university of groningen

faculty of mathematics and natural sciences

Masterthesis

Blockchain in Education

In collaboration with Quintor B.V.

by

Peter Jens Ullrich
S2273942
p.j.ullrich@student.rug.nl

supervised by
Primary: dr. Vasilios Andrikopoulos
Secondary: dr. Mircea Lungu
External: Johan Tillema, CEO of Quintor B.V.

Acknowledgments

I would like to thank Quintor B.V., Coffee, and The Pineapple Thief for making this thesis possible. Many albums have been made love to, only few have been written a thesis to. Where We Stood is one of the few. Also, I would like to thank my supervisor for his support and lightning fast responses.

Abstract

Blockchain technology shows great potential for financial or logistical applications, however the potential in education has not yet been explored. Application processes in higher education for programs like Erasmus+ have hard requirements for authenticity and data integrity, but offer only little to none privacy to the applicant. Traditional technologies can meet the requirements, but are unable to provide the applicant with a reasonable amount of privacy. This thesis researched the potential of blockchain technology for providing applicants in the Erasmus+ program with a Self-sovereign Identity by giving them the full and only control over their personal data. The research findings were implemented in the StudyBits project in collaboration with Quintor B.V. The findings showed that the purpose-built blockchain Hyperledger Indy can be used successfully to automatize application processes and provide applicants with an Self-sovereign Identity.

Contents

List of Figures

List of Tables

Glossary

dBFT delegated Byzantine Fault Tolerance.

DDO DID Descriptor Object.

DID Decentralized Identifier.

dPDS Decentralized Personal Data Service.

DUO Education Executive Agency/the Ministry of Education, Culture & Science.

EVM Ethereum Virtual Machine.

GDN Groningen Declaration Network.

GDPR General Data Protection Regulation.

LLL Low-level Lisp-like Language.

PDS Personal Data Service.

PKI Public Key Infrastructure.

RBFT Redundant Byzantine Fault Tolerance.

RUG University of Groningen.

SSI Self-sovereign Identity.

StePS Stichting ePortfolio Support.

TNO Netherlands Organization for Applied Scientific Research.

UGent University of Gent.

Chapter 1

Introduction

In 2008, a person or group called Satoshi Nakamoto published the Bitcoin whitepaper [1] and started a technology movement whose aim is to remove third parties from payments and data transfer and to put the individual back into control over her personal data and finances. The currency called Bitcoin, which was proposed by Nakamoto, has and will certainly change how we think about money, but in his white-paper, Nakamoto proposes another technology whose impact might even outdo the one of Bitcoin. The name of this technology was later on coined **Blockchain**. A Blockchain is a data structure in which data manipulations (i.e. transactions) are collected into batches (i.e. blocks) and added to a linked list of prior batches (i.e. the chain). The history and order of the batches in the linked list are cryptographically secured, effectively rendering the linked list and data manipulations immutable. The advantages of blockchain technology depend heavily on the context in which it is deployed, but in general, blockchain technology is most useful in three situations ([2],[3],[4]).

First, it can help remove intermediaries from networks where they play the role of a trusted third-party. Instead, it enables the network participants to make direct peer-to-peer transactions of a currency or data. Second, it can function as an auditing tool, where updates to a dataset (e.g. business expenses) are recorded in an immutable and chronological manner. This can significantly simplify record keeping and compliance especially in larger firms, where auditing becomes a complex and time-consuming task. Third, it can give back the control over personal data to the individual and enables the individual to decide for herself which part of her personal data she wants to share with whom and when to revoke the access to her data. This control over personal data is called **Self-sovereign Identity (SSI)** and will be the main focus of this thesis.

Promising use-cases of blockchain technology were already identified in a multitude of sectors, like logistics [5], payments [6], auditing and compliance [7], and supply-chain management [8]. However, the research into how blockchain can be used in higher education is rather limited and scarce so far. The purpose of this thesis is to investigate the applicability of blockchain in the educative context. In particular, this research will focus on how to improve application processes for Erasmus+ exchange positions within the StudyBits project. Towards this goal, this thesis was written in collaboration with the Dutch software company, Quintor B.V.

1.1 StudyBits Project

The StudyBits project is an innovation project spearheaded by Quintor B.V. and a collaboration between the Education Executive Agency/the Ministry of Education, Culture & Science (DUO), the Netherlands Organization for Applied Scientific Research (TNO), , Quintor B.V., Groningen Declaration Network (GDN), Stichting ePortfolio Support (StePS), University of Groningen (RUG), and Rabobank Groningen [9]. These organizations work together in the *Blockchain Field Lab Education* in Groningen, The Netherlands, and strive to use blockchain technology to create new high-quality employment and business opportunities in the Groningen area.

StudyBits is the first project of this field lab. It aims to improve the application process for students at the University of Groningen for the Erasmus+ program. Erasmus+ is a funding scheme of the European Union to support programs in education, training, youth, and sport. Two-thirds of its budget of EUR 14.7 billion is allocated to facilitate learning opportunities abroad for individuals [10], both within the European Union and worldwide. With around 750.000 Europeans going abroad every year [11] to study, train, or volunteer, the Erasmus+ program is struggling with high administrative costs and students encounter problems with having their diplomas and credits recognized by their origin university [12]. Additionally, students have to submit a significant amount of their personal data whenever they apply for an exchange position. This data is centrally stored and passed on to internal and external auditors without further consent by the student and is only deleted after a time period of 10 years [13]. During these 10 years, the student gives up her control over her personal data and is not informed of with whom this data is shared.

The aim of the StudyBits project is to alleviate the administrative and most of all, privacy problems that the Erasmus+ program is facing. In particular, its aim is to investigate how blockchain technology can be used to enable fast and verified data transfer between universities and to provide applying students with a Self-sovereign Identity (SSI). Additionally, the inspection and verification of applications should be automatized, obliterating the need for official stamps, copies, and (paper) certificates.

The scope of this thesis in regards to the StudyBits project is the application process for exchange positions at a foreign university with the main focus being to provide students with their own SSI with which they can: receive digital documents signed by their origin university (so-called "Claims"), apply for exchange positions and automatically fulfill the requirements of certain positions. The exchange university can then accept or reject student applications and verify that the data given by the students was indeed given out by the origin university. The purpose of the StudyBits project is to research the usefulness of blockchain technology. A definition of blockchain technology and its use-cases will be established in Chapter 2.

1.2 Problem Definition

The existing application processes for exchange positions in higher-education are inefficient and don't protect the privacy of the applying student ([14],[12]). Applications need to be handed in on paper and are manually verified and processes, which is a time-consuming and costly process. Additionally, the level of privacy of an applying student is very limited since most application documents contain all personal data of a student and an oversight about which employee sees and handles what personal data is mostly missing and can not easily be established [14]. Often, no tracking system for application documents exist, which means that the student has no information about which institution holds what personal data and with whom that data was shared. Also, the student has often no way of revoking the access to her personal data without putting in an unreasonable effort.

The privacy issue mentioned above can be defined as an issue of self-sovereignty. Students have to give up the full control over their personal data. A solution to this issue is the concept of a Self-sovereign Identity (SSI). Giving students a SSI implies to give them full control over their official documents and with that, full control over their personal data. A student with a SSI can decide herself with whom she wants to share her data and whether she wants to share all her personal data or only certain attributes like her date of birth or address [14]. Whenever the student thinks that the counter-party does not need her personal data anymore, she can revoke the access to that data.

Based on this problem definition, this thesis aims to answer the following research question:

How can blockchain technology enable Self-sovereign Identity?

1.3 Contributions

The aim of this thesis is to research how application processes can be improved regarding privacy and SSI as part of the StudyBits project. In particular, the research question is how blockchain technology can help to establish SSIs for exchange students. To answer this question, a literature study is done on SSI and how blockchain can facilitate SSIs. Then, a comparison of four different blockchain technologies and an evaluation of their advantages and disadvantages regarding privacy and SSI is created. Based on this evaluation, a recommendation is made for which blockchain technology would be best suited for creating a SSI-enabling application.

After this literature-based research is conducted, five use-cases of the StudyBits project are selected and a software design to implement these use-cases is developed. The design is then implemented in collaboration with Quintor and is evaluated and discussed at the end of this thesis.

1.4 Outline

I will first talk about previous research on SSI and blockchain in Chapter 2, followed by an analysis of the chosen StudyBits use-cases in Chapter 3. Based on this analysis, a design is devised in Chapter 4, whose implementation will be described in Chapter 5. Eventually, the implementation is evaluated in Chapter 6, followed by a general discussion in Chapter 7.

Chapter 2

Related Works

Before the StudyBits project is covered in more detail, this thesis will explore how SSI can be implemented with and without using the blockchain. Based on this analysis, differences between the two approaches are discussed. Before any SSI solutions are described, first a definition of blockchain is given.

2.1 Blockchain

The first blockchain was proposed and deployed by the anonymous person or group, Satoshi Nakamoto, in 2008 [1]. Nakamoto developed a decentralized peer-to-peer electronic cash system that leveraged a new technology, later being labeled "Blockchain", to create a scarce and non-duplicatable currency called Bitcoin. The blockchain technology functioned as a decentralized, distributed and immutable ledger that was governed by a consensus protocol called "Proof of Work".

In general, a blockchain is a database that is decentralized, meaning that no central authority has the full control over the database or can change its data to its liking. Furthermore, the database is distributed, which means that every node in the blockchain network holds a full copy of the database. By decentralizing and distributing the database, there is no authority that could change or remove its data, therefore the blockchain database is said to be "immutable". Thus, once data is added to the database, it cannot be removed or changed after the fact. The only allowed functions are updates and additions of data and these rules are enforced by the consensus protocol.

The consensus protocol coordinates how data is added to the blockchain and ensures the immutability of the data. A variety of consensus protocols have been proposed since Nakamoto's original Bitcoin white paper, but the most common protocol is still the Proof of Work consensus protocol. In the Proof of Work (POW) consensus protocol, participating nodes in the network, so-called “Miners”, take the updates and additions, which were not yet officially applied to the database, add a random nonce (number only used once) to the data and hash the combination of data and nonce. The aim is to set a nonce which combined with the data leads to a hash, whose numerical value is lower than a global target value. Whenever a miner finds such a combination of data and nonce, it broadcasts the combination to the network. Every node that receives the combination verifies that the combination leads to a hash that is lower than the current global target and if the verification was successful, adds the data to their own copy of the database.

Most blockchains can be used to send funds in the form of cryptocurrencies like Bitcoin between accounts, but some blockchains support an additional feature called **Smart Contracts**. Smart Contracts are programs with a set of pre-programmed rules that execute in a deterministic and tamper-proof manner [15]. When Smart Contracts are put on the blockchain, their execution and output are verified by every node in the blockchain network. Therefore, Smart Contracts run on the blockchain are trustless, truly deterministic, and can serve as an independent mediator between multiple parties. Smart Contracts are also useful for storing information and preserving its integrity and authenticity.

In summary, a blockchain is a database that only allows additions and updates of its data and keeps the whole history of every addition and update ever performed. Addition and update actions are called “Transactions” and are applied in batches to the database. These batches are called “Blocks”. Every block keeps a hash of all transactions of the previous block and therefore blocks are irreversibly linked to each other. The right to create and add a block to the linked list of blocks, so-called “Chain”, is determined by the consensus protocol of the blockchain.

2.2 Blockchain Platforms

With Bitcoin, the first blockchain was put into production, but it has not been the last one. A range of projects was launched in the past years, all of which with the aim to improve the original Bitcoin blockchain. In general, these new blockchains can be categorized into *Permissionless* and *Permissioned* blockchains.

2.2.1 Permissioned vs. Permissionless Blockchains

Permissionless blockchains allow anyone to join the blockchain network and operate in a trustless and decentralized manner [16]. No single authority has control over the network, the historical data, or the transactions. There is no single point of failure and every node in the network typically holds a full history of all transactions ever broadcasted. Since all data ever stored on the blockchain is shared with every node, there is no possibility for confidentiality of e.g. personal data or interactions between parties. This transparency is a necessity for permissionless blockchains since every node needs to be able to verify the correctness of every transaction. If a transaction would hide data, then not every full node could verify its integrity and correctness since the data needed would simply not be available to the full node. Due to this limitation to confidentiality, companies like IBM, Intel, and Evernym started to work on permissioned blockchains.

Permissioned blockchains control which nodes are allowed to join the blockchain network and assign roles to certain nodes [17]. Typically, only a few nodes collect transactions and create the blockchain without distributing the right to add to the blockchain over all nodes in the network as it is the case in a permissionless blockchain. Therefore, permissioned blockchains introduce a certain degree of trust back into the blockchain network. However, this trust enables the permissioned blockchain to scale better than permissionless blockchains [18] since not every node in the network needs to store and verify every single transaction anymore. Additionally, permissioned blockchains enable network participants to interact and transfer data confidentially and unbeknownst to the other network participants.

2.3 Self-sovereign Identity (SSI)

The concept of Self-sovereign Identity evolved from the concept of user-centric identity management, which had two goals [19]. First, to put the user as the main actor in the identity management process. The user should have control over her data [20]. This entailed that the user can view, modify, and delete as well as grant and revoke access to her personal data. Secondly, the user should be able to re-use the same identity over a multitude of services, eliminating the need to create a new identity for every service. For this purpose, a range of protocols and standards were created, which were intended to facilitate data sharing and therefore identity re-use. Examples of such protocols are OpenID, OpenID 2.0, OpenID Connect, OAuth, and FIDO.

The user-centric identity concept aimed even higher: to give users the full and only control over and access to their data. However, this goal was never fully achieved since the ownership of the identities remained with the entities that registered them [20]. This created a range of drawbacks: The registering entities retained full access to the personal data of the user, rendering the user's privacy void. Registering entities rarely offered migration services for their users' data. This limited the users' ability to freely choose their identity provider since migrating from one to another could be time-consuming and costly. Since registering entities retained ownership over user-centric identities, they had the power to delete these identities without notice [21].

The concept of Self-sovereign Identity (SSI) was created to address these drawbacks and is meant to replace the user-centric identity concept. SSI puts the user in full control of her identity. The user must be the only authority that can create, modify, or erase an identity. The identity must be interoperable across multiple services as well as transportable to any identity provider that the user chooses. The user must have the only access to the data and can grant access to all or subsets of the personal data. A list of ten principles specifies in detail what requirements a SSI must fulfill [20]. These ten fundamental principles can be combined into three higher level principles [22], **Controllability**, **Portability**, and **Security**, as follows:

Controllability	**Portability**	**Security**
Existence	Interoperability	Protection
Access	Portability	Minimization
Control	Transparency	Persistence
Consent		

Table 2.1: Fundamental SSI principles grouped by their high-level principle

These three high-level principles can be defined as follows [22]:

- Controllability: *the extent to which the user is in control of who can access her data*
- Portability: *the number of services on which the user can use identity and the extent to which the user is bound to a single SSI provider*
- Security: *the extent to which the user's data is guarded against unauthorized access*

For a system to offer a SSI to its user, the ten principles have to be implemented. In the following, the thesis will analyze how the principles can be fulfilled with and without using the blockchain. The analysis will be based on the three high-level principles instead of each fundamental principle individually.

2.4 SSI without Blockchain

SSI solutions that do not use the blockchain can fulfill the SSI principles to a certain extent. Particularly, SSI solutions offered by a Personal Data Service (PDS) can fulfill the principles to a decent degree. PDSs are centralized storage systems that hold all of a user's personal data and share said data with entities upon request mostly with, but also sometimes without the user's consent [23]. The user is said to have full control over her personal data and receives an overview of what personal data was shared with which entities. If needed, the user can revoke said data access.

With most PDSs, whenever the personal data changes (e.g. the user changes her address), then all entities with access to the updated data receive a notification about the data change. This removes the need for the user to notify every entity manually, which can save both the user and the entity significant amounts of time and costs. Examples of PDSs are the American company Digi.me and the Dutch company Qiy, but also large corporations like Facebook and Google can act as a PDS by letting users log into and sharing personal data with different websites [24].

The advantages of centralized PDSs are apparent. The user can stay in control of her personal data and keep an overview of with whom her personal data was shared. If not needed anymore, the user can revoke access to her data, which improves her privacy. Sharing and updating data becomes seamless for both the user and entities with access to her data. Therefore, the privacy and self-sovereignty of the user are improved and data sharing is significantly simplified.

However, disadvantages of such personal data aggregators are discernible as well [23]. The PDS becomes the ultimately trusted intermediary with mostly full access to the user's personal data. Both the user and the entities connecting to the PDS have to trust the PDS fully. The user has to trust the PDS to not share her data without her consent and the connecting entities have to trust the PDS for the authenticity of the shared data. Both, the user and the connecting entity have only limited means to monitor and audit the behavior of the PDS. It is hard to spot if the PDS becomes malicious and starts transferring or selling personal data without the user's consent. PDSs typically try to strengthen the weak foundation for trust in their system by issuing voluntary commitments and rulebooks (e.g. [25], [26]) for how the user's data is processed, but such commitments stay voluntary and can't be used to hold the PDS legally accountable for their actions.

Regarding the three main principles of SSI, PDSs can offer high levels of security, and medium levels of controllability and portability to the user. A centralized solution can be guarded well, which increases the security of the user's data. However, the user cannot keep the PDS from accessing her data, which limits the controllability of PDSs. PDSs typically do not offer migration services to move the personal data to and from other PDSs. This limits the portability of PDS solutions since the user is bound to use a single provider [22].

In summary, SSI solutions without blockchain, so-called PDSs, can simplify storing and sharing personal data, but become heavily trusted intermediaries whose actions cannot always be monitored and verified, especially if their software is proprietary, and cannot be audited by its users [23]. Although advertised differently, the user is never really under the full control of her data and only relies on the goodwill of the PDS for not sharing or selling her data. Now, that SSI solutions without blockchain were discussed, the thesis will cover how SSI solutions can be implemented using the blockchain.

2.5 SSI with Blockchain

SSI solutions that use the blockchain are able to offer higher levels of controllability, portability, and security than SSI solutions that do not use the blockchain. By storing the personal data on a distributed ledger instead of a centralized database, these SSI solutions overcome the drawbacks of centralized PDSs by eliminating their own data access and by facilitating provider switching since the data is stored publicly accessible ([27],[22]). Since the advent of blockchain technology, a range of identity projects that use the blockchain has been launched. Companies like Civic and uPort, just to name a few, promise to offer a decentralized and self-sovereign identity to the user. In the following, companies like Civic and uPort will be referred to as a **Decentralized Personal Data Service (dPDS)**.

The key difference of dPDS to SSI solutions without blockchain is that users are actually in full control over their personal data [23]. This is made possible by the private-public key infrastructure on which blockchains are built upon [28]. Accounts on the blockchain are typically private-public key pairs of which the public key serves as an identifier on the blockchain. On Smart Contract-enabled blockchains like Ethereum, the public key can be used to collect and reveal personal data. The owner of the private key to the public key is then in full control of the data that was issued to the public key. Nobody without the private key, not even the dPDSs, can re-sell, access or remove the personal data. dPDSs use Smart Contracts to store the personal data in an encrypted or hashed form on the blockchain [29].

