethereum and Chain Core. These implementations were chosen because each brought something different in terms of privacy and smart-contracts. We defined four major properties, that we considered for each technology: Actors, Services, Processes and Data models. All of the properties were modeled using known notations: 1)ArchiMate 2) BPMN 3) UML.

How to build the reference model? - We used the information gathered on the state of the art and we presented an analysis, where we compared the considered properties for each of the selected technology. From that comparison, we built the reference model based on the similarities, with the addition of specific entities and components for public/private blockchains and smart-contracts.

What are the means of validating the model? - We validated the model in two distinct ways. First, the accuracy validation, where we directly validate how well does the reference model perform in comparison to existing technologies. Secondly, we used the reference model and supported it with examples from state of the art technologies to present a security assessment on the technology.

6.3 Conclusion

Results of the accuracy validation reflect that the model performed good enough to cover the known blockchain technologies. We discovered subtle differences in comparison to the four initial technologies, that were used to build the model, as well as differences in Data models and processes regarding newly chosen technologies. Overall we consider these results acceptable due to the differences that privacy and smart-contract capabilities present.

Results of the security assessment indicate that the reference model can be used for such an assessment and we can see, which actors are affected, what services do they use and what parts of the processes are vulnerable (or cause the threat). We also discussed mitigation and compared different implementations to observe where the vulnerability is present and where it is not. Additionally, we discussed specific tradeoffs between public and private blockchains in terms of security. The results potentially indicate how to develop this fast growing technology further and in a more secure way by expanding its disruptive nature to other application domains.

When it comes to standardisation of the blockchain technology and its presentation it in a reference way, there exist some studies that thrive towards this

goal. In [11] the technology is defined using three layers - Essential, Infological and Datalogical layer. The information is presented in the UML model and explains the general overview. However, it lacks details to define relationships between actors and processes. In our work, we have used different notations to present the reference model. We explicitly present the link between the users and processes and expand these processes by showing targeted private and public activities. We hope that it would guide the business analysts and help them to communicate with the developers when it comes to working with the blockchain technology. In terms of implementation, we hope the security assessment helps to understand what are certain risks in some components, how to mitigate them and what are the tradeoffs between different implementation.

6.4 Future Work

This type of research on the blockchain technology is a first and hopefully a basis for future research. The future vision with such a reference model would be to see if the model can be used during a real blockchain technology development and how does this model contribute to the general understanding of the technology. By no means this model is complete and we do not claim that this model should be used for developing new blockchain implementations. Rather, we hope that this inspires others to do research on the blockchain and work towards standardising the technology, developing more up-to-date models as the technology progresses.

References

[1] Tendermint Documentation. `https://tendermint.com/docs`. Accessed: 2017-03-05.

[2] Bitcoin Developer Guide. Technical report, 2009.

[3] Devon Allaby. The Trust Trade-Off: Permissioned vs Permissionless Blockchains. *Fjord*, October 2016.

[4] Olga Altuhhova, Raimundas Matulevičius, and Naved Ahmed. An Extension of Business Process Model and Notation for Security Risk Management. *Int. J. Inf. Syst. Model. Des.*, 4(4):93–113, October 2013.

[5] Andreas M. Antonopoulos. *Mastering Bitcoin: Unlocking Digital Crypto-Currencies*. O'Reilly Media, Inc., 1st edition, 2014.

[6] Vitalik Buterin. On Public and Private Blockchains, 2015.

[7] Vitalik Buterin. Thoughts on UTXOs by Vitalik Buterin, Co-Founder of Ethereum, 2016. [Online; accessed 10-January-2017].

[8] Chain. Chain Protocol Whitepaper. Technical report.

[9] P.W.D. Charles. Protocol Specification. `https://github.com/hyperledger-archives/fabric/blob/master/docs/protocol-spec.md`, 2016.

[10] Roberto Capodieci David Moskowitz, Tim Swanson. A gentle introduction to blockchain technology, 2015.

[11] Joost de Kruiff. Understanding the Blockchain Using Enterprise Ontology. 2017.

[12] Ethereum. "A Next-Generation Smart Contract and Decentralized Application Platform, 2016. [Online; accessed 6-October-2016].

[13] E. Grandry, C. Feltus, and E. Dubois. Conceptual Integration of Enterprise Architecture Management and Security Risk Management. In *2013 17th IEEE International Enterprise Distributed Object Computing Conference Workshops*, pages 114–123, Sept 2013.

