University of Zurich[UZH]

MASTER THESIS – Communication Systems Group, Prof. Dr. Burkhard Stiller

# Identity Management for a Blockchain-based Certificate Issuance

*Vasileios Koukoutsas*
*Zurich, Switzerland*
*Student ID: 16-718-991*

Supervisor: Christian Killer, Eder John Scheid
Date of Submission: March 2nd, 2020

University of Zurich
Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

ifi

Master Thesis
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland
URL: http://www.csg.uzh.ch/

# Abstract

Academic certificates have major relevance in the labor market, signaling capability, and the level of education and skills of the recipient. Unfortunately, recent years have seen an increase in fraud, ranging from inflating academic grades to fake diplomas. A countermeasure use-case applicable to academic certificates is Proof-of-Existence (PoE), which effectively timestamps a certificate, thus, proving the existence of exactly this certificate, without leaking information about its content (the certificate's data). Prior work presented the initial requirements for a solution targeted at the University of Zurich [1], an essential building block of such a solution is identity and access management (IAM).
The goal of this master's thesis is to design and implement a suitable private distributed ledger (DL) solution with an integrated identity and access management module. The resulting solution is intended to be used in the Swiss Educhain service [2] to satisfy the main requirement of the various stakeholders, which is the issuance and verification of digital certificates utilizing blockchain technology. The Proof-of-Concept (PoC) implementation is evaluated against the identified requirements and the prototype's functionality. This master's thesis provides a blockchain-based identity and access management solution as an integral part of the produced Swiss Educhain PoC implementation.

# Zusammenfassung

Akademische Abschlüsse haben eine grosse Bedeutung für den Arbeitsmarkt, da sie die Fähigkeit und das Bildungsniveau des Empfängers signalisieren. Leider haben in den letzten Jahren die Betrugsfälle mit gefaelschten Diplomen zugenommen. Eine Gegenmassnahme, die auf akademische Zertifikate anwendbar ist, ist Proof-of-Existence (PoE). Mit PoE wird ein Zertifikat mit einem Zeitstempel versehen und so die Existenz genau dieses Zertifikats nachgewiesen, ohne dass Informationen über den Inhalt durchsickern. In vorhergegangenen Arbeiten wurden die ersten Anforderungen für eine Lösung an der Universität Zürich vorgestellt [1]. Ein wesentlicher Bestandteil einer solchen Lösung ist das Identitäts- und Zugriffsmanagement.
Das Ziel dieser Masterarbeit ist es, eine geeignete Lösung mit einem Distributed Ledger (DL) und integriertem Identitäts- und Zugriffsmanagementmodul zu entwerfen und zu implementieren. Die daraus resultierende Lösung soll im Swiss Educhain Projekt [2] eingesetzt werden, um die Hauptanforderung der verschiedenen Requirements zu erfüllen, dazu gehört die Ausstellung und Verifizierung von digitalen Zertifikaten unter Verwendung der Blockchaintechnologie. Die Implementierung eines Proof-of-Concepts (PoC) wird anhand der identifizierten Requirements und der Funktionalität des Prototyps evaluiert. Diese Masterarbeit bietet eine blockchainbasierte Identitäts- und Zugriffsmanagementlösung als integralen Bestandteil der resultierenden Swiss Educhain PoC Implementierung.

# Acknowledgments

I am grateful to Simon Müller for our close collaboration that resulted in the successful development of the Swiss Educhain PoC. I want to thank Christian Killer for his continuous support and guidance, as well as Eder John Scheid for his valuable feedback. Moreover, I want to thank Prof. Dr. Burkhard Stiller for the opportunity to work on this thesis and the Swiss Educhain project. Lastly, I want to thank my family for their endless help and support throughout the duration of my studies.

# Contents

# Chapter 1

# Introduction

Academic certificates have major relevance in the labor market, signaling capability, and the level of education and skills of the recipient. Unfortunately, recent years have seen an increase in fraud, ranging from inflating academic grades to fake diplomas. Even several organizations focus on providing illegitimate academic degrees and diplomas (also called diploma mills). Estimating globally the number of individuals with fake diplomas is a hard task. In 2015, estimations indicated that about 41% of job applicants presented falsified information about their education in the US (United States) [3]. In 2017, it is estimated that about 500 fake doctoral diplomas are sold monthly in the US [4]. Thus, the release and verification of academic certificates is a known problem, tackled by academia [2], [1], [5], [6], and also private companies.

Public blockchains can be considered tamper-proof, transparent, without any centralized control, and they offer applications to a wide range of domains [1]. The main use-case applied to academic certificates is the Proof-of-Existence (PoE), e.g., by first generating a unique cryptographic hash digest of a certificate and then publishing that hash to a public blockchain, effectively timestamping the certificate, thus, proving the existence of exactly this certificate, without leaking information about its content, typically the certificate's data. Recognizing the potential benefits of such a blockchain-based approach, prior work presented the initial requirements for a solution targeted at the University of Zurich (UZH) [1], an integral part of such a solution is identity and access management (IAM).

## 1.1 Motivation

Providing a trustworthy, decentralized, and publicly available data storage solution, public blockchains have become a disruptive technology that has seen interest across academia and industries alike. Many interesting projects (blockchain-based or otherwise) have explored the possibility to digitally verify diplomas to counteract the trend of fake degrees.

Blockcerts [5] is an initiative by the Massachusetts Institute of Technology (MIT) to create an open standard for issuing and verifying credentials on the Bitcoin blockchain. The system is now in use at MIT [5] and empowers graduates to use the service through a

mobile app [7]. Similar to that approach, the National Research and Education Network of Greece (GRNET) [8] also persists diploma hashes to a public blockchain. However, the GRNET project [16] differs from Blockcerts because not only hashes of diplomas can be stored, but also the entire verification process. Therefore, verification requests, successful or unsuccessful proof and the forwarding of the result to its requester are steps that will be stored. Another mentionable initiative is led by the Trust::Data Consortium [9] from MIT, aiming to provide safe distributed computation, enabling privacy-preserving data sharing [9]. Further, University of Nicosia (UNIC) [6] initiated a blockchain-based project to issue and verify academic certificates. UNIC aims to digitize and decentralize their internal processes issuing their first academic certificates as a Proof-of-Concept (PoC).

Generally, the same approach can be found in almost all related and blockchain-based work of academic certification. Most projects only persist the hash of the certificate into the public blockchain, while the certificate data is then sent to the recipient, who can share it with others, such as an employer. These credentials can be used to create the same fingerprint that can be found in the blockchain and, thus, verify its veracity. The amount of related work tackling the problem of academic certification highlights its necessity.

The main goal of this master's thesis is to design and implement a suitable private DL (Distributed Ledger) solution with an integrated identity and access management module. The resulting