2.5.1 Decentralized Personal Data Services (dPDSs)

dPDSs use the combination of private-public key pairs and Smart Contracts to connect a real-world identity to a digital identity on the blockchain. Using Smart Contracts, these companies issue certifications for the digital identity that can be used for authentication and verification of personal information. Initially, dPDSs attest that the digital identity belongs to the real-world identity by checking the user's ID, passport, or drivers license. From that point onward, third-parties like banks, governments, or universities can add confirmations to the digital identity that it truly belongs to the real-world identity. By adding such verifications, the digital identity becomes more trustworthy given the multiple sources of verification [30].

Next to identity verification, some dPDSs like uPort offer the issuing of certificates or documents to the user's identity [29]. These certificates can be specified freely and contain any information that the user wishes to put on the blockchain. The hashes of the certificates are stored on the blockchain, which simplifies certificate verification. A verifier can simply create the hash of a received certificate and compare the created hash with the stored hash from the blockchain. If the hashes match, then the received certificate is authentic and hasn't been tampered with. Additionally, the verifier can check which real-world identity issued the certificate by comparing the public key that issued the certificate with a list of known public keys. These known public keys belong to known real-world identities, which means that the verifier can check easily which real-world identity issued the received certificate.

The user can keep a list of all certificates and pass on the identifier to any of these certificates to e.g. an exchange university for verification. The user's certificates cannot be linked to each other, which means that by passing on one certificate, the user does not reveal her possession of any other certificates, which improves her privacy. Instead of passing on a full copy of a certificate, the user can also choose to only pass on attributes (e.g. birth date, address, nationality) which are absolutely necessary for an application, while retaining personal data that does not need to be shared. By revealing only the minimum amount of personal data necessary, the privacy of the user is improved even further.

dPDSs like Civic and uPort typically store hashes of the personal data on the Ethereum blockchain, which improves fault-tolerance and availability of the systems, but could become a major problem of confidentiality if the hash algorithm used is broken and can be reverted someday in the future. In that case, all of the user's personal data would be readily available to anybody connected to the Ethereum network. Additionally, with the General Data Protection Regulation (GDPR) coming into effect on May 25th, 2018, the hashes of personal data stored on the open Ethereum blockchain become private data since hashing is only considered pseudoanonymization and not full anonymization by the GDPR [31]. Such private data are subject to the *Right to erasure*, which gives the user the right to request full erasure of her private data from a company like Civic. Since the blockchain is an immutable data storage, erasure is impossible, putting companies like Civic into a position where they cannot comply with GDPR law, making them vulnerable to fines of up to EUR 20 million [32].

2.5.2 Sovrin

Given the issues of SSI solutions like Civic and uPort, the American company Evernym took a different SSI approach with their product Sovrin (also called *Hyperledger Indy*). Sovrin, or "Indy", provides users with a SSI and certificate verification without storing personal data on the Sovrin blockchain [33]. Sovrin only stores meta and connection data on-chain and handles personal data off-chain. There are three key components [33] that make up the Sovrin system:

1. Connections
2. Claims
3. Proofs

Connections

Any interaction between two parties on the Sovrin network begins with setting up a **Connection**. For every connection, both parties generate a new private-public key pair, which is only going to be used for that particular connection. Both parties specify an endpoint via which they are available. They can specify a self-hosted endpoint or an Agent endpoint, which is a service that can interact with the Sovrin network on behalf of a user if she agrees to. This enables mobile devices with changing IPs to connect to the Sovrin network via an Agent that keeps a static IP. All meta-information about a Connection, which includes the public keys and the endpoints of the two parties, is stored on the Sovrin blockchain. Whenever a party sends information to the other party off-chain, she encrypts the information first with the public key that is stored on the blockchain and sends the data to the endpoint specified in the Connection.

Claims

In the Sovrin network, certificates are called **Claims**. Claims are based upon a **Schema** and a **Claim Definition**. Schemas specify the data types, attribute names and formats which are used by a Claim. Claim definitions are issued only by **Issuers**, which are nodes in the network with special issuing rights. A Claim Definition contains information about which Schema it uses, the issuers who published the definition, and the structure of the claim including which attribute names and types from the Schema are used in the Claim. Once the Schemas and Claim Definitions are published, users can either self-assert Claims to themselves or receive Claims from Issuers, whose real-world identity is typically known (e.g. Universities, Banks, Governments, etc.).

Proofs

Whenever one entity on the Sovrin network wants to request personal information from another entity, she first creates a **Proof Request** with the attributes that she wants to know. The inquired entity then fills in the requested attributes with information from her Claims and sends back the **Proof** to the Proof Request. The inquiring entity can verify the integrity and authenticity of the filled-in information using a cryptographic algorithm called *Idemix* or also *Bluemix* developed by IBM [34]. How Idemix works exactly lies outside of the scope of this thesis, but using the Idemix algorithm, the inquiring entity can verify the integrity and authenticity of the received information.

Whenever the inquired entity shares her personal data with the inquiring entity, a **Proof of Consent** is stored on the blockchain, which can be used as evidence for when and which personal data was shared. If the inquired entity decides to stop sharing the information, then a **Revocation** can be stored on the blockchain. Revocations contain the relevant claim definition, and any attributes that are revoked. Attributes are not directly stored on the blockchain, but rather "accumulated". The accumulated version of the attributes can be used to check whether an attribute is revoked, without disclosing any of the other attributes in the Claim or in the Revocation [33].

Sovrin vs. Civic and uPort

By keeping personal data off-chain, Sovrin argues to be GDPR-compliant [35], which would give Sovrin an edge over the approach of Civic and uPort. The verification process is more complex and obscure with Sovrin than the simple hash comparison of Civic and uPort. However, since less data is stored on the blockchain overall, Sovrin is more scalable than the Ethereum-based alternatives. Sovrin can only be run on a permissioned blockchain, whereas Civic and uPort can use the Ethereum blockchain.

2.6 SSI Solutions with vs without Blockchain

The key difference between SSI solutions built with or without blockchain is the full control that the user has over her personal data. Without blockchain, the user is ultimately dependent on the integrity and goodwill of the SSI service [23]. Even if the SSI service does not become malicious, it is still a single point of failure, both for system availability and data security. When a PDS system is compromised, an attacker can easily and quickly retrieve vast amounts of personal data since data is stored centrally and mostly shares a single or a few encryption keys. With decentralized, blockchain-based solutions, an attacker could not gain access to many different identities at once since every single one is controlled by a different key pair. Therefore, attackers can still target and compromise the personal data of individuals, but stealing personal data from a large group of individuals becomes much less feasible.

Regarding the main principles of SSI, solutions using the blockchain can offer higher levels of principle fulfillment than solutions without the blockchain. This difference depends on the solution design, but technically solutions without blockchain are able to gain access to the user's data without her consent and these solutions can restrict provider migration more easily than SSI solutions using the blockchain. Therefore, technically speaking, blockchain-based systems are better suited to create SSI solutions than central systems ([22], [36]).

Now that the different SSI solutions with and without blockchain and were discussed, the thesis will explore different blockchain technologies and their ability to support SSI solutions on them.

2.7 Overview of Blockchains

In the following, four blockchain projects are introduced and analyzed regarding their ability to enable SSI. Two permissionless and two permissioned blockchains were chosen and a minimal SSI was implemented to evaluate their maturity and ease of development. The blockchains were chosen based on their market adoption and developmental maturity.

2.7.1 Ethereum

In an attempt to improve Bitcoin's limited abilities to run scripts, a team of developers lead by Vitalik Buterin created Ethereum as an alternative to Bitcoin's blockchain. Ethereum was initially described in Buterin's white paper in 2014, funded by a public crowdsale of its cryptocurrency called *Ether* later that year, and launched in 2015 [37]. Its main goal is to offer a distributed and trustless platform on which Smart Contracts can be run [38].

Technical Overview

Ethereum uses a permissionless blockchain that creates a new block approximately every 15 seconds. Ethereum's cryptocurrency is called *Ether* and can be used to pay for Smart Contract deployment and execution. The price of deploying and executing a Smart Contract is paid in *Gas*, which is a secondary currency that can be bought with Ether. The gas price is specified within the deployment transaction and can vary depending on the demand for block inclusion in the Ethereum network. Ethereum uses the Proof-of-Work consensus protocol at the time of writing but is planning on implementing a Proof-of-Work/Proof-of-Stake hybrid consensus protocol called *Casper* in 2018 [39],[40].

Five programming languages can be used to write Ethereum Smart Contracts: Solidity [41], Serpent (deprecated) [42], Vyper [43], Mutan (deprecated) [44], and Low-level Lisp-like Language (LLL) [45]. Of the five languages, Solidity is the most used and accessible one [46], whereas Vyper was designed for readability and security. The least-often used language LLL is an Assembly-like language that gives full control over the low-level operation codes of the Ethereum Virtual Machine (EVM). Serpent and Mutan are deprecated for either their lacking safety protections [47] or because they were replaced by Solidity [48]. All languages compile to byte-code that can then be executed in the EVM.

Maturity

Of the presented blockchain technologies, Ethereum has by far the most frameworks that facilitate and support development for the EVM. Development frameworks exist for almost every major language like JavaScript/Node.js (Truffle), Python (Populus), Java (EthereumJ), and C# (Nethereum). The Ethereum blockchain offers an API that can be used with the *Web3* framework, which is a wrapper for the Ethereum API. Versions of the Web3 framework are available for Javascript (Web3.js), the mighty Python (Web3.py), Java (Web3j), and C# (Nethereum).

To the author, the Ethereum community appears to be very active and global and a few companies even offer certified development courses (ConsenSys Academy, B9lab Academy). The official forum is very active and the dialog between the Ethereum development team and its community seems very active as well. The GitHub repository of the Ethereum project appears to be very active as well with 68 merged Pull Requests, 75 commits by 27 authors, and 4 releases within the period of one month (13th of May - 13th of June, 2018).

SSI on Ethereum

Ethereum's Smart Contracts serve as a good base for SSIs. Projects like uPort and Civic built Smart Contracts that hold verifiable information about a user on the publicly available Ethereum blockchain. The typical approach for SSIs on Ethereum is to store a hash of personal data on the blockchain against which a hash of received data can be compared. If the hashes match then the received personal data is verified. This approach facilitates data integrity since the hashes will not match if the received personal data was tampered with, and also facilitate authenticity since the verifier can check by whom the hash on the blockchain was issued. If the data on the blockchain was issued by a public key known to belong to a real-world identity (e.g. a University), then the verifier can be certain that the received data was issued by the real-world identity and nobody else.

Although only a hash of personal data is stored on the Ethereum blockchain, SSI solutions on Ethereum diminish confidentially in three ways. First, the same hash on the blockchain will serve as proof for every verifier, who is checking received personal data. This means that e.g. two independent verifiers will use the same hash to verify the data they received. The proof is not unique per verification. This can pose a problem if a verifier becomes malicious and forwards the personal data together with the proof to an unauthorized third-party. The third-party can use the same proof to verify the data integrity and authenticity of the received data without the knowledge of the data owner. In theory, the user gives up her control over her personal data every time when she shares her personal data since the shared-with party gains full control and access to the personal data as well.

The second problem is that even though only hashes of personal data are stored on the blockchain, pseudo-anonymous data can be used to re-identify the owner since typically all personal data hashes are issued to the same public address of the receiver. Therefore, if the user reveals a single hash to a verifier, the verifier can check whether other data has been issued to the same address and in the worst case, demand the extra information as well. This issue could be mitigated by creating a new public address for every issued claim, but the separation between claims is nullified if the user has to send proofs using two or more claims, effectively admitting that the two addresses to which the claims were issued belong to the same user.

Eventually, since the hashes of personal data are stored on a public and immutable blockchain, the hashed data is accessible to every node in the network. As the data is hashed, it is unreadable for anybody but the owner, however, hash algorithms tend to be "broken" every couple of decades [49]. This means that the hashed data on the blockchain is only secure until the used hash algorithm can be reverted and the personal data can be guessed successfully. Since the Ethereum blockchain is immutable, all data that was previously deemed safe since hashed would be freely accessible to anybody with enough computational power. This issue can be mitigated by using stronger hash algorithms when storing the data, but this workaround does not address the fundamental issue that stored publicly data will become available to everybody eventually.

Ease of Development

For testing the maturity of the Ethereum development environment, a basic SSI Smart Contract was developed in Solidity v0.4.24 and deployed on a local Ethereum blockchain run with TestRPC. The Smart Contract can be found on GitHub. Development instructions were retrieved from the Solidity Documentation and Populus v2.2.0 was used to facilitate testing and deployment of the Smart Contract. In the Smart Contract, the user can create, retrieve, update, and delete claim hashes. The document hash can be issued by a known entity (e.g. a University) and used by any verifier to verify received data against the stored hash.

The development processes were very simple and straight-forward, particularly setting up the local development environment with Populus and TestRPC. The development language, Solidity, was very intuitive and without complexity. However, given the author's experience with Solidity, the ease of development cannot be compared objectively against the ease of development with the other blockchains.

2.7.2 NEO

NEO was created around the same time as Ethereum in 2014/15, first under the name of *AntShares*. Its focus lay on creating a scalable blockchain that is Smart Contract-enabled and could be used to create a so-called "Smart Economy" in which business transactions are handled automatically on the blockchain. NEO supports "Digital Assets", which are digital representations of real-world assets like funds, machines, or goods. On NEO's blockchain, digital assets can easily be controlled by Smart Contracts that can handle e.g. tracking the ownership or the physical location and supply chain of real-world assets.

Technical Overview

NEO's blockchain is quite similar to Ethereum's with a short block time of 15 to 25 seconds and the division of currency (NEO) and execution costs (NeoGas or Gas). NEO's blockchain is permissionless as well, which means that anybody can join, read, and interact with NEO's blockchain. Unlike Ethereum, where no maximum amount for Ether exists, the total amount of NEO ever being created is limited to 100 million, 50 million of which were created during the initial public offering and with 15 million added to the network every year until the ceiling of 100 million is reached [50].

In NEO's network, a predefined amount of Gas is created for every block that is added to the NEO blockchain. Initially, 8 units of Gas were created for every block, however, the amount of Gas is decreasing by 1 unit every year until only 1 unit of Gas is created per block. That rate of 1 Gas is kept for 22 years until 100 million units of Gas were created. At that point, no more Gas will be created. The units of Gas created with each block are distributed evenly over the NEO token in circulation. Therefore, Gas can be acquired via this distribution or by buying it from other NEO token holders.

Given the scarcity of Gas and the predefined prices for certain transactions [51], using the NEO blockchain is prohibitively expensive. Deploying a Smart Contract onto the NEO blockchain can cost between 100 and 1000 Gas depending on what functions (e.g. storage, dynamic calling) are required. At the time of writing (June 2018), one unit of Gas costs 13.95 USD. Therefore, deploying a Smart Contract to the NEO blockchain costs between $100 * \$13.95 = \$1395$ and $1000 * \$13.95 = \$13.950$. For comparison, deploying a Smart Contract to the Ethereum blockchain costs around $41000 Gas * 7 Gwei = 287.000 Gwei = 0.000287 ETH = \$0.139$, which is around 1/10.000th of the costs of deployment in the NEO network.

Smart Contracts for the NEO network can be written in either C#, Java, C/C++, JavaScript, and Python. Contracts written in either language are compiled to the instruction set that can be executed on the Neo Virtual Machine (NeoVM). NEO uses a *delegated Byzantine Fault Tolerance (dBFT)* consensus mechanism that is theoretically able to process 10.000 transactions per second, which is a claim that has not been proven yet by the NEO project. In dBFT, network participants delegate their voting rights to a few nodes called "Bookkeepers". These bookkeepers vote on which block to add to the NEO blockchain in a byzantine fault-tolerant manner [52], [50].

Maturity

The NEO community appears to be very active and is organized within the City of Zion community that regularly hosts hackathons, coding competitions, and member meet-ups. The communication between the community and the development team appears not too active to the author. The community seemed to be split into an Asian community and a Western community. Given that the development team resides in the Asian community and the author only researches the Western community, an objective assessment of the communication between the developers and the community cannot be given. The NEO ecosystem hosts considerably fewer companies than the Ethereum ecosystem, but this assessment might again be influenced by the fact that the Asian community is not properly taken into account.

The development environment of NEO seems to be advanced to the author, but less advanced than the Ethereum ecosystem. Only the SDK for C# offers a convincing set of functionality, whereas the Python and Java SDKs are still under heavy development. The fact that the NEO network went offline after a single Node disconnected during block creation supports the notion that the NEO network is not yet production ready [53].

SSI on NEO

Developing SSIs for the NEO blockchain faces the same difficulties as on the Ethereum blockchain. Data integrity and authenticity can be ensured, but the Smart Contract lacks confidentiality for all the reasons as described before. The NEO project claims to have a special kind of asset, called a *Contract asset* that possesses a private storage area, which is inaccessible to anyone but the Smart Contract possessing it. This private storage could be used well for storing and managing access to personal data or to create zero-knowledge proofs with that data. However, despite the author's efforts to gather more information about contract assets, no explanation about how these contracts work could be found in the documentation, the code, or in the user forums.

Ease of Development

The author developed the same Smart Contract as for the Ethereum blockchain in Python, which can be found on GitHub. Given the author's experience with Python and the thorough documentation [54] of the NEO-Python project, it was straightforward and quick to develop the Smart Contract for NEO. A local NEO blockchain was used to deploy and test the Smart Contract. Setting up the local development environment was uncomplicated and swift.

An impediment of writing NEO Smart Contracts in Python was the beta state of the NEO-Python Software Development Kit (SDK), which offered only a limited set of functionality. Common language features like classes, switch-statements, and list comprehensions were not implemented. Also, a constructor function was missing, which was used in the Ethereum Smart Contract to set the owner of the Smart Contract. In NEO, the owner had to be hard-coded into the Smart Contract, which diminishes the dynamic of developing Smart Contracts for NEO. Since the NEO project is developed in C#, the C# SDK is further developed than the Python SDK, but it hasn't been properly tested by the author, which means that a comment on its quality cannot be given.

2.7.3 Hyperledger Fabric

In order to combine companies' needs for confidentiality with the potential for data integrity and authenticity of the blockchain, IBM created the Hyperledger Fabric project in 2016. According to IBM, Hyperledger Fabric offers a modular blockchain architecture combined with high degrees of confidentiality, resiliency, flexibility, and scalability [55]. It is primarily seen as an enterprise solution for business communication and collaboration. Hyperledger Fabric is developed by IBM in collaboration with the Linux Foundation and around 30 other organizations [56].

Technical Overview

Hyperledger Fabric offers a permissioned blockchain, which indicates that the blockchain is not publicly accessible and nodes can only join upon invitation. Nodes are of one of three types: Client, Peer, or Orderer. Client nodes are end-user nodes that cannot connect to the blockchain directly. They keep the private/public key pairs of the user, but in order to broadcast transactions to the network, they need to connect to a Peer. A peer node stores the blockchain, can broadcast transactions, and receives new blocks from the Orderer nodes. The Orderers are dedicated nodes that collect transactions and add them to the blockchain.