[14] Dr Gideon Greenspan. MultiChain Private Blockchain - White Paper, 2015. [Online; accessed 12-December-2016].

[15] IBM and Hyperledger Project. Hyperledger Fabric Documentation. Technical report, 2017.

[16] Richa Kaushal. Bitcoin: Vulnerabilities and Attacks. *Imperial Journal of Interdisciplinary Research*, 2(7), 2016.

[17] Casey Kuhlman. How I (currently) Explain The State of Blockchains To Executives and Researchers, 2015.

[18] Jae Kwon. Tendermint: Consensus Without Mining. *URLhttp://www.the-blockchain.com/docs/Tendermint Consensus without Mining.pdf*, 2014.

[19] Jameson Lopp. Bitcoin's Security Model: A Deep Dive, 2016.

[20] Nicolas Mayer, Jocelyn Aubert, Eric Grandry, Christophe Feltus, and Elio Goettelmann. An Integrated Conceptual Model for Information System Security Risk Management and Enterprise Architecture Management based on TOGAF, ArchiMate, IAF and DoDAF. *CoRR*, abs/1701.01664, 2017.

[21] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. Technical report.

[22] Steven Norton. CIO Explainer: What Is Blockchain? *The Wall Street Journal*, 2016.

[23] Kasey Panetta. Blockchain Combines Innovation with Risk. *Gartner*, October 2016.

[24] Marc Pilkington. Blockchain Technology: Principles and Applications. *Research Handbook on Digital Transformations, edited by F. Xavier Olleros and Majlinda Zhegu. Edward Elgar*, 2016.

[25] Nicolas van Saberhagen. CryptoNote v 2.0, 2013, 2014.

[26] Nxt Wiki. Whitepaper:nxt — nxt wiki, 2016. [Online; accessed 26-February-2017].

[27] Dr. Gavin Wood. Ethereum: A Secure Decentralised Generalised Transaction Ledger Homestead Revision. Technical report.

A Table comparing Bitcoin to the reference model

Table 12: Overview of Bitcoin and reference model entities

Bitcoin model	Reference model
Actors	
Client	User
Miner	Consensus Operator
Services	
Create transactions	Transaction creation
	Transaction submission
Mine bitcoins	Block generation
	Block validation
	Blockchain access
Processes	
Network discovery	
Query DNS seed list, Connect to the node	Peer discovery
Transmit version message, Send verack and version message back	Handshake
Send IP to neighbour, Forward IP to neighbour nodes, Request IP addresses of other peers, Send IP addresses	Discover additional peers
Block synchronization	Block synchronization
Transactions	
Enter amount and receivers address	Add transaction data
	Permissions check
Construct the transaction	Sign the transaction
Transmit transaction to the network	Broadcast the transaction to the network
Block generation	
Build a new block	Create new block
Collect unverified transactions	Collect unverified transactions
Add fingerprint of previous block	Add previous block metadata
	Apply permission changes
Verify all transaction	Verify transactions according to consensus rules
-	Consensus process
-	Sign the block
Calculate proof-of-work, Add generation transaction	Provide proof

Submit block to the network	Broadcast the block to the network
Consensus	
Validate Block data structure, Validate block header, Validate block timestamp, Validate block size	Validate block according to consensus rules
Validate proof of work	Validate proof of work
Validate generation transaction	Validate generation transaction
Validate all transactions	Validate transactions
Accept the new block	Add the new block to blockchain
Data models	
Block	Block
Block header	Block Header
Transaction (Transaction Output, Transaction Input)	Transaction
	Account
	State

B Table comparing MultiChain to the reference model

Table 13: Overview of MultiChain and reference model entities

MultiChain model	**Reference model**
Actors	
Client	User
Miner	Consensus operator
Services	
Create Assets	
Create transactions	Transaction creation
Grant permissions	
Revoke permissions	
Mine blocks	Block generation
	Block validation
	Blockchain access
Processes	
Network discovery	
Connect using blockchain name, IP and Port	Peer discovery
Transmit version, verack and verack-ack message, Verify that connecting node is on the same blockchain, Verify blockchain parameters, Verify version and permitted public address, Send challenge message, Sign with private key and send challenge message back, Sign with private key and send back	Handshake
Assumed to be similar to Bitcoin	Discover additional peers
Assumed to be similar to Bitcoin	Block synchronization
Transactions	
Set asset name, quantity and receivers address, Set asset name and quantity, Set stream name and details, Set address and permission(s)	Add transaction data
Verify an input with permission, Verify an output with permission	Permissions check
Sign the transaction	Sign the transaction