Transactions can be created by a Client or a Peer node and need to get endorsed first. After creation, a transaction is first sent to one or multiple endorses nodes, which are dedicated Peer nodes, which check the transaction and sign it if the transaction complies with the endorsement policies. Endorsement policies are pre-defined and updating or addition of policies is not allowed but might be supported in the future [57]. If a transaction receives enough endorsements, the transaction can be sent to one or multiple Orderer nodes, which include the transaction in a block and add it to the blockchain. New blocks are broadcasted to every Peer node, which in turn update their local blockchain with the new block. The target time between blocks can be configured and can range from seconds to minutes or hours if preferred.

Hyperlegder Fabric offers a platform for Smart Contracts as well, which need to be written in Golang. Smart Contracts, or how IBM calls them: Chaincode, can be deployed to the Hyperledger Fabric network just as in the Ethereum or NEO network, however, no fees apply when creating Smart Contracts. The state of Smart Contracts is not accessible to other Smart Contracts unless specified otherwise. This means that Smart Contracts cannot interact with each other unless specifically permitted to.

One feature that makes Hyperledger Fabric stand out from other blockchains is the possibility to create *Channels* between network nodes. Such channels can be used to securely and confidentially transfer data or funds between network participants without the knowledge of other participants. This feature enables Hyperledger Fabric to comply with the requirements for confidentiality that businesses typically pose.

Maturity

Hyperledger Fabric is in development since mid-2016 and has received stable releases in mid-2017 with version 1.0 and 1.1 in early-2018. According to IBM, the two releases contain already much of the envisioned functionality and can be used in production. Release 1.2 is scheduled for June 2018, which shows that the Hyperledger Fabric project is well maintained, but still under heavy development.

The development is spearheaded by IBM and in collaboration with around 30 other companies. Since Hyperledger Fabric is an enterprise solution, it does not enjoy a vibrant open source community of volunteers or frameworks and extensions that were developed independently of the consortium of companies.

SSI on Hyperledger Fabric

Given that Hyperledger Fabric was developed with flexibility in mind, its Smart Contract platform can be used just as freely as Ethereum's or NEO's. This means that the same sort of Smart Contracts can be developed for Hyperledger Fabric's blockchain as for e.g. Ethereum's. There are two major differences to permissionless blockchains though:

First, the fact that only approved nodes can read and write to Hyperledger Fabric's blockchain reduces the risk that unauthorized third-parties gain access to personal data as it was discussed for Ethereum or NEO. Since the real-world identities of the network participants are known, a malicious actor who forwards personal data to unauthorized third-parties can easily be identified and prosecuted.

Secondly, since Hyperledger Fabric offers the creation of channels between parties, personal data can be transferred confidentially and discretely between network participants. Even further, the channels can be "closed" or deleted after the transfer has been completed, which means that no personal data will be stored on-chain for longer than necessary.

Ease of Development

The author created the same Smart Contract as for the previously discussed blockchains. The code can be found on GitHub. The development of the Smart Contract itself was rather straight-forward and uncomplicated given the very helpful documentation of Hyperledger Fabric.

Setting up the local development environment, however, was rather tedious and intricate. The author was not able to deploy and test the Smart Contract in a reasonable amount of time. The configuration of the local network was rather convoluted and not easy to oversee. There was no obvious option to test the Smart Contract automatically, making verification of the soundness of the Smart Contract difficult.

2.7.4 Sovrin/Hyperledger Indy

Dedicated identity solutions like the aforementioned Civic and uPort operate on the existing and independently developed Ethereum blockchain and are therefore ultimately dependent on the generic infrastructure that Ethereum provides. Since the provided blockchain of Ethereum does not offer much confidentially, all solutions built upon it can offer only limited confidentiality as well. The company Evernym took a different approach by building their own blockchain that is tailored towards offering the necessary features to provide a SSI. Their product was originally called *Sovrin*, but attained the name *Hyperledger Indy* since its admission to the Hyperledger program in May 2017 [58]. In the following, the Sovrin project will be referred to by *Hyperledger Indy* or in short as *Indy*.

Technical Overview

The Indy blockchain was designed around the concept of a Decentralized Identifier (DID). DIDs are comparable to traditional identifiers like table rows, public keys, IP addresses, or email addresses. However, they are not stored centrally but decentralized on a blockchain. Therefore, they cannot be revoked, removed, modified, or otherwise made inaccessible. DIDs are also not controlled by a central authority, but rather by individuals, which is a fundamental basis for SSIs.

A DID resolves to a DID Descriptor Object (DDO), which contains metadata in a simple JSON format that proves the ownership and control over a DID [59]. Particularly, the DDO contains machine-readable descriptions of the owner's public keys and endpoints via which the owner is available. Endpoints could be IP addresses or URLs or any other form of contact information. The Public Key Infrastructure (PKI) which is created by storing DIDs on the Indy blockchain can be used to establish encrypted peer-to-peer connections between two parties. Personal data can be exchanged via these connections and is never stored on the Indy blockchain itself, which greatly improves the confidentiality and scalability of the blockchain.

The Hyperledger Indy blockchain uses IBM's Identity Mixer (Idemix) for issuing and verification of personal data. The math behind Idemix is intricate and lies outside of the scope of this thesis, but in summary, Idemix allows for authentication and proof of data integrity without revealing any unnecessary data [60]. The technical combination of DIDs with Idemix allow Hyperledger Indy to offer high levels of confidentiality, authenticity, and data integrity.

The Hyperledger Indy blockchain is a permissioned blockchain, which can be made publicly available if needed, and it assigns network nodes to certain roles. The most fundamental role is the *Steward* role, which manages a full copy of the blockchain, can assign roles to other nodes, and add *Trust Anchors* to the network. Trust Anchors can issue and verify claims and add accounts, which are labeled *Identity Owners*. Identity Owners are the end-users of the system, who can receive and store claims, create proofs for proof requests, and create new connections. Identity Owners cannot issue claims for others, but only self-assured claims, which contain data about themselves.

The Indy network runs the Redundant Byzantine Fault Tolerance (RBFT) consensus algorithm in which a pre-defined Master node creates new blocks and broadcasts them to the network [61]. Backup nodes create new blocks themselves and monitor the Master node continuously. Whenever the Master node becomes malicious and creates illegal blocks, the Backup nodes replace the Master node. A Master node can also be replaced if its performance drops below a specified target performance. Thus, if a Master node becomes corrupted, malicious, or damaged, the Backup nodes elect a new Master node and continue its work. The consensus algorithm is run by the Steward nodes in the Hyperledger Indy network.

Maturity

Given that the Sovrin/Hyperledger Indy product was started only two years ago in 2016 and became part of the Hyperledger project in 2017, it has already produced a working product and partnered with over 58 partners [62]. The Indy community is largely professional and lead by the Evernym development team. Large community projects are still scarce in the Hyperledger Indy environment. The largest project currently developed on Indy is the Verifiable Organizations Network (VON) developed by the University of British Columbia, which tries to build a platform on which organizations can share information and data with each other. The Indy project is still under heavy development and latest versions often include breaking changes and small to medium design changes, which makes Indy usable, but still work-in-progress.

SSI on Sovrin/Hyperledger Indy

The Hyperledger Indy blockchain was designed and custom-built to host SSIs, which makes it the most promising solution for blockchain-based SSIs. SSIs are enabled by giving Identity Owners the full control and possession of their personal data all while keeping sensitive data off-chain. Data transfers and verifications are held off-chain as well, providing Identity Owners with the high levels of confidentiality and privacy. Data sharing is recorded on the blockchain and can be used to hold parties accountable if they decide on forwarding personal data to unauthorized third-parties. Issued claims can be revoked, which makes the Indy issuing system more dynamic than static solutions like Civic or uPort, which don't support revocation of issued claims.

Given that Hyperledger Indy was built with a special use case in mind, it is significantly less flexible than the previously discussed blockchains. Most of the functionality and business logic needs to be built outside of the blockchain since Smart Contracts are not supported in Indy. Since business logic is run on centralized servers instead of on-chain introduces a high-level of trust into the centralized system again, particularly if an Identity Owner needs to use a centralized system to access and use the Indy blockchain.

Ease of Development

Developing for the Hyperledger Indy blockchain was significantly more difficult than for the previously mentioned blockchains. The Evernym development team offered Java, Python, and C# SDKs, but these wrappers were very low-level and it was unclear how to use them. During the initial development for Indy, documentation for the SDKs was limited and the processes for e.g. issuing and verifying claims were obscure. Even after extensive communication with the development team, it was difficult to understand the process flow of Hyperledger Indy. The fact that business logic was not centralized in e.g. a single Smart Contract meant that the processes of multiple backends had to be connected and synchronized. Additionally, it was unclear when and where calls to the blockchain had to be made. Setting up the development environment, however, was facilitated tremendously by the Dockerfiles provided by the Evernym team.

2.7.5 Summary of Blockchain Overview

As the analysis above already indicates, existing blockchain solutions differ significantly regarding their technologies and security models. Each blockchain applies to specific use cases that seldom overlap with the use cases of other blockchains. Therefore, a thorough analysis of the requirements of a project has to be conducted in order to choose a blockchain that matches the requirements appropriately.

Chapter 3

Use-Cases & Requirements

3.1 Scenario Description

The StudyBits project focuses on students from University of Groningen (RUG), who want to go abroad for an exchange and apply through the Erasmus+ program. The StudyBits project aims to include all universities, which currently offer Erasmus+ exchange positions. However, the scope of this thesis will be limited to a single scenario. In particular, this thesis will focus on the user journey of a single student of RUG, who wants to go to University of Gent (UGent) in Belgium on an exchange. For narrative purposes, this student will be called "Lisa" in the following.

3.1.1 Scenario of this Thesis

This thesis focuses on the following scenario: Lisa is a student at RUG and would like to go on an exchange to UGent in Belgium. An employee of UGent creates an exchange position and specifies which requirements need to be fulfilled by an applying student in order for her to be eligible for the position. Lisa views the position on the StudyBits website. She sees the necessary requirements for the position and could apply through the website as well. Before she applies, Lisa retrieves her personal documents like a Proof of Enrolment or Transcript of Records from her origin university, RUG. These personal documents are stored locally in her personal wallet and are fully under her control.

Lisa decides to apply for the open position at UGent. She retrieves only the information from her personal documents, which are necessary to fulfill the requirements of the position. She sends these to UGent together with a proof that the information was handed out by RUG. UGent checks automatically whether Lisa fulfills the requirements, which she does. UGent accepts Lisa for the exchange position. Lisa views the progress of her application on the StudyBits website and is notified when she is accepted for the exchange position.

3.2 Use Cases

The described scenario includes a multitude of smaller use cases, five of which will be implemented during this thesis. First, a use case diagram is shown to visualize the relationships between actors and the use cases. Thereafter, the chosen five use cases are described.

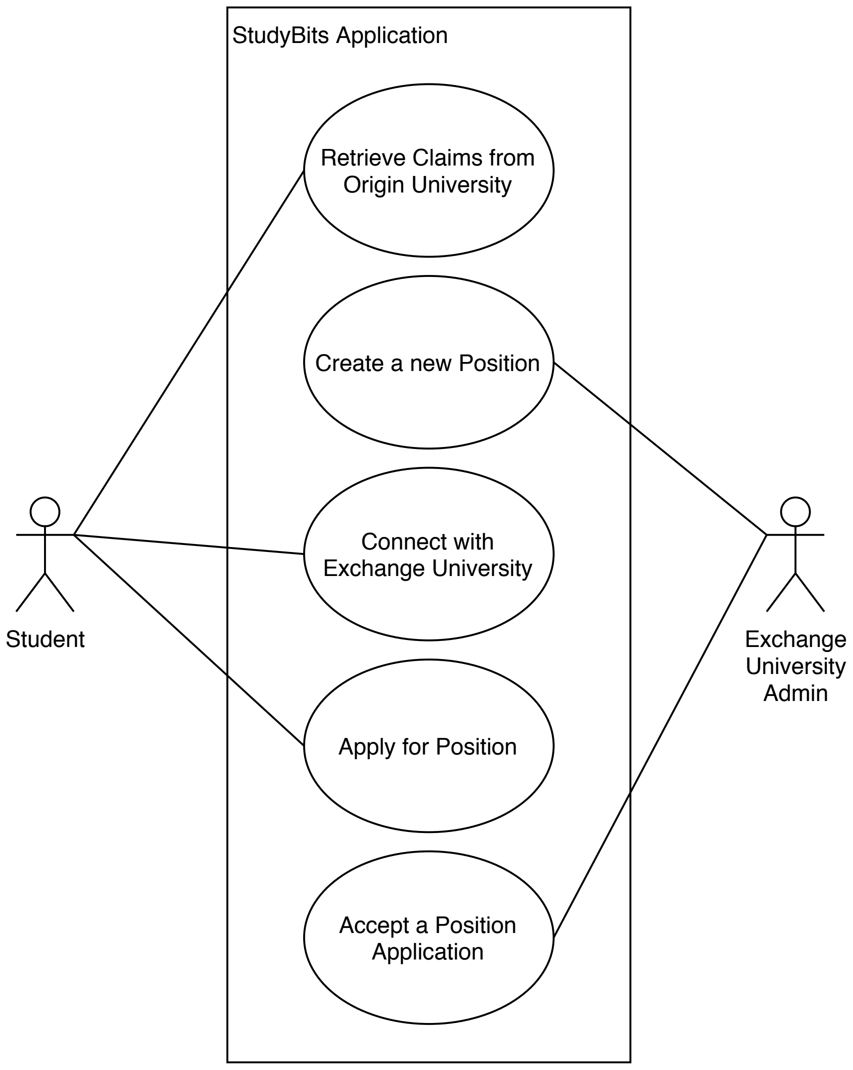

StudyBits Application
Retrieve Claims from Origin University
Create a new Position
Connect with Exchange University
Apply for Position
Accept a Position Application
Student
Exchange University Admin

Figure 3.1: Use case Diagram for the 5 use cases chosen

3.2.1 UC1: Student retrieves Claims from Origin University

Primary actor	Student
Scope	StudyBits Application
Level	Primary Task
Goal	Retrieve Claims from origin university and save them locally
Preconditions	1. Student is logged in. 2. Student is connected with Origin University. 3. Student account is confirmed by Origin University.
Main success scenario	1. Student navigates to “Claims” section. 2. Student clicks the “Update” button. 3. Application retrieves claims from Origin University. 4. Application saves claims locally. 5. Application displays all claims to Student.
Extensions	3a. Application cannot retrieve claims from Origin University. 3b. Application shows error message to Student. 3c. Use-case returns to Step 2.
Post conditions	New claims are stored locally and displayed to the Student.

Table 3.2: Use-case: Student retrieves Claims from Origin University

3.2.2 UC2: Exchange University creates a new Position

Primary actor	Exchange University Admin
Scope	StudyBits Application
Level	Primary Task
Goal	Create a new Position
Preconditions	1. Exchange University Admin is logged in.
Main success scenario	1. Admin opens the “Positions” section. 2. Admin clicks on the “New Position” button. 3. Application displays the Position Creation form to Admin. 4. Admin fills in the Requirements for the Position. 5. Admin clicks on the “Create” button. 6. Application creates a Proof Request for the Position. 7. Application saves the Position. 8. Application shows all Positions including the new Position to Admin.
Extensions	7a. Application cannot create a Proof Request for the new Position. 7b. Application displays error message to Admin. 7c. Use-case returns to Step 2. 8a. Application cannot save the new Position. 8b. Application displays error message to Admin. 8c. Application removes the Proof Request created in Step 7. 8c. Use-case returns to Step 2.
Post conditions	A new Position is created.

Table 3.4: Use-case: Exchange University creates new Position

3.2.3 UC3: Student connects with Exchange University

Primary actor	Student
Scope	StudyBits Application
Level	Primary Task
Goal	Connect with Exchange University and prove Identity
Preconditions	1. Student is logged in. 2. Student has performed UC1. 3. Student does not have a connection with Exchange University yet.
Main success scenario	1. Student opens the “Connections” section. 2. Student clicks on “New Connection” button. 3. Application displays Connection Creation form to Student. 4. Student selects the Exchange University. 5. Student clicks the “Connect” button. 6. Application creates new Connection with the Exchange University. 7. Application automatically proves Identity Proof Request from Exchange University. 8. Application stores the new Connection locally. 9. Application displays all Connections including the new Connection to the Student.
Extensions	5a. There are no Universities to connect with. 5b. Use-case ends. 8a. Application cannot prove the Identity Proof Request. 8b. Application displays error message to Student. 8c. Application removes the new Connection. 8d. Use-case ends.
Post conditions	Student is connected with Exchange University and Identity is proven.

Table 3.6: Use-case: Student connects with Exchange University

3.2.4 UC4: Student applies for Exchange Position

Primary actor	Student
Scope	StudyBits Application
Level	Primary Task
Goal	Apply for Exchange Position
Preconditions	1. Student is logged in. 2. Student has performed UC1. 3. Student has performed UC3.
Main success scenario	1. Student opens the “Position” section. 2. Application displays all Positions to Student. 3. Student selects the Position to apply for. 4. Student clicks on the “Apply” button. 5. Application retrieves the Proof Request for the Position from Exchange University. 6. Application proves the Proof Request using the Student claims. 7. Application sends proven Proof Request to Exchange University. 8. Application retrieves Position Application from Exchange University. 9. Application saves the Position Application locally. 10. Application shows Success message to Student.
Extensions	5a. Application cannot retrieve Proof Request from Exchange University. 5b. Application shows Error message to Student. 5c. Use-case returns to Step 2. 6a. Application cannot prove Proof Request. 6b. Application shows Error message to Student. 6c. Use-case returns to Step 2.
Post conditions	Student applied for Position and Position Application is created.

Table 3.8: Use-case: Student applies for Exchange Position

3.2.5 UC5: Exchange University accepts a Position Application

Primary actor	Exchange University Admin
Scope	StudyBits Application
Level	Primary Task
Goal	Accept a Position Application
Preconditions	1. Admin is logged in. 2. Exchange University has received a Position Application from Student. 3. Position applied for is still open.
Main success scenario	1. Admin opens the “Applications” section. 2. Application displays all Position Applications (PA) to Admin. 3. Admin selects the PA to accept. 4. Admin clicks the “Accept” button. 5. Application updates the status of the PA. 6. Application stores the updated version of the PA. 7. Application displays a Success message to Admin. 8. Application displays all PAs including the updated PA to Admin.
Extensions	6a. Application cannot store the updated version of the PA. 6b. Application displays an Error message to Admin. 6c. Application disregards the updated version of the PA. 6d. Use-case returns to Step 2.
Post conditions	Position Application is accepted.

Table 3.10: Use-case: Exchange University accepts Student for Exchange Position

3.3 Requirements

In the following, functional and non-functional requirements are extracted from the use cases. The ISO/IEC 25010:2011 standard for Systems and software Quality Requirements and Evaluation [63] is used for the extraction of the non-functional requirements.

3.3.1 Functional Requirements

Based on the use cases, the following functional requirements were extracted.

ID	**Priority**	**Description**
FR-1	**Must**	A Student must be able to retrieve and view her claims from her origin university.
FR-2	**Must**	An Admin must be able to create and view exchange positions.
FR-3	**Must**	An Admin must be able to specify the requirements for an exchange position.
FR-4	**Must**	A Student must be able to connect to an exchange university.
FR-5	**Must**	A Student must be able to prove her identity to the exchange university.
FR-6	**Must**	A Student must be able to apply to an exchange position.
FR-7	**Must**	A Student must be able to provide proof that she fulfills an exchange position's requirements.
FR-8	**Must**	An Admin must be able to accept or reject a student's application.
FR-9	**Must**	A Student must be able to view the state of her application.

Table 3.11: Functional requirements

3.3.2 Non-functional Requirements

Each described use-case has its own quality requirements. Therefore, in order to retrieve the system-wide requirements, the requirements that are most important to most use-cases are determined. The following table shows the requirements per use-case. Based on discussions that the author had with the product owner, the author gave out certain points for requirements, which are most important (30 Points), of medium importance (20 Points), of little importance (10 points), or not important at all (0 Points). The most important requirements are marked in bold. The least important non-functional requirements, Availability, was excluded from further analysis since it was considerably less important than the other three requirements.

	Confidentiality	**Integrity**	**Authenticity**	Availability
UC1	30	20	30	10
UC2	0	30	0	20
UC3	0	30	30	10
UC4	30	10	30	10
UC5	30	30	0	20
Total	90	120	90	70

3.3.3 Overall Non-functional Requirements

Based on the requirements extraction, three main requirements were chosen against which the blockchain technology will be evaluated. They are: *Integrity*, *Confidentiality*, and *Authenticity*. According to the ISO/IEC 25010:2011 standard [63], the definitions for the chosen requirements are as follows:

Integrity: *degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data*

Confidentiality: *degree to which a product or system ensures that data are accessible only to those authorized to have access*

Authenticity: *degree to which the identity of a subject or resource can be proved to be the one claimed*

In the following, these three non-functional requirements are used to lead the design decisions of the StudyBits project as well as the choice of blockchain technology which will be used during the implementation.

Chapter 4

Design

In the following, the design of the StudyBits project is presented. The design is based on the findings from chapter 2 and chapter 3. The first step in the design process is to choose a blockchain technology on top of which the StudyBits project will be implemented. The blockchain decision is then used as a foundation for designing the overall StudyBits system.

4.1 Blockchain Design Decision

The analysis in section 2.7 has shown that the four presented blockchain technologies fulfill the requirements of confidentiality, integrity, and authenticity to varying degrees. Table 4.1 shows to what degree the blockchain technologies fulfill the requirements extracted in chapter 3. Possible levels of fulfillment are **low**, **medium**, or **high** fulfillment. The following values were assigned by the author based on the prior analysis in section 2.7.

	Confidentiality	**Integrity**	**Authenticity**
Ethereum	low	high	medium
NEO	low	high	medium
Hyperledger Fabric	medium	high	high
Hyperledger Indy	high	high	high

Table 4.1: Blockchain Platforms' fulfillment of the non-functional requirements

Table 4.1 shows that the four blockchain technologies all fulfill the requirement for data integrity very well, however they differ in the levels of confidentiality and authenticity they can provide. The open and permissionless blockchains, Ethereum and NEO, both show only weak confidentiality since data saved on the blockchains is freely accessible for anybody, which might lead to severe data breaches in the future. Hyperledger Fabric offers a slightly higher level of confidentiality since the blockchain is permissioned and third-parties can be blocked from reading the blockchain, thus the stored personal data can be made inaccessible to unauthorized parties if needed. Hyperledger Indy shows the highest level of confidentiality since no personal data is stored on-chain at all. Additionally, data is sent exclusively through encrypted off-chain connections, which increases confidentiality even further.

The second requirement with varying degrees of fulfillment is authenticity. In permissionless blockchains, anybody can join and create an account without ever revealing his or her real-world identity. Thus, authenticity can be strengthened by collecting and correlating claims that verify parts of an identity, but the trustworthiness of the connection between on-chain and real-world identity stays probabilistic. On permissioned blockchains, identities have to be created by trusted and known network nodes, therefore the connection between on-chain and real-world identity is ultimately assured by the issuing node. The trustworthiness of an on-chain identity is directly correlated to the trustworthiness of the issuing node. The identities of nodes in a permissioned blockchain are typically known and enjoy a high-level trust. Therefore, the trustworthiness of issued identities is typically high as well, which allow permissioned blockchains to offer a high level of authenticity.

Based on the analysis summarized in Table 4.1, the decision to use **Hyperledger Indy** for the StudyBits was made by the author after discussions with Quintor B.V. The Hyperledger Indy project shows the best match with the StudyBits' requirements, which outweighs the projects immature code base. In the following, the system architecture is presented with the Hyperledger Indy blockchain at its core.

4.2 Architecture

The StudyBits project consists of three sub-systems. The university system, the student system, and the Indy blockchain. Both, the student and the university systems are connected to the Indy blockchain and store and retrieve information about DIDs, Schemas, Claim definitions, and the Proof of Sharing Consents [33]. The student system retrieves from and sends data to the university system, but not vice-versa. This simplifies the flow of information and prioritizes pulling over pushing of data. In the following sections, the design of the three systems is described in more detail. The 4+1 architectural view model [64] is used to give a holistic overview of the StudyBits system. In the following, the logical, process, development, and physical views are given. The scenario view was presented in Figure 3.1.

4.2.1 Logical View

The first view presented is the Logical view. A Blueprint diagram as described originally by Kruchten, 1995 [64] was chosen to represent this view. The diagram can be found in Figure 4.1. The blue classes are part of the Student and University Backend and the orange classes are part of the Indy Wrapper package. The blueprint diagram shows the relationships primarily between classes within the Indy Wrapper and their connection to the most important backend classes.

The Wallet Owner class comprises a focus point in the class hierarchy of the wrapper classes. The wallet owner serves as the main entry point to the more specific functionality of the Prover and Verifier classes, which implement functionality for fulfilling and verifying the responses to proof requests. A Wallet Owner is connected to a single Indy Wallet, which holds information about the main public key of the user.

The second main entry point is the Trust Anchor class, which has a one-to-one relationship with a university class. Universities can access the functionality of the Issuer class via the Trust Anchor class to issue claims and create student accounts. The Trust Anchor class can also be used to instantiate new connections between users, the information of which is stored in the Wallet Owner class.

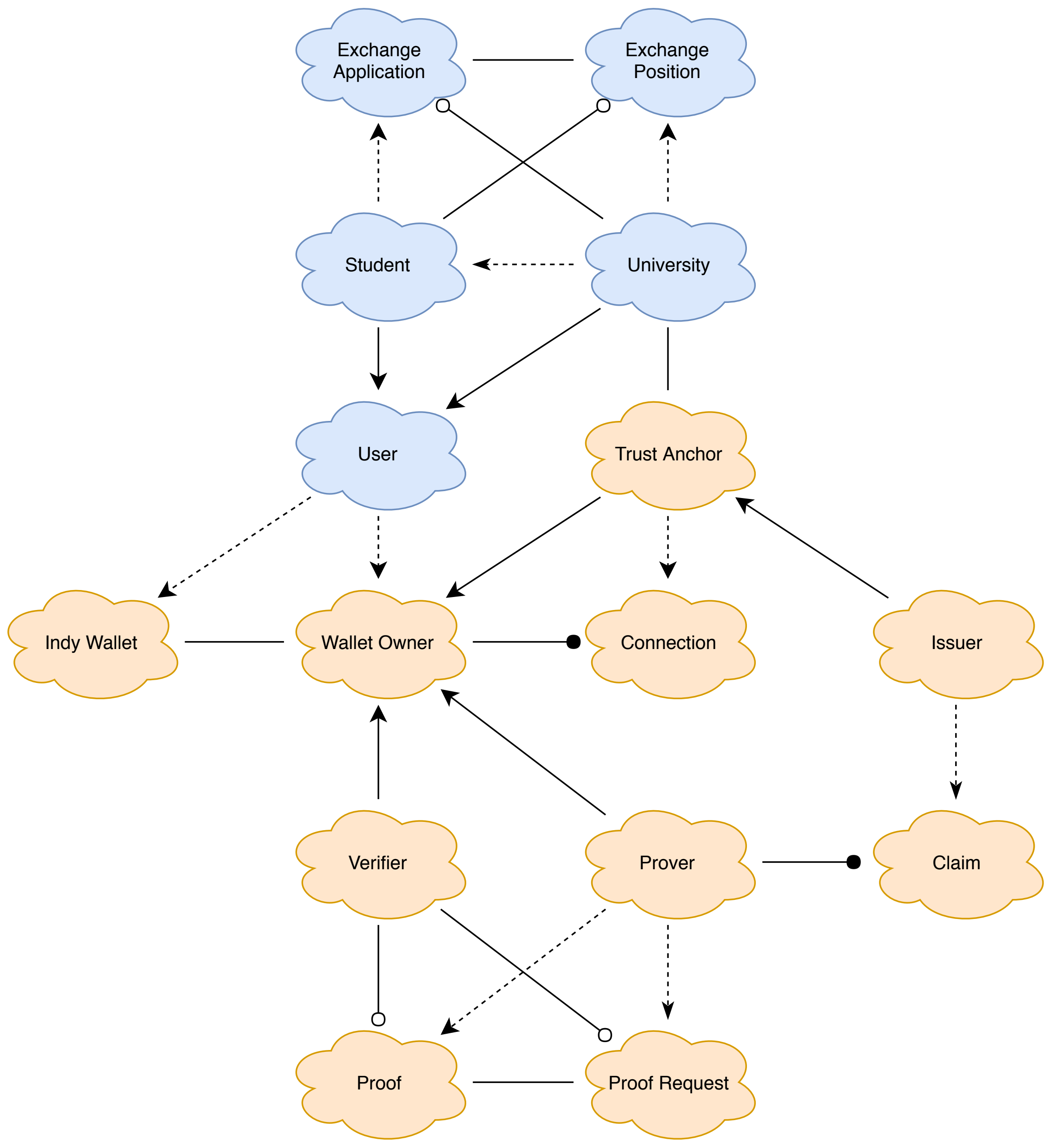


Figure 4.1: Blueprint diagram for the StudyBits Project

4.2.2 Development View

Figure 4.2 describes the different components that are part of the StudyBits system. The project consists of a common *Frontend-Backend-Database* stack with the additional connection to the Hyperledger Indy Blockchain via a Java wrapper. This Model-View-Controller design was chosen because it helps to separate the business logic, user interaction, and data storage nicely and makes it easy to secure and control access to the personal data stored in the database. The Indy Blockchain component is out of the scope of this thesis and is developed and maintained by the company Evernym.

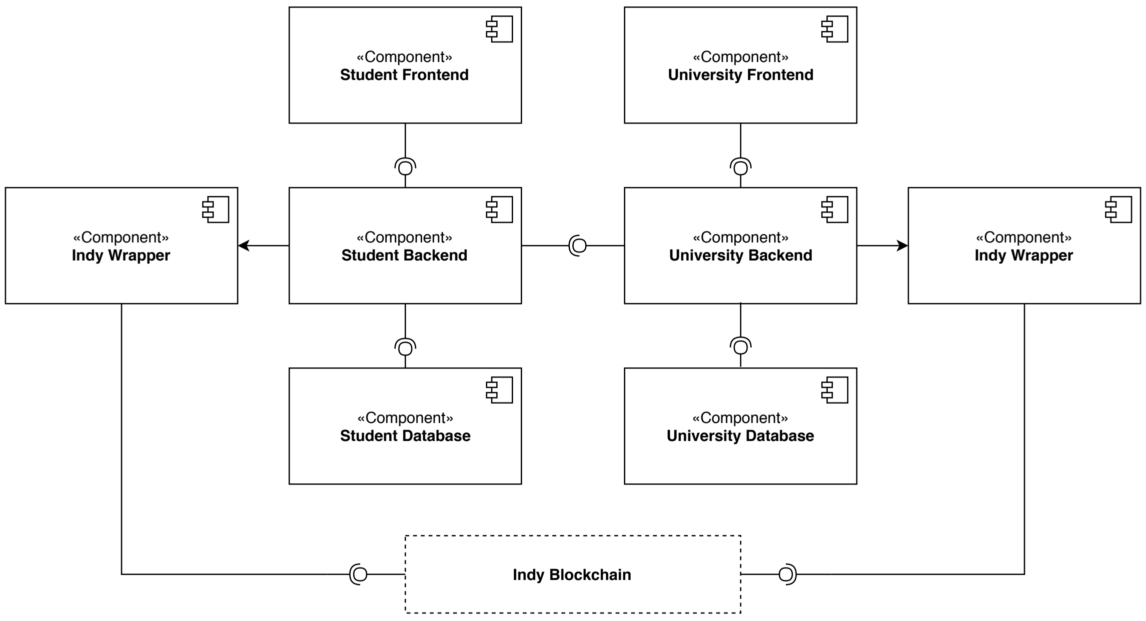


Figure 4.2: Component Diagram of StudyBits Project

The main connection in the system lies between the student and the university backend. Most information, like Exchange Positions and Applications, Claims, Proof Requests, and Connection Requests are handled through this connection. The communication is one-sided, meaning that only the student backend sends requests to the university backend and not vice-versa. This decision was made to simplify the flow of information in the system.

Both the student and university backends have a database to which they connect and where all data is stored, even the data that is also stored on the blockchain. This decision was made since it facilitates the development process and makes the system more efficient overall. The database is used to store the sensitive personal data, which should not be put on the blockchain, but also serves as a cache in order to speed up repeated requests to the backend.

Both backends serve a frontend through which the user can interact with the StudyBits system. Each frontend offers its own functionality tailored to the needs of the student or university user. The communication between the frontend and backend is one-sided for now, meaning that the frontend needs to call the backend API in order to receive data or updates. This was done to speed up the development process of the project for now. However, this could be improved in future releases with a websocket connection between frontend and backend via which the backend pushes updates to the frontend.

Eventually, each backend holds an identical Indy Wrapper component. This component abstracts away the complexity of communicating with the Indy Blockchain. The wrapper can be used to store and retrieve data from the blockchain and to create and fulfill Proof Requests.

4.2.3 Process View

In the process view, an activity diagram shows the usual flow of the StudyBits application: a student applies for an Exchange position, is accepted or rejected, and is informed of the result. This activity flow is based on the functional requirements and determines which functionality will be implemented as part of this thesis. Aside from the first three activities, which were marked with numbers from 1 to 3, all activities happen sequentially. The three marked activities can happen in any order but must have happened before the activity flow continues.

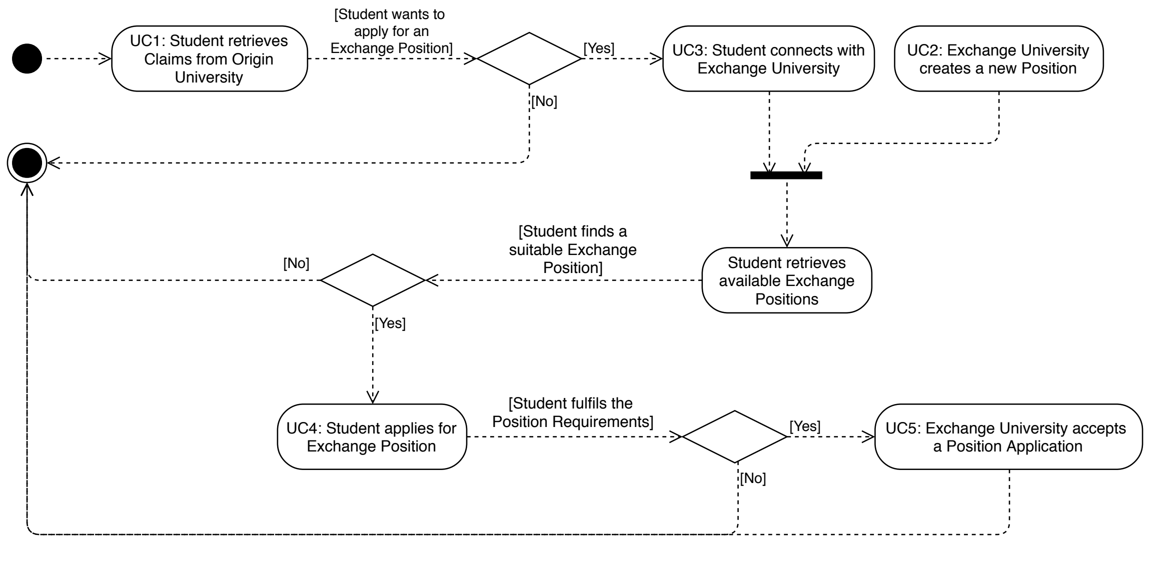

UC1: Student retrieves Claims from Origin University
[Student wants to apply for an Exchange Position]
[Yes]
[No]
UC3: Student connects with Exchange University
UC2: Exchange University creates a new Position
Student retrieves available Exchange Positions
[Student finds a suitable Exchange Position]
[No]
[Yes]
UC4: Student applies for Exchange Position
[Student fulfils the Position Requirements]
[Yes]
[No]
UC5: Exchange University accepts a Position Application

Figure 4.3: Activity Diagram for StudyBits Project

The activity diagram in Figure 4.3 can be used to illustrate how the subsystem of the StudyBits project, as defined in the Component diagram in Figure 4.2, work together. For simplicity, the Backend and Frontend components will be grouped and only referred to by *Student system* or *University system*. Activities marked in blue comprise communication between the Student, the (Origin or Exchange) University system, and the Indy Blockchain. Activities in red include communication between the Student system and the Indy Blockchain. Activities in orange are University system internal activities. System Diagrams were created to explain each activity in more depth. Each activity, with the exception for "Student retrieves available Exchange Positions", refer to a use case as described in Chapter 3. In the following, each system diagram is briefly explained and put into the context of the overall user journey of a student applying for an exchange position.

The user journey begins with the student retrieving her claims from her origin university. Figure A.2 displays the exact flow of this activity. The student instructs her system to retrieve the claims for her. Her copy of the student system then contacts the origin university, which fetches all information about available claim info for the student and sends it back to the student system. For each piece of information, the student system retrieves a Claim Offer from the origin university, which is filled in by the student system with the public key of the student. With the completed Claim Offer, the student system then retrieves the claim and stores a list of all claims in the local database.

The next step in the user journey is the creation of an Exchange Position by the Exchange University. The admin simply specifies the requirements for the new position and the university system stores the new position in its local database as can be seen in Figure A.1. Before the student can request a list of all exchange positions from the exchange university, she first has to create a connection with the exchange university. The exact flow of this activity is shown in Figure A.3. First, the student system requests a new connection request from the exchange university, which the university creates and returns to the student system. The student system creates a new private/public key pair for the connection, fills out the connection request with this information and sends the completed connection request to the exchange university. The university then creates a new key pair on its own, accepts the connection request with the new key pair and notifies the student system that the connection was successfully established.