Send the transaction to the network	Broadcast the transaction to the network
Block generation	
Permission check	
Construct a new block	Create new block
Collect unverified transactions	Collect unverified transactions
Assumed to be similar to Bitcoin	Add previous block metadata
Apply permission changes in order	Apply permission changes
Count number of new miners	
Calculate spacing	
	Verify transactions according to consensus rules
	Consensus process
	Sign the block
	Provide proof
Submit new block	Broadcast the block to the network
Consensus	
	Validate block according to consensus rule
	Validate proof of work
	Validate generation transaction
	Validate transactions
	Add the new block to blockchain
Data models	
Block	Block
Block header	Block Header
Transaction (Transaction Output, Transaction Input)	Transaction
-	Account
-	State

C Table comparing Ethereum to the reference model

Table 14: Overview of Ethereum and reference model entities

Ethereum model	**Reference model**
Actors	
External account	User
Contract account	
Miner	Consensus operator
Services	
Create transactions	Transaction creation
Create contracts	
Send messages	
Mine ether	Block generation
	Block validation
	Blockchain access
Processes	
Network discovery	
Receive list of recent peers	Peer discovery
Connect and synchronize with other peers	Peer discovery, Handshake, Discover additional peers
Transactions	
	Permission check
Add contract code, Add contract creation fee, Add contract address and input data, Add value and receivers address, Add recipients address, value and data, Add receivers address and set the value of ether to be sent	Add transaction data
	Sign the transaction
Submit the transaction to the network	Broadcast the transaction to the network
Block generation	
Create a new block	Create new block
Determine transactions	Collect unverified transactions
Determine ommers	
	Add previous block metadata
	Apply permission changes
Compute a valid state	Verify transactions according to consensus rules

-	Consensus process
-	Sign the block
Calculate proof-of-work, Add block miner reward	Provide proof
Submit the block to the network	Submit block to the network
Consensus	
Verify that previous block exists, Verify block's timestamp, Validate block number, difficulty, transaction root, uncle root and gas limit	Validate block according to consensus rule
Validate proof of work	Validate proof of work
Block miner rewards validation	Validate generation transaction
Verify state transition from previous block	Validate transactions
	Add the new block to blockchain
Data models	
Block	Block
Block Header	Block Header
Transaction	Transaction
Account	Account
State	State

D Table comparing Chain Core to the reference model

Table 15: Overview of Chain Core and reference model entities

Chain Core model	**Reference model**
Actors	
Issuer / Spender	User
Blockchain operator (Block generator, Block signer)	Consensus operator
Services	
Define and issue asset	
Submit tranaction	Transaction creation
Gather valid transactions	Block generation
Generate block	Block generation
Sign block	Block generation
Publish block	Block generation
Validate block	Block validation
Determine who can participate in the network	Blockchain access
Processes	
Network discovery	
Connect and synchronize with other peers	Peer discovery
Specify block generator URL, network token and blockchain ID	Handshake
	Discover additional peers
Download blockchain data	Block synchronization
Transactions	
Issuance and control programs	Permission check
Issue a new asset	Add transaction metadata
Spend existing assert	Add transaction metadata
	Sign the transaction
Submit the transaction to the network	Submit the transaction to the network
Block generation	
Batch valid transactions into a block	Create new block
Collect transactions	Collect unverified transactions
-	Add previous block metadata
-	Apply permission changes

Validate transactions	Validate transactions according to consensus rules
Validate the block, Gather signatures, Send the block back to block generator	Consensus process
Sign the block and send to block signers / Sign the block	Sign the block
-	Provide proof
Submit the block to the network	Submit block to the network
Consensus	
Validate block structure, Validate header, Validate metadata	Validate block according to consensus rule
Validate proof-of-stake	Validate proof of work
	Validate generation transaction
Validate transactions	Validate transactions
	Add the new block to blockchain
Data models	
Block	Block
Block Header	Block Header
Transaction (Transaction Output, Transaction Input)	Transaction
Asset	-
Program (Control program, Issuance program, Consensus program)	-
-	Account
-	State

Non-exclusive licence to reproduce thesis and make thesis public

I, Andreas Ellervee (date of birth: 29th of April 1993),

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

1.1 reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

1.2 make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

A Reference Model for Blockchain-Based Distributed Ledger Technology

supervised by Raimundas Matulevičius and Nicolas Mayer

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, 17.05.2017