Once the student selected an exchange position to apply for, the student system requests the proof request for the position from the exchange university, as can be seen in Figure A.4. The student system fulfills the proof request with the information stored in the claims which were previously retrieved from the origin university. The student system sends back the completed proof request to the exchange university fully encrypted with the key pairs, which were created for the connection with the exchange university. The university then decrypts the completed proof request, verifies the received information, notifies the student system about the result of the verification, and stores the application if the verification was successful.

In the last step of the application process, an admin of the exchange position selects an application from the list of all exchange applications and sets the application to either accepted or rejected as can be seen in Figure A.5. The exchange university system then stores the updated version of the exchange application in the local database. What is not shown in Figure A.5 is that the student fetches any updates from the university system and displays the application with the new status to the student. This concludes the user journey and the activity flow of the logical view.

4.2.4 Physical View

The components are grouped by their functionality and dependencies and each group is put into its own Docker image. As Figure 4.4 shows, the system is split into five Docker images, represented by boxes, which communicate with each other.

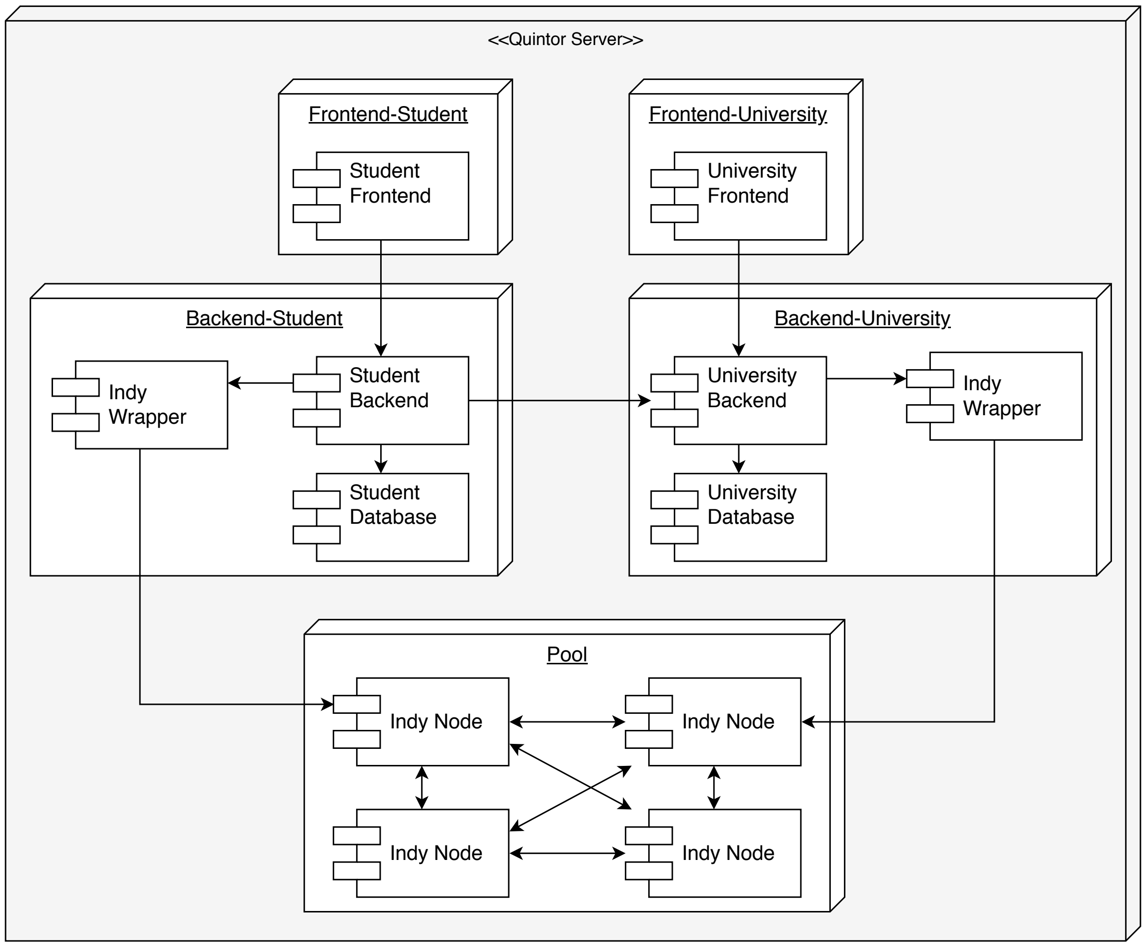


Figure 4.4: Deployment Diagram of StudyBits Project

The frontends were put into their own images since this way they can be quickly spun up and down depending on whether they are necessary for the current stage of development. Both backends received their own image since they have to be spun up and down independently of each other during development and because it increases the modularity and fault-tolerance of the overall system. Eventually, the Indy Nodes which represent the Indy Blockchain were put into their own Docker image, since this was the constellation recommended for development by the maintainer [65], Evernym.

As indicated in Figure 4.4, the Student application is designed to run on a Students mobile device for easy and ubiquitous access. The University application does not need to be ubiquitously available since it is likely to be used primarily from workstations. Therefore, each university can host their own application if preferred or a central service can deploy the university application once and make it accessible to all universities.

A node of the Hyperledger Indy blockchain is meant to be run by every participant in the system. However, to ease development and facilitate rolling releases, the blockchain infrastructure will be hosted by Quintor B.V. for now. Eventually, this infrastructure will be decentralized and each participant will run their own node. The student frontend and backend most likely will be extracted to a mobile platform, which would make the system easier to use and more portable.

The presented design will be used to implement a Proof of Concept of the StudyBits project based on the functional requirements, which were extracted in Chapter 3. In the following, the implementation details will be documented in greater detail.

Chapter 5

Implementation

In the following, an overview of the implementation process and technologies used is given. The implementation is conducted in collaboration with Quintor B.V. The author was embedded at Quintor in their headquarters in Groningen, The Netherlands, for the time span of this thesis. The StudyBits code base is open-sourced under the Apache-2.0 license on GitHub. Deployments of the StudyBits demo can be accessed via `https://studybits.quintor.nl/student` for the Student dashboard and `https://studybits.quintor.nl/university` for the University dashboard.

5.1 Development Specifics

Table 5.1 gives an overview of the technologies and their versions used for the development of the StudyBits project. The StudyBits project is developed using Java 8 and the Spring Boot framework. Ubuntu 16.04 is used as an operating system since the Hyperledger Indy core library, *libindy*, is developed particularly for this Linux distribution. The frontend is developed using Angular 5 with Angular Material components which allows rapid development.

The Java backends and the wrapper are packaged using Maven. All components, wrapper, backend, and frontend, are individually deployed using Docker containers. The StudyBits system is managed by a single docker-compose file, which can be found on GitHub. Integration tests using JUnit and REST-assured are used to verify the functionality and aid the collaboration between the developers. The integration tests can be found on GitHub as well.

Name	Version
Hyperledger Indy	1.4.0
Java	OpenJDK 1.8.0_171
Spring Boot	2.0.0.RELEASE
H2 Database	1.4.196
Dozer	5.4.0
Maven	3.3.9
JUnit	4.12
REST-assured	3.0.7
Ubuntu	16.04
Angular	5.2.0
Angular Material	5.2.4
Docker	18.03.1-ce
Docker-compose	1.20.1

Table 5.1: Overview of development technologies and their version number

5.2 Deployment Specifics

The StudyBits project is deployed using Docker images on a Virtual Machine hosted at the private cloud infrastructure by Quintor. The components were split into five different docker images, which are represented as boxes in Figure 4.4. All Docker images were run on the same virtual machine and made use of the host network to simplify the development and setup process. A docker-compose file managed the deployment of the different docker containers on the virtual machine.

The StudyBits project is made accessible via the frontends, which are available on `https://studybits.quintor.nl/student` for the Student dashboard and `https://studybits.quintor.nl/university` for the University dashboard. The URL routing on the virtual machine is handled by Apache. Table 5.2 gives an overview of the technologies and versions used on the virtual machine.

Name	Version/Specifics
centOS	7
CPU	1 x 2.1GHz Intel i5
Memory	3GiB DIMM RAM
Apache	2.4.6
Docker	18.03.1-ce
Docker-compose	1.21.0

Table 5.2: Overview of deployment technologies and their version number

5.3 Code Structure

The codebase is separated into two repositories, one for the backend code and one for the frontend code. Both repositories, Frontend and Backend, can be found on GitHub.

5.3.1 Backend

The backend code is structured into 4 modules. The *indy-wrapper* module contains the Java wrapper for the Hyperledger Indy library *libindy*. Libindy allows communication with the Hyperledger Blockchain programmatically, however offers only very low-level functionality, which is why the high-level *indy-wrapper* is developed. The indy-wrapper module does not contain dedicated unit- or integration-tests but tests its own functionality within the *main* function, which is executed only for testing purposes.

The student and university backends both have their own module called *student* and *university*. Both backends offer a REST API developed with Spring boot and communicate via these APIs. The modules both use the indy-wrapper module and contain business logic to interact with a local H2 database. For demo purposes, both backends seed example data upon startup, which contains students, university admins, connections, and claims.

The *integration-tests* module tests the communication between the student and the university backend and contains tests for every step in the StudyBits user journey. The integration tests are run whenever a commit is made to the GitHub repository using TravisCI. Pushing to the master branch is only permitted if the integration tests succeed. The indy-wrapper and integration-tests modules are ignored when StudyBits is deployed using the docker-compose file.

5.3.2 Frontend

The two frontend modules contain functionality for the student and the university dashboard. They communicate with the respective backends via the REST APIs offered by the backends. Each module is deployed in their own Docker container. The student and university Angular projects are run in development mode and do not offer production-grade security yet.

Frontend User Flow

The StudyBits system offers two dashboards, one for students, and one for universities. Figure A.6 visualizes the student user flow and Figure A.7 visualizes the user flow of exchange university admins using screen-shots from the actual (frontend) implementation. The typical user journey through these dashboards is briefly explained in the following.

First, a student logs into the student dashboard and reviews the accuracy of the issued claims, which can be viewed in the “Claims” tab. After review, the student creates a new connection with the exchange university of choice in the “Connections” tab. After an initial connection is created, the student proves her identity by submitting her first and last name and social security number to the exchange university.

After this initial step, the exchange university admin logs into the university dashboard and reviews the received information in the “Students” tab. After review, the admin creates a new exchange position in the “Positions” tab. The admin fills in the requirements for the new position and creates the position eventually.

After the creation of the position, the student accesses the open exchange position in the “Positions” tab. The student selects a position and applies for the position by clicking the “Apply” button. Using the Indy blockchain, the student backend checks whether the student user fulfills the position requirements. If this is the case, a new application for the selected position is created and all necessary application data is automatically shared with the exchange position.

Next, the exchange university admin reviews the application and either accepts or rejects the application by clicking the respective buttons in the “Applications” tab. The status of the application is updated and can be viewed by the student in the “Application” tab of the student dashboard. This concludes the user journey, which is considered by this thesis.

5.4 Team Communication

The StudyBits project was developed by the author together with two full-time developers employed by Quintor B.V. The author and one of the developer worked at the Quintor headquarters in Groningen, The Netherlands. The other developer worked from the Quintor office in The Hague, The Netherlands for two days a week. The author communicated in person with the developer in Groningen and during daily stand-ups with the developer in The Hague. Otherwise, the team communicated via Gitter and GitHub issues in the backend repository.

The CEO of Quintor, Johan Tillema, was the contact person at Quintor for the author and answered product related questions, however, the development team had great freedom for decisions on the design and implementation of the StudyBits system. Whenever needed, the development team reached out to Hyperledger Indy developers employed by Evernym via the dedicated Rocket Chat or via the Sovrin forum.

The work was distributed evenly between the three developers, where one Quintor developer worked on the indy-wrapper module, one on the university backend, and the author worked on the student backend as well as the two frontend modules. The deployment of the system was done by the author as well as the implementation of the functionality for exchange positions and applications in both student and university backends and frontends.

Overall, the communication between the developers and Quintor B.V. was successful and all deadlines were made in time. The initial version of the StudyBits project, without the exchange position and application functionality, was presented at the conference of the Groningen Declaration Network in Paris, France from 18th to the 20th of May, 2018. The final version of the StudyBits project as part of this thesis was presented on the 28th of June, 2018.

In the following chapter, the final version of the StudyBits project is evaluated based on the extracted requirements and the user experience of the overall system.

Chapter 6

Evaluation

The following chapter evaluates the implemented StudyBits system against the functional and non-functional requirements extracted in Chapter 3. The evaluation was done in collaboration with the developers of Quintor. A performance analysis of the system was considered by the author but was regarded as meaningless given the low expected workload and the demo-oriented deployment plan of the implementation. Therefore, a user experience study was conducted to evaluate the usability of the created system. The results will be presented and evaluated in the later parts of this chapter.

6.1 Requirements revisited

6.1.1 Functional Requirements

Table 6.1 revisits the functional requirements extracted in chapter 3 again and specifies to which extent the functional requirements were fulfilled. The degree of fulfillment was devised together with the development team at Quintor. Possible levels of fulfillments are: **Full**, **Partial**, or **Not implemented**.

ID	Priority	Description	Fulfillment
FR-1	**Must**	A Student must be able to retrieve and view her claims from her origin university.	Full
FR-2	**Must**	An Admin must be able to create and view exchange positions.	Full
FR-3	**Must**	An Admin must be able to specify the requirements for an exchange position.	Full
FR-4	**Must**	A Student must be able to connect to an exchange university.	Full
FR-5	**Must**	A Student must be able to prove her identity to the exchange university.	Full
FR-6	**Must**	A Student must be able to apply to an exchange position.	Full
FR-7	**Must**	A Student must be able to provide proof that she fulfills an exchange position's requirements.	Full
FR-8	**Must**	An Admin must be able to accept or reject a student's application.	Full
FR-9	**Must**	A Student must be able to view the state of her application.	Full

Table 6.1: Functional requirements with Degree of Fulfillment

As Table 6.1 shows, all functional requirements were fully implemented. Some requirements need to be extended in order to be production-ready for the StudyBits project. For example, FR-3 is implemented with pre-defined requirements fields and could be extended with optional and selectable requirement fields. FR-6 was implemented with strict string comparisons for checking whether requirements are met. This leads to the issue of failing requirement checks if the information issued in claims deviates from the required information by only e.g. a comma. A standard for format and names for the issued and required information could be devised in order to alleviate the issues with string mismatches. However, all agreed upon functionality was implemented during the course of this thesis.

6.1.2 Non-functional Requirements

The key drivers for the design and implementation of the StudyBits project were the three non-functional requirements extracted in chapter 3: **Confidentiality**, **Authenticity**, and **Integrity**. Table 6.2 shows the degree of satisfaction of each of the requirements. Possible levels of satisfactions are: **Full**, **Partial**, **Not satisfied**.

The levels of satisfaction were discussed and agreed upon with the development team at Quintor. The level of satisfaction for each requirement will be discussed and evaluated hereafter.

Requirements	**Satisfaction**
Integrity	Partially
Confidentiality	Partially
Authenticity	Partially

Table 6.2: Non-functional requirements and the level to which they were satisfied by the implementation

Confidentiality

The non-functional requirement of confidentiality is only partially satisfied, not because Hyperledger Indy or blockchain technology, in general, is not capable of offering a satisfactory level of confidentiality, but because the student sub-system is deployed to a Quintor owned virtual machine. Given that Quintor currently runs the virtual machine on their own servers, all personal data is currently stored in a simple SQLite database means that Quintor has full access to the data. However, the communication between backends is encrypted and cannot be read or altered by Quintor even if intended. Therefore, confidentiality is partially satisfied, because of the decision to host user databases on servers owned by Quintor.

Deploying the current StudyBits system to a central server allowed the developers to quickly create and present a demo of the StudyBits system, but the infrastructure needs to be eventually distributed over university-owned servers and students' mobile phones. Hyperledger Indy made the impression that it could serve high levels of confidentiality if distributed onto end-user devices properly. However, since this distribution has not been done during the thesis, the requirement for confidentiality could only be ranked as partially satisfied.

Authenticity

The requirement of authenticity is only partially fulfilled for the same reasons as confidentiality. Since Quintor hosts the current StudyBits system, it can create, modify, and remove user accounts at will. Created accounts cannot be trusted fully since they can be created by Quintor without external oversight. If the account creation would be more distributed over multiple actors and institutions and be connected to an ID check, then the level of authenticity could be considered satisfactory. Eventually, the level of authenticity will always depend on the level of trust one can put into the account creation and verification process. This process is independent of

Hyperledger Indy or blockchain in general and solely depends on the Trust Anchors to be honest and not to collude. If the fact that the system is currently hosted by Quintor is ignored, the requirement of authenticity can be considered as being fully implemented. Hyperledger Indy offers the appropriate level of authenticity if deployed correctly.

Integrity

The level of integrity in the StudyBits project was only partially satisfactory for the same reasons as for confidentiality. The databases in which personal data is stored are not under the sovereign control of the user herself, but hosted and therefore accessible and modifiable for Quintor. However, Hyperledger Indy increased the level of integrity since it relies on the Idemix framework, which, if used appropriately, cannot be tampered with, even if an actor has full control over a system. Therefore, integrity checks of the data issued cannot have been tampered with even though Quintor hosted all system on their servers.

One caveat is that data could have been corrupted before it was issued, meaning that all subsequent integrity checks would still pass, but the data itself is inherently incorrect. Both these issues of unrestricted data access and manipulation of data issuance can be mitigated by having end-users run and control their own StudyBits software, which is not part of the scope of this thesis. However, if the system would be deployed correctly, it would offer a level of integrity with which the integrity requirement would be fully satisfied.

6.1.3 Summary of Requirements

From the previous analysis, one can conclude that the functional requirements were fully fulfilled, but that the non-functional requirements were only partially fulfilled due to the deployment decisions made. Deploying to a central server diminished the levels of all three non-functional requirements, but allowed the development team to create an initial version of the system that can be used for demonstration purposes. The overall state of the system is not meant to be production-ready, which is why the levels of the non-functional requirements were not up to expectation yet. Future plans for deployments mitigated the aforementioned issues by running the software on user-owned devices, which would decentralize the system and revoke the unrestricted access of Quintor to the user's data.

6.2 User Experience Study

In order to evaluate the usability of the student and university frontends, a user experience study was conducted in which the participants judged the frontends on ease of use and intuitiveness. Additionally, the participants' knowledge and expectations for blockchain technology and SSI were inquired.

6.2.1 Methodology

The survey was conducted with in total 51 participants, 15 of which being employees of Quintor B.V. and the remaining 36 participants were undergraduate and graduate computer science or artificial intelligence students from the University of Groningen. First, the participants were briefly introduced to the concept of SSI and a short explanation of blockchain technology was given. This introduction was followed by an overview of the StudyBits project and its purpose. Then, the participants were introduced to the student and university websites/frontend by the author. The author walked the participants through the user journey of Lisa going to Gent.

It should be noted that the author presented the websites and conducted the user actions during the presentation, but that the participants had the opportunity to use the websites on their own, which most participants did to explore the website more. Eventually, the author answered any remaining questions and asked the participants to fill out the survey, which can be found in the Appendix in Table A.1.

6.2.2 Results

Initially, the results were analyzed for any differences between the responses of the Quintor employee group and the university student group. This analysis showed no significant differences in either familiarity with blockchain ($p = 0.335$, $df = 27$, $t = 0.982$), familiarity with SSI ($p = 0.290$, $df = 19$, $t = 1.088$), evaluation of the ease of use ($p = 0.386$, $df = 26$, $t = 0.882$) or intuitiveness of the frontends ($p = 0.675$, $df = 22$, $t = 0.424$). Given that the responses between the groups were not significantly different on the metrics of main focus, the groups and their responses were unified during further analysis.

To evaluate the knowledge level of the survey audience, the participants were asked to self-assess their familiarity with blockchain technology and SSI. As can be seen in Figure 6.1, the participants self-assessed their familiarity with blockchain technology significantly higher than their familiarity with SSI ($p = 0.014$, $df = 100$, $t = 2.503$).

A Spearman rank correlation was conducted to test whether the familiarity with blockchain or SSI was connected to a higher perceived usefulness of blockchain to enable SSI. The Spearman rank correlation was chosen since both metrics were collected using a discrete Likert scale, which means that the response values were heavily bounded and lacked the continuous nature needed for the Pearson correlation. The correlations showed only weak connections between the level of familiarity and perceived usefulness with $r_s = 0.362$ for blockchain familiarity and $r_s = 0.242$ for SSI familiarity.

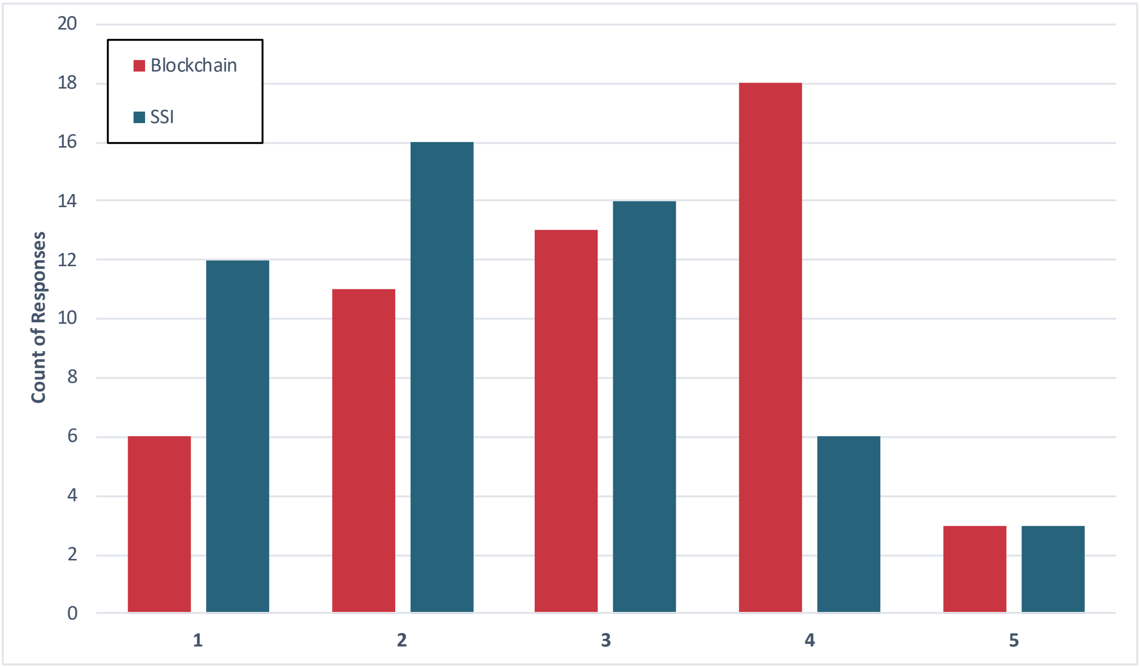


Figure 6.1: Participants' familiarity with Blockchain and SSI from 1=*Not at all* to 5=*Very much*

The surveys' primary purpose was to inquire about the ease of use and intuitiveness of the frontends, which was done in questions 3 and 4 of the survey. In general, both metrics were rated highly by the participants as can be seen in the appendix in Figure 6.2. A significant amount of participants rated the frontends with a 4 on a 5 point scale with 5 being *very much usable* or *very much intuitive*. The ease of use received significantly more 5 point evaluations than the intuitiveness of the frontends, which lead to the final ratings of, on average, 4.22 points for ease of use and 3.88 points for intuitiveness. The difference between the two metrics is significant with $p = 0.022$, $df = 100$, $t = 2.336$.

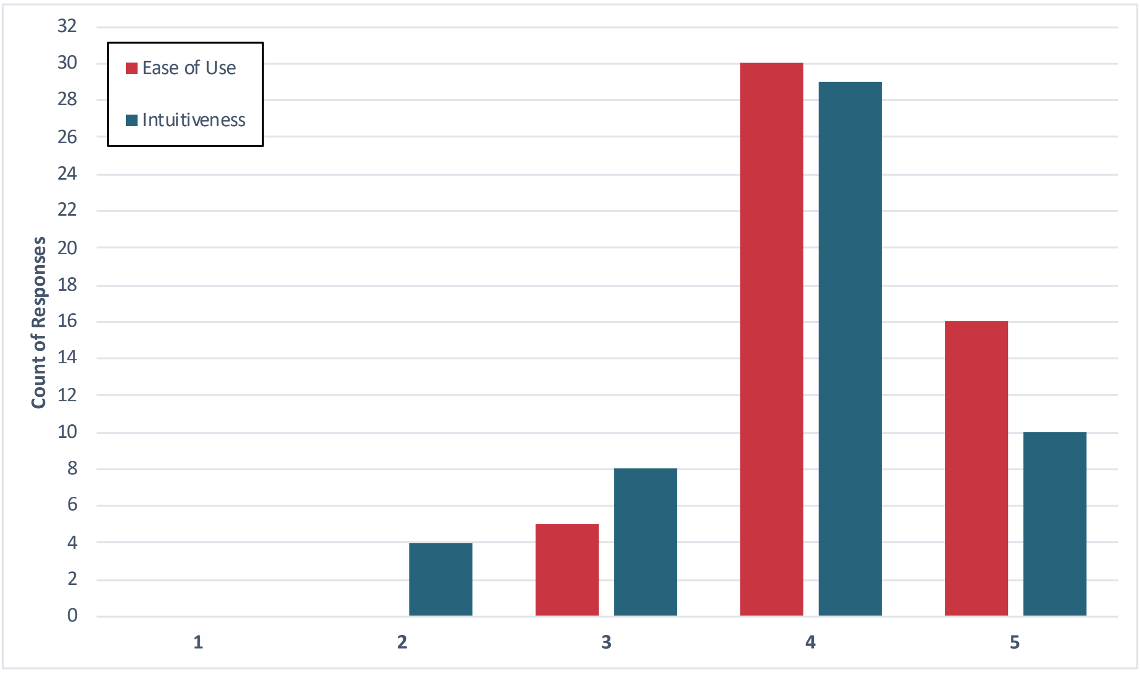


Figure 6.2: Participants' evaluation of Ease of Use and Intuitiveness of the StudyBits application from 1=*Not at all* to 5=*Very much*

Since the purpose of the StudyBits project was to give students a SSI and put them in control of their own data, the participants were asked on how much these goals were achieved by the StudyBits application in questions 5 and 6 of the survey. The participants rated the gained data control higher than the SSI support but gave both metrics a high score with gained data control having an average rating of 4.16 and SSI support of 3.88. These averages are however only weakly statistically different with $p = 0.105$, $df = 100$, $t = 1.633$. A diagram of the distribution of responses can be found in the appendix in Figure 6.3. Additionally, the familiarity with blockchain and SSI was correlated with the perceived data control and SSI support, but showed only weak to no significant connection between familiarity and perceived data control or SSI support with $r_s = 0.018$ and $r_s = 0.170$ for blockchain familiarity on data control and SSI support and with $r_s = 0.199$ and $r_s = 0.182$ for SSI familiarity on data control and SSI support respectively.

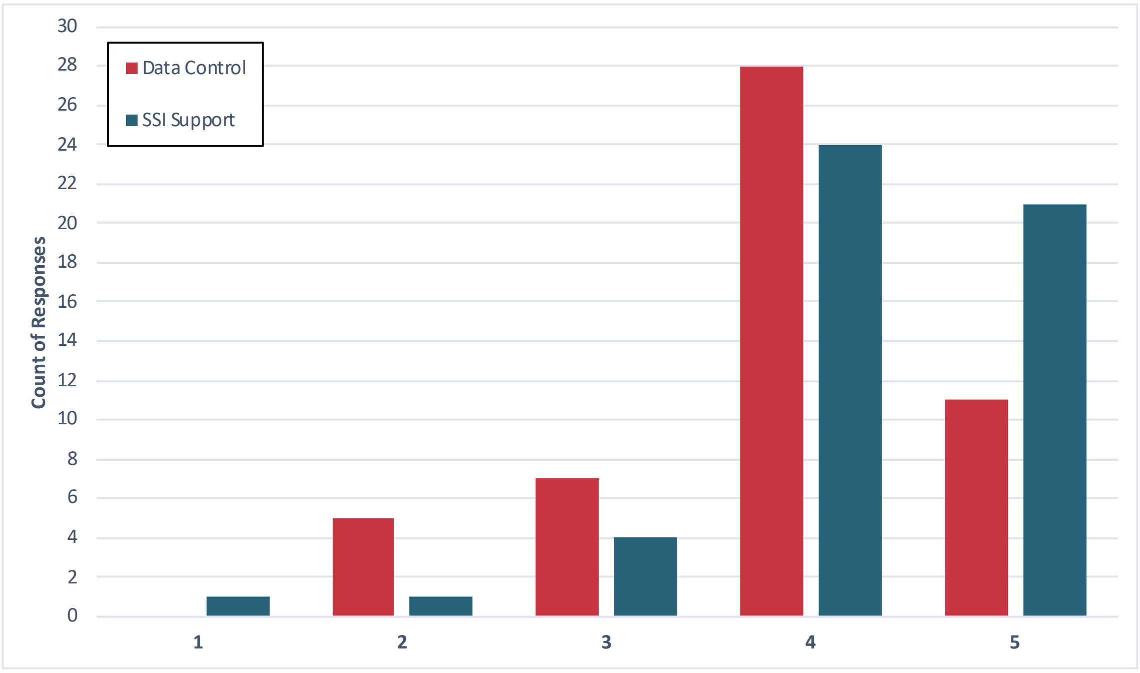


Figure 6.3: Participants' evaluation of gained Data control and SSI support by StudyBits from 1=*Not at all* to 5=*Very much*

6.2.3 Discussion

The overall evaluation of the ease of use and intuitiveness of the StudyBits frontends was quite positive, which indicates that the frontend implementation had an acceptable level of usability. Additionally, the StudyBits frontend successfully put the users into control over their personal data and succeeded in providing the users with a SSI as well. Given that the frontends were developed to demonstrate the underlying blockchain technology, the 4 out of 5 ratings were quite acceptable and no perfect score was expected given that the frontends were not yet optimized for user flow and comprehensibility.

6.3 Fulfillment of Self-sovereign Identity Principles

section 2.3 described the three main principles of SSI: **Controllability**, **Portability**, and **Security**. In the following, the extent to which these principles are fulfilled by the implementation is evaluated.

Controllability

The principle of controllability is connected to the perceived control by the users and the design and implementation decisions of the StudyBits project. As previously discussed, the users rated the gained control over their personal data as well as the extent to which the StudyBits application supports SSI as high. Furthermore, the design and implementation decisions were primarily focused on giving the user the full and only control over her personal data. This focus lead to the design decision of separating the student and university system so that students can run the application and store their personal data on their own devices.

The blockchain choice was guided by this principle as well and the Hyperledger Indy blockchain was chosen. The Indy blockchain makes it technically practically impossible for anybody but the student to share or revoke her data. However, the implementation decision of centralizing the StudyBits project to ease development diminished the level of control the user has over the creation and deletion of her account. Given these implementation decisions, the principle of controllability was fulfilled to only a medium extent. The given drawbacks could be mitigated by decentralizing the StudyBits system over multiple servers owner by independent stakeholders or by moving the StudyBits project to a permissionless blockchain, which however would entail choosing a different blockchain platform.

Portability

The principle of portability has been fulfilled both to a high or only to a low extent, depending on how the scope of a student's identity is defined. Within the StudyBits project, the student identity can be re-used with every university connected to the network. If the scope of the student's identity is defined as the StudyBits network only, then the principle of portability can be said to be fulfilled to a high extent since the identity can be re-used with every service connected to the network. However, if the scope of the student's identity is defined beyond the StudyBits system, then the principle of portability is fulfilled only to a low extent since the StudyBits identity as of now cannot be used with or ported to any other entity outside of the StudyBits network. Since the scope of this thesis was only the StudyBits project, the scope of the student's identity is defined as being only the StudyBits network. Thus, within the scope of this thesis, the implementation fulfilled the principle of portability to a high extent.

Security

The principle of security is connected to the non-functional requirement of confidentiality, which was evaluated as being fulfilled to a medium extent given the deployment decisions. Therefore, the principle of security was fulfilled to only a medium extent as well. The users can share data among each other in a confidential and secure way and once the StudyBits system is deployed to servers of multiple independent entities, the principle of security will be fulfilled to a higher extent.

Summary of SSI Principles

The fulfillment of the SSI principles has not yet achieved satisfactory levels given the deployment decisions, which were focused on easing the development process and complied with demonstration purposes. As for the non-functional requirements, the SSI principles will be fulfilled to a significantly higher extent once the StudyBits system is deployed to servers owned by multiple independent stakeholders. Once this deployment has happened, the StudyBits system should fulfill all SSI principles and therefore can be said to offer a true Self-sovereign Identity.

Chapter 7

Conclusion

The purpose of this thesis was to find an answer to the question *"How can blockchain technology enable Self-sovereign Identity?"*. This question entailed another, more basic, question, which needed to be answered first. **Can** blockchain technology enable SSI? For the specific context of the part of the StudyBits project that was considered in this thesis, the answer is a sound *yes*. In this project, blockchain technology could fulfill all functional requirements like storing, sharing, and revoking personal data. Additionally, blockchain technology seems to be able to fulfill all non-functional requirements for SSIs which are *confidentiality*, *data integrity*, and *authenticity*. Different blockchains fulfill these requirements to a varying degree, but particularly the purpose-built Sovrin/Hyperledger Indy blockchain offers satisfactory levels of fulfillment of these requirements.

Since this thesis established that blockchain can enable SSI, the main question was: **How** can blockchain technology enable SSI? This thesis answered the question by designing and implementing a proof-of-concept using the StudyBits project as a use case. The answer is multi-faceted, but can be broken down to: *By storing and transferring sensitive data off-chain and using the blockchain only for looking up non-sensitive and pseudo-anonymous data.* This approach mitigates the disadvantages of storing sensitive data on an immutable database as well as the scalability issues of current blockchain solutions. The approach also puts the user in full control over their data and fulfills all functional and non-functional SSI requirements. Therefore, within the limited scope of this thesis, the discussed approach using Hyperledger Indy can be considered a satisfactory answer to the research question.

Overall, the results of this thesis can be seen as satisfactory. The initial research yielded valuable insight, which lead to a design and implementation of the proof-of-concept that satisfied both functional and non-functional requirements to a satisfactory extent. This successful work proposed an answer to the research question that fulfilled all requirements and expectations and can be the basis for fruitful future research.

7.1 Contribution to the State of the Art

The SSI solution proposed and implemented during this thesis shows significant advantages over existing solutions like Personal Data Service (PDS) or existing Decentralized Personal Data Service (dPDS) as discussed in Chapter 2. Unlike with PDSs, the proposed solution enthrones the user as the sole owner and accessor to her personal data. Only the user can collect, share, or revoke access to her personal data. No third-party, not even the providers of this thesis's solution, are able to access, modify, or delete the user's data (in a production-ready deployment). Therefore, the proposed solutions offer all advantages of a PDS without its disadvantages like trust and dependence on the PDS.

The proposed solution also offers significant advantages over existing dPDSs like uPort or Civic since it does not store personal data on-chain. This fact makes this solution GDPR compliant, future-proof, and allows for better tracking of what data was shared with which party. Additionally, the underlying blockchain becomes more scalable as well since only little data has to be stored on-chain and transferred between and verified by all participating network nodes. By using a different blockchain than existing dPDSs, the proposed solution offers levels of scalability and confidentiality that cannot be matched by current existing SSI solutions.

Given that certain SSI solutions do exist and are operational already indicates that the proposed solution is not the only way of solving the research problem. Alternatives could be to embrace a generic blockchain and to build SSI functionality using its Smart Contract functionality like the project as uPort and Civic did. Generic blockchains offer higher levels of customizability and potentially higher censorship resistance as for example the public Ethereum blockchain does. These advantages can be significant, but their influence on the decision which blockchain to use depends on the nature of the project. In the case of StudyBits, all actors that would host the blockchain could be assumed to be trustworthy since no considerable political or financial motivations were present among the project collaborators. However, if malicious behavior is a potential threat to the network on which a SSI solution should be built, then the proposed solution might not be most suitable given its dependence on the benevolence of the network actors.

7.2 Future Work

During this research, certain questions arose regarding Sovrin that might be fruitful for future research. The Hyperledger Indy blockchain was only tested on a small scale during this thesis, therefore more research is needed to assess how the Indy blockchain scales to higher data throughput. Additionally, the security of the Idemix framework and therefore of the Indy blockchain was assumed to be appropriately high during this thesis. More research is needed to asses the actual level of security of the Hyperledger Indy blockchain. In order for StudyBits to become an adopted standard for exchange applications, the system needs to become more usable and ubiquitous. Therefore, future research could focus on how to extract the student system and blockchain connection into a mobile application with a strong focus on user experience and overall usability.

Bibliography

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," p. 9, 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[2] S. Underwood, "Blockchain beyond bitcoin," *Communications of the ACM*, vol. 59, no. 11, p. 15–17, Nov 2016. [Online]. Available: https://dl.acm.org/citation.cfm?id=2994581

[3] G. Greenspan, "Avoiding the pointless blockchain project." [Online]. Available: https://www.multichain.com/blog/2015/11/avoiding-pointless-blockchain-project/

[4] S. Psaila. [Online]. Available: https://www2.deloitte.com/mt/en/pages/audit/articles/mt-blockchain-a-game-changer-for-audit.html

[5] T. Acar, M. Forster, and B. Gockel, "Blockchain in Logistics," p. 28. [Online]. Available: https://www.logistics.dhl/content/dam/dhl/global/core/documents/pdf/glo-core-blockchain-trend-report.pdf

[6] "Coming in 2017: "Live" Blockchain Deployments Promise to Accelerate Payment Processing Services and Trade Finance." [Online]. Available: /us/content/foreign-exchange/articles/blockchain-to-accelerate-payment-processing-services/

[7] C. Akmeemana, "Using Blockchain to Solve Regulatory and Compliance Requirements," Jan. 2017. [Online]. Available: https://medium.com/@akme_c/using-blockchain-to-solve-regulatory-and-compliance-requirements-16290f4b4ac1

[8] S. Apte and N. Petrovsky, "Will blockchain technology revolutionize excipient supply chain management?" *Journal of Excipients and Food Chemicals*, p. 3, 2016. [Online]. Available: https://jefc.scholasticahq.com/article/910-will-blockchain-technology-revolutionize-excipient-supply-chain-management

[9] "Groningen to facilitate the Netherlands' first Blockchain Field Lab | News | Center for Information Technology | Society/Business | University of Groningen." [Online]. Available: https://www.rug.nl/society-business/centre-for-information-technology/news/groningen-to-facilitate-the-netherlands_-first-blockchain-field-lab?lang=en

[10] "European Commission - PRESS RELEASES - Press release - Green light for Erasmus+: More than 4 million to get EU grants for skills and employability." [Online]. Available: http://europa.eu/rapid/press-release_IP-13-1110_en.htm

[11] "725,000 Europeans went abroad with Erasmus+ in 2016 | Erasmus+." [Online]. Available: http://ec.europa.eu/programmes/erasmus-plus/news/725000-europeans-went-abroad-erasmus-2016_en

[12] B. D. Moor and P. Henderikx, "International curricula and student mobility," p. 24, 2013. [Online]. Available: https://www.leru.org/files/International-Curricula-and-Student-Mobility-Full-paper.pdf

[13] "Specific Privacy Statement - Lifelong Learning and Erasmus + Programmes – data stored and processed in Mobility Tool | Erasmus+." [Online]. Available: http://ec.europa.eu/programmes/erasmus-plus/specific-privacy-statement_en

[14] A. Grech and A. Camilleri, "Blockchain in education," *JRC Science for Policy Report*, vol. 132, no. S, p. 137, 2017.

[15] N. Szabo, "Smart contracts: Building blocks for digital markets," 1994. [Online]. Available: http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html

[16] L. Severeijns, "What is blockchain? how is it going to affect business?" p. 31. [Online]. Available: https://beta.vu.nl/nl/Images/werkstuk-severeijns_tcm235-869851.pdf

[17] "Checking the Ledger: Permissioned vs. Permissionless Blockchains," Jul. 2016. [Online]. Available: https://www.ibm.com/blogs/think/2016/07/checking-the-ledger-permissioned-vs-permissionless-blockchains/

[18] M. Scherer, "Performance and Scalability of Blockchain Networks and Smart Contracts," p. 46. [Online]. Available: https://umu.diva-portal.org/smash/get/diva2:1111497/FULLTEXT01.pdf

[19] A. Jøsang and S. Pope, "User centric identity management," p. 13. [Online]. Available: http://folk.uio.no/josang/papers/JP2005-AusCERT.pdf

[20] C. Allen, "The path to self-sovereign identity," Apr 2016. [Online]. Available: http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html

[21] E. G. Hassine and W. Ben, "Changes to facebook's "real names" policy still don't fix the problem," Dec 2015. [Online]. Available: https://www.eff.org/deeplinks/2015/12/changes-facebooks-real-names-policy-still-dont-fix-problem

[22] A. Tobin and D. Reed, "The inevitable rise of self-sovereign identity," p. 24, 2017. [Online]. Available: https://sovrin.org/wp-content/uploads/2017/06/The-Inevitable-Rise-of-Self-Sovereign-Identity.pdf

[23] D. Baars, "Towards self-sovereign identity using blockchain technology," Ph.D. dissertation, University of Twente. [Online]. Available: http://essay.utwente.nl/71274/1/Baars_MA_BMS.pdf

[24] U. Der, S. Jähnichen, and J. Sürmeli, "Self-sovereign identity – opportunities and challenges for the digital revolution," p. 6. [Online]. Available: https://arxiv.org/pdf/1712.01767.pdf

[25] "Qiy schema rulebook." [Online]. Available: https://www.qiyfoundation.org/qiy-scheme/qiy-scheme-rulebook/

[26] "Digi.me privacy notice." [Online]. Available: https://digi.me/notice

[27] P. Dunphy and F. A. P. Petitcolas, "A first look at identity management schemes on the blockchain," *arXiv:1801.03294 [cs]*, Jan 2018, arXiv: 1801.03294. [Online]. Available: http://arxiv.org/abs/1801.03294

[28] G. Zyskind, O. Nathan, and A. Pentland, "Decentralizing privacy: Using blockchain to protect personal data." IEEE, May 2015, p. 180–184. [Online]. Available: http://ieeexplore.ieee.org/document/7163223/

[29] D. C. Lundkvist, R. Heck, J. Torstensson, Z. Mitton, and M. Sena, "uport: A platform for self-sovereign identity," p. 17. [Online]. Available: http://blockchainlab.com/pdf/uPort_whitepaper_DRAFT20161020.pdf

[30] C. Technologies, "Civic - whitepaper," 2017. [Online]. Available: https://tokensale.civic.com/CivicTokenSaleWhitePaper.pdf

[31] H. Lovells, "A guide to blockchain and data protection," p. 24, Sep 2017.

[32] "Gdpr - fines and penalities." [Online]. Available: https://www.gdpreu.org/compliance/fines-and-penalties/

[33] A. Tobin, "Sovrin: What goes on the ledger?" p. 10, 2017. [Online]. Available: https://www.evernym.com/wp-content/uploads/2017/07/What-Goes-On-The-Ledger.pdf

[34] Jul 2015. [Online]. Available: https://securityintelligence.com/identity-security-and-privacy-for-electronic-user-authentication/

[35] J. Snoek, "Digital identity management in the context of gdpr and sovrin." [Online]. Available: https://blog.tykn.tech/digital-identity-management-in-the-context-of-gdpr-sovrin-43028247378b

[36] D. Gisolfi, "Self-sovereign identity: Why blockchain?" Jun 2018. [Online]. Available: https://www.ibm.com/blogs/blockchain/2018/06/self-sovereign-identity-why-blockchain/

[37] "Cut and try: building a dream," Feb 2016. [Online]. Available: https://blog.ethereum.org/2016/02/09/cut-and-try-building-a-dream/

[38] V. Buterin, *Ethereum - A Next-Generation Smart Contract and Decentralized Application Platform*, Jan 2014. [Online]. Available: https://github.com/ethereum/wiki

[39] V. Buterin and V. Griffith, "Casper the friendly finality gadget," *arXiv:1710.09437 [cs]*, Oct 2017, arXiv: 1710.09437. [Online]. Available: http://arxiv.org/abs/1710.09437

[40] "Etherum development roadmap," Jun 2018. [Online]. Available: https://github.com/ethereum/wiki

[41] "Documentation of solidity." [Online]. Available: http://solidity.readthedocs.io/en/v0.4.24/

[42] "Ethereum programming languages: Serpent," Jun 2018. [Online]. Available: https://github.com/ethereum/serpent

[43] *Ethereum Programming Languages: Vyper.* ethereum, Jun 2018. [Online]. Available: https://github.com/ethereum/vyper

[44] J. Wilcke, "Ethereum programming languages: Mutan," Apr 2018. [Online]. Available: https://github.com/obscuren/mutan

[45] "Documentation of low-level lisp-like language (lll)." [Online]. Available: http://lll-docs.readthedocs.io/en/latest/lll_introduction.html

[46] “Ethereum programming languages: Usage,” Jun 2018. [Online]. Available: https://github.com/ethereum/wiki

[47] V. Buterin, “Psa: I now consider serpent outdated tech; not nearly enough safety protections by current standards.” Jul 2017. [Online]. Available: https://twitter.com/VitalikButerin/status/886400133667201024

[48] “Mutan depreciation note.” [Online]. Available: https://forum.ethereum.org/discussion/922/mutan-faq

[49] “Announcing the first sha1 collision,” February 2017. [Online]. Available: https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html

[50] “Neo white paper.” [Online]. Available: http://docs.neo.org/en-us/index.html

[51] “Neo: Fees for system calls.” [Online]. Available: http://docs.neo.org/en-us/sc/systemfees.html

[52] “Neo’s consensus protocol: How delegated byzantine fault tolerance works,” Jul 2017. [Online]. Available: https://steemit.com/neo/@basiccrypto/neo-s-consensus-protocol-how-delegated-byzantine-fault-tolerance-works

[53] “Neo blockchain goes down after a single node disconnects temporarily,” Mar 2018. [Online]. Available: https://bitsonline.com/neo-blockchain-tanks/

[54] “neo-python - python node and sdk for the neo blockchain.” [Online]. Available: https://neo-python.readthedocs.io/en/latest/

[55] “Hyperledger fabric documentation.” [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-1.1/blockchain.html

[56] “Linux foundation: Hyperledger fabric.” [Online]. Available: https://www.hyperledger.org/projects/fabric

[57] “Architecture explained — hyperledger fabric documentation.” [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-1.1/arch-deep-dive.html

[58] Hyperledger, “Hyperledger welcomes project indy,” May 2017. [Online]. Available: https://www.hyperledger.org/blog/2017/05/02/hyperledger-welcomes-project-indy

[59] D. Reed, L. Chasen, C. Allen, and R. Grant, “Did (decentralized identifier) data model and generic syntax 1.0 implementer’s draft 01,” Nov 2016. [Online]. Available:

https://github.com/WebOfTrustInfo/rebooting-the-web-of-trust-fall2016/blob/master/final-documents/did-implementer-draft-10.pdf

[60] J. Camenisch, M. Dubovitskaya, A. Lehmann, G. Neven, C. Paquin, and F.-S. Preiss, “Concepts and languages for privacy-preserving attribute-based authentication,” in *Policies and Research in Identity Management*, ser. IFIP Advances in Information and Communication Technology. Springer, Berlin, Heidelberg, Apr 2013, p. 34–52. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-37282-7_4

[61] P.-L. Aublin, S. B. Mokhtar, and V. Quema, “Rbft: Redundant byzantine fault tolerance.” IEEE, Jul 2013, p. 297–306. [Online]. Available: http://ieeexplore.ieee.org/document/6681599/

[62] “Hyperledger indy - partner directory.” [Online]. Available: https://www.hyperledger.org/resources/vendor-directory

[63] “ISO/IEC 25010:2011 - Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models.” [Online]. Available: https://www.iso.org/standard/35733.html

[64] P. Kruchten, “Architectural blueprints—the “4+1” view model of software architecture,” *IEEE Software*, vol. 12, no. 6, p. 15, Nov 1995. [Online]. Available: https://pdfs.semanticscholar.org/2cea/7b593a045031f866d768dd7efe31bdd5c35a.pdf

[65] “Contribute to indy-sdk development by creating an account on GitHub,” May 2018, original-date: 2017-03-21T08:28:44Z. [Online]. Available: https://github.com/hyperledger/indy-sdk

Appendix A

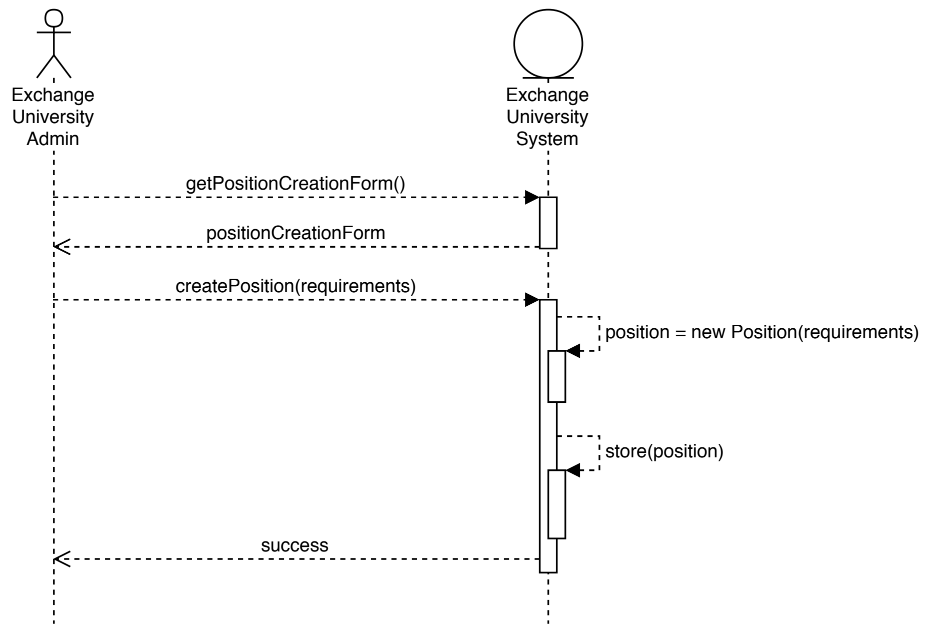

Exchange University Admin
Exchange University System
getPositionCreationForm()
positionCreationForm
createPosition(requirements)
position = new Position(requirements)
store(position)
success

Figure A.1: System Diagram for creating an Exchange Position

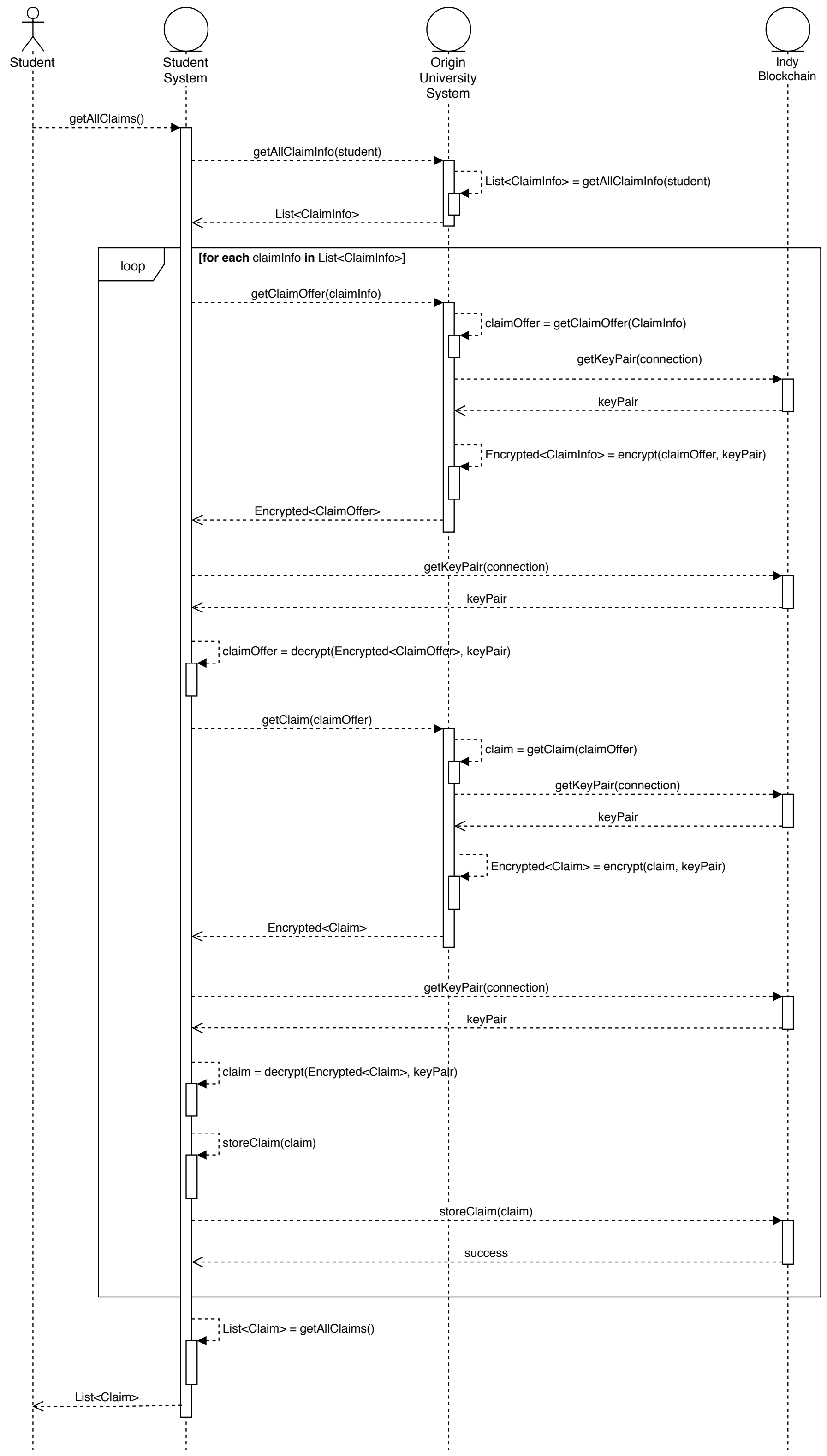


Figure A.2: System Diagram for retrieving Claims from Origin University

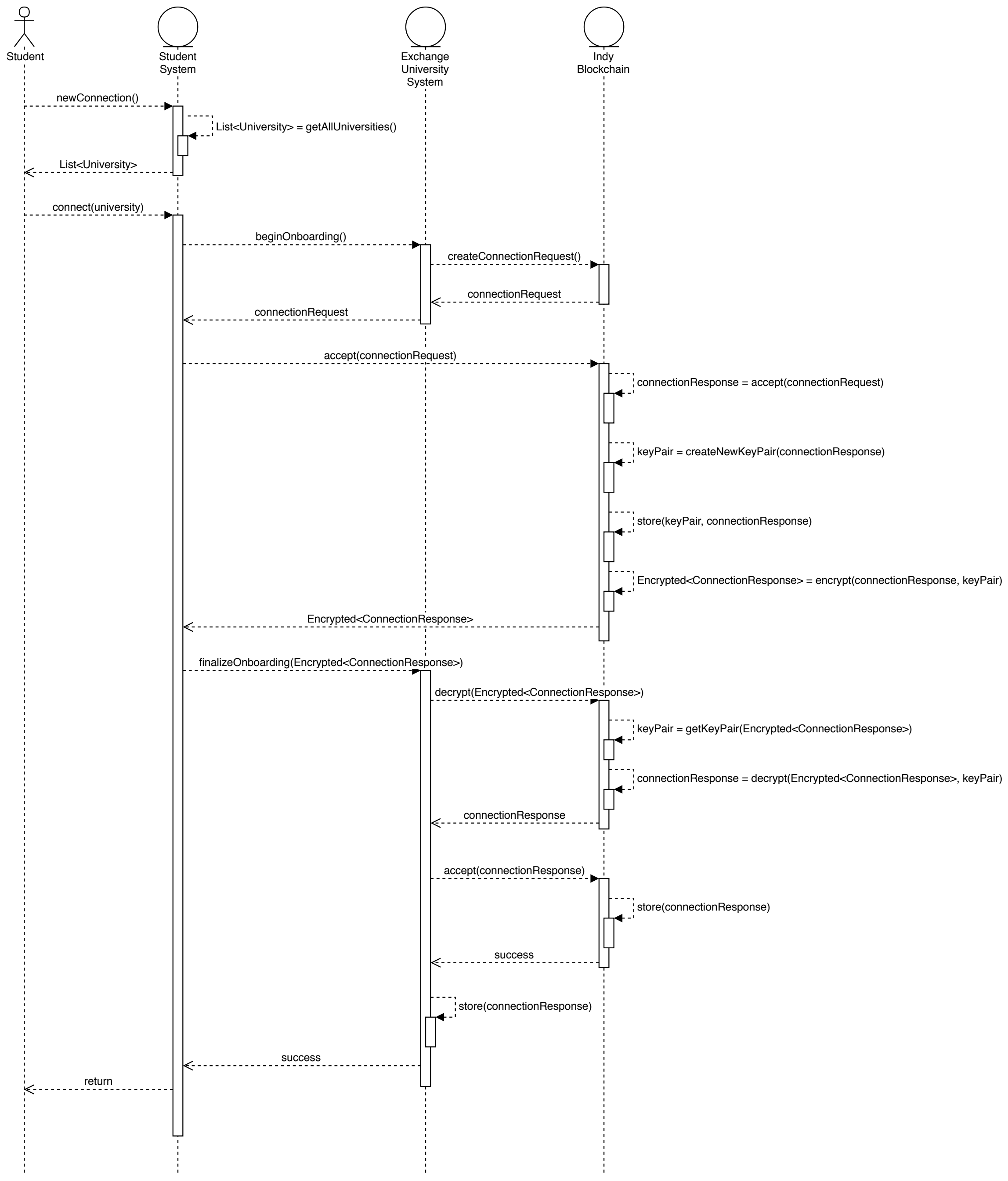

Student
Student System
Exchange University System
Indy Blockchain
newConnection()
List<University> = getAllUniversities()
List<University>
connect(university)
beginOnboarding()
createConnectionRequest()
connectionRequest
connectionRequest
accept(connectionRequest)
connectionResponse = accept(connectionRequest)
keyPair = createNewKeyPair(connectionResponse)
store(keyPair, connectionResponse)
Encrypted<ConnectionResponse> = encrypt(connectionResponse, keyPair)
Encrypted<ConnectionResponse>
finalizeOnboarding(Encrypted<ConnectionResponse>)
decrypt(Encrypted<ConnectionResponse>)
keyPair = getKeyPair(Encrypted<ConnectionResponse>)
connectionResponse = decrypt(Encrypted<ConnectionResponse>, keyPair)
connectionResponse
accept(connectionResponse)
store(connectionResponse)
success
store(connectionResponse)
success
return

Figure A.3: System Diagram for connecting with an Exchange University

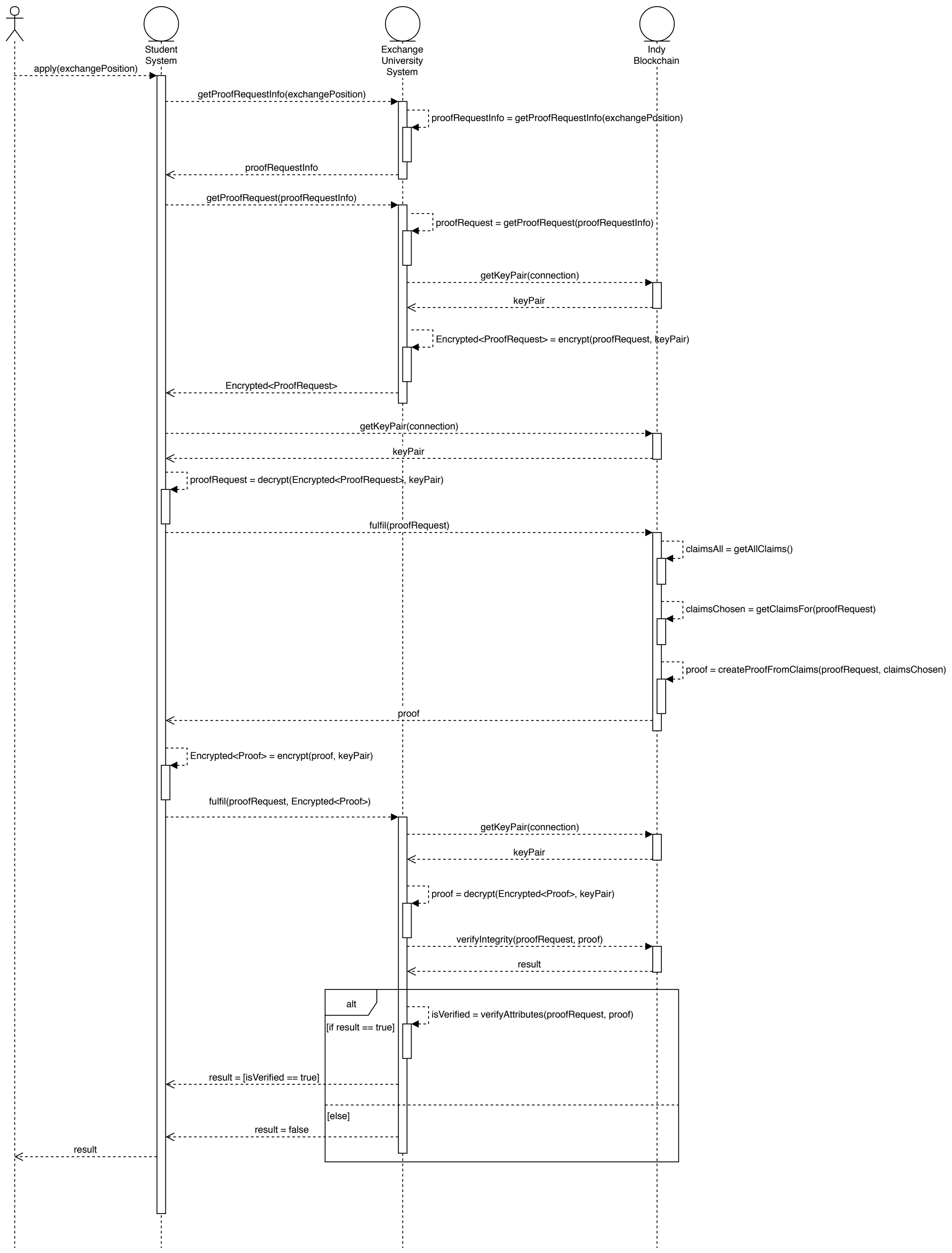

Student System
Exchange University System
Indy Blockchain
apply(exchangePosition)
getProofRequestInfo(exchangePosition)
proofRequestInfo = getProofRequestInfo(exchangePosition)
proofRequestInfo
getProofRequest(proofRequestInfo)
proofRequest = getProofRequest(proofRequestInfo)
getKeyPair(connection)
keyPair
Encrypted<ProofRequest> = encrypt(proofRequest, keyPair)
Encrypted<ProofRequest>
getKeyPair(connection)
keyPair
proofRequest = decrypt(Encrypted<ProofRequest>, keyPair)
fulfil(proofRequest)
claimsAll = getAllClaims()
claimsChosen = getClaimsFor(proofRequest)
proof = createProofFromClaims(proofRequest, claimsChosen)
proof
Encrypted<Proof> = encrypt(proof, keyPair)
fulfil(proofRequest, Encrypted<Proof>)
getKeyPair(connection)
keyPair
proof = decrypt(Encrypted<Proof>, keyPair)
verifyIntegrity(proofRequest, proof)
result
alt
[if result == true]
isVerified = verifyAttributes(proofRequest, proof)
result = [isVerified == true]
[else]
result = false
result

Figure A.4: System Diagram for applying for an Exchange Position

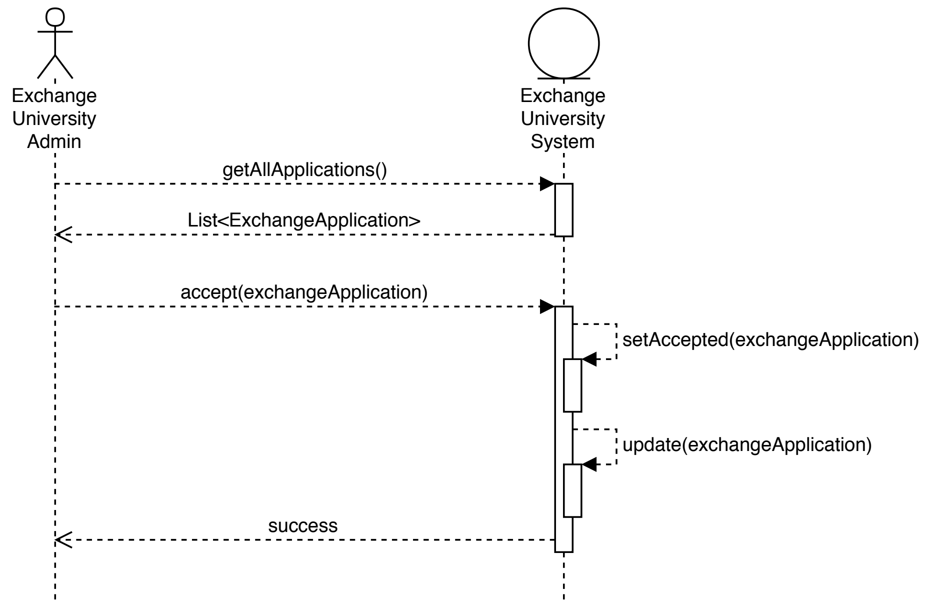
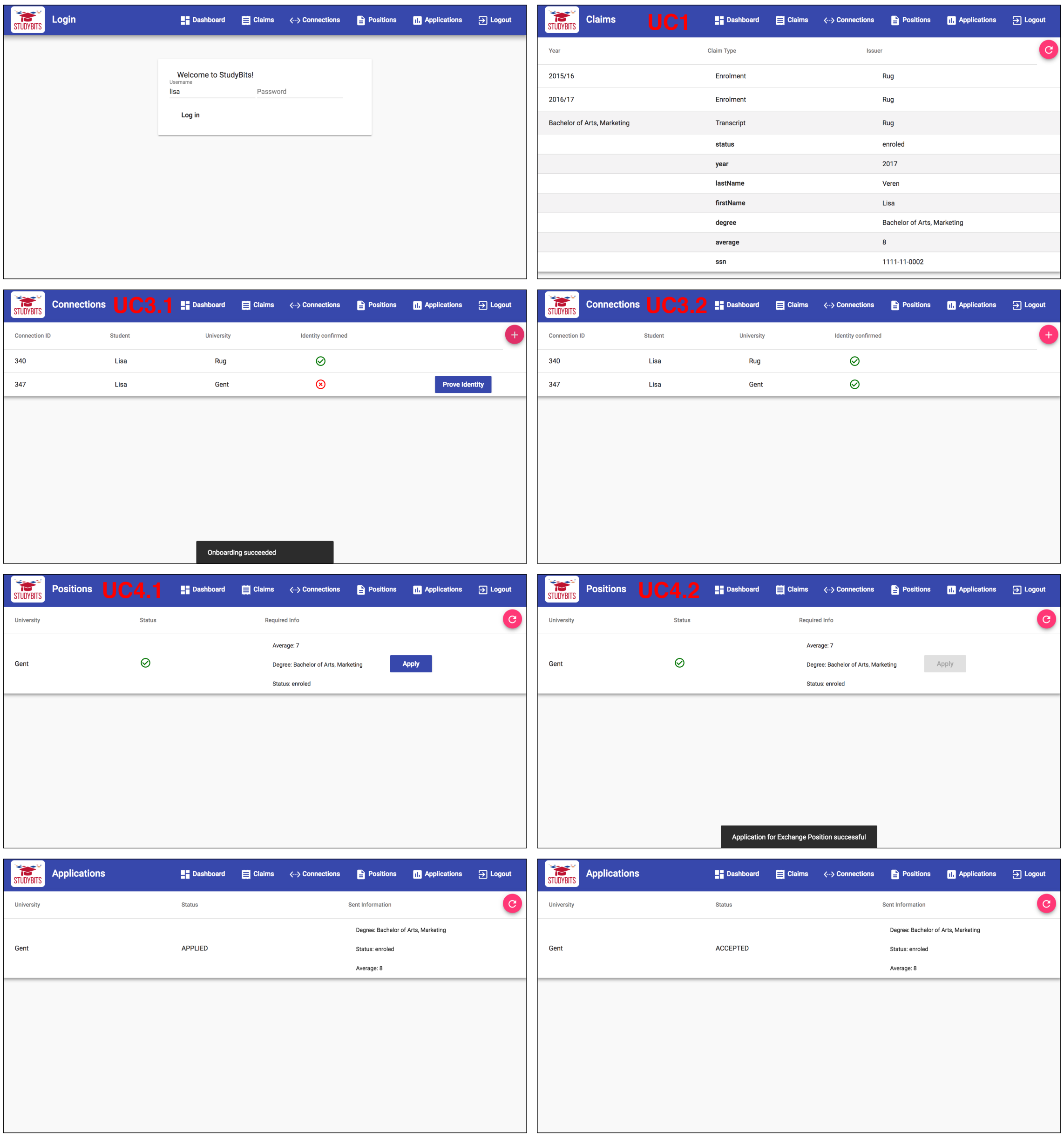

Exchange University Admin
Exchange University System
getAllApplications()
List<ExchangeApplication>
accept(exchangeApplication)
setAccepted(exchangeApplication)
update(exchangeApplication)
success

Figure A.5: System Diagram for accepting an Exchange Application



Welcome to StudyBits!
Username: lisa
Password

Year	Claim Type	Issuer
2015/16	Enrolment	Rug
2016/17	Enrolment	Rug
Bachelor of Arts, Marketing	Transcript	Rug
	status	enroled
	year	2017
	lastName	Veren
	firstName	Lisa
	degree	Bachelor of Arts, Marketing
	average	8
	ssn	1111-11-0002

Connection ID	Student	University	Identity confirmed	
340	Lisa	Rug	✓	
347	Lisa	Gent	✗	Prove Identity

Onboarding succeeded

Connection ID	Student	University	Identity confirmed
340	Lisa	Rug	✓
347	Lisa	Gent	✓

University	Status	Required Info	
Gent	✓	Average: 7 Degree: Bachelor of Arts, Marketing Status: enroled	Apply

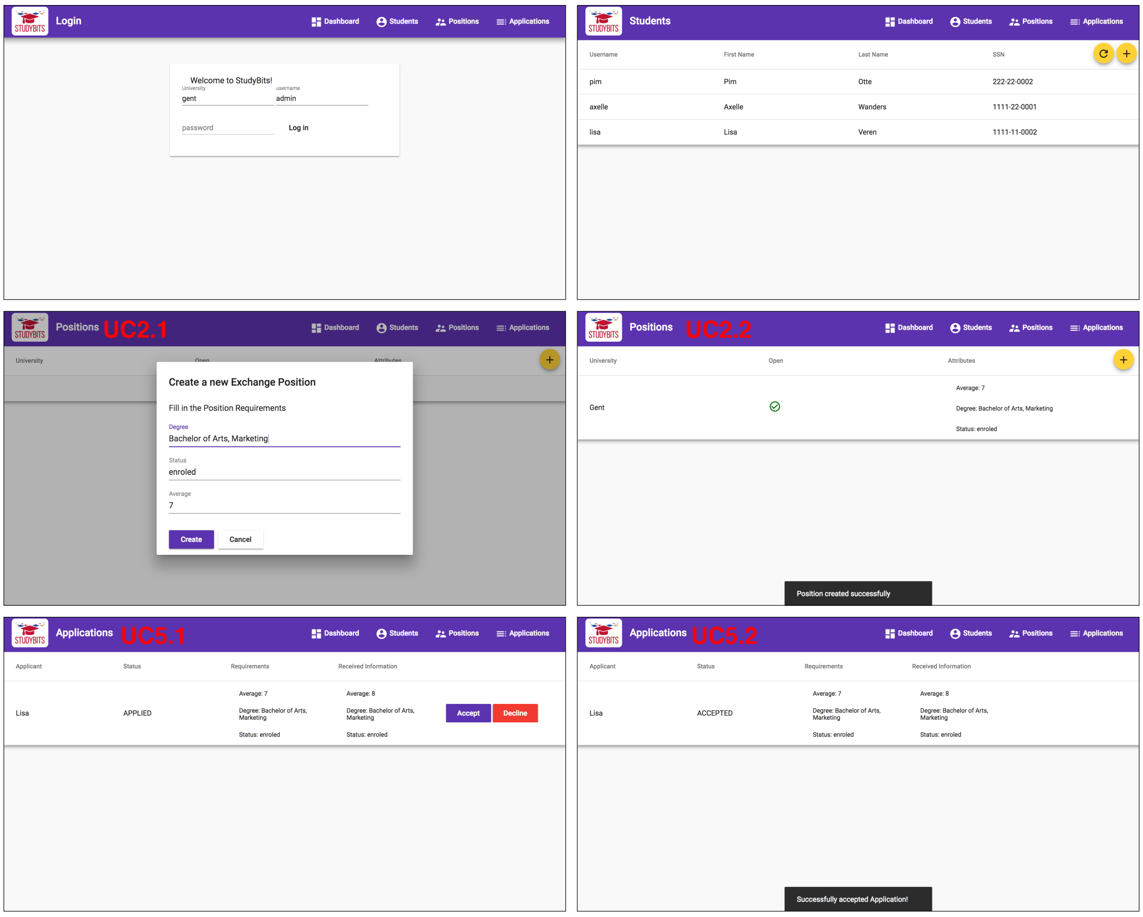
University	Status	Required Info	
Gent	✓	Average: 7 Degree: Bachelor of Arts, Marketing Status: enroled	Apply

Application for Exchange Position successful

University	Status	Sent Information
Gent	APPLIED	Degree: Bachelor of Arts, Marketing Status: enroled Average: 8

University	Status	Sent Information
Gent	ACCEPTED	Degree: Bachelor of Arts, Marketing Status: enroled Average: 8

Figure A.6: User Flow for Student with Use Case annotations from *top left* to *bottom right*



Username	First Name	Last Name	SSN
pim	Pim	Otte	222-22-0002
axelle	Axelle	Wanders	1111-22-0001
lisa	Lisa	Veren	1111-11-0002

University	Open	Attributes
Gent	✓	Average: 7 Degree: Bachelor of Arts, Marketing Status: enroled

Applicant	Status	Requirements	Received Information
Lisa	APPLIED	Average: 7 Degree: Bachelor of Arts, Marketing Status: enroled	Average: 8 Degree: Bachelor of Arts, Marketing Status: enroled

Applicant	Status	Requirements	Received Information
Lisa	ACCEPTED	Average: 7 Degree: Bachelor of Arts, Marketing Status: enroled	Average: 8 Degree: Bachelor of Arts, Marketing Status: enroled

Figure A.7: User Flow for Exchange University admin with Use Case annotations from *top left* to *bottom right*

Position	Text
1.	How familiar are you with Blockchain technology?
2.	How familiar are you with the concept of Self-Sovereign Identity?
3.	How easy was it to use the StudyBits application?
4.	How intuitive was the StudyBits application?
5.	How much did the StudyBits application put the user in control of his or her personal data?
6.	To what extent did the StudyBits application support the user towards Self-sovereign Identity?
7.	How much would you like to use the StudyBits application for applying for an Exchange Position?
8.	How useful do you think is Blockchain technology for enabling Self-sovereign Identity?

Table A.1: User Experience Survey. Possible answers were on a scale from 1=Not at all to 5=Very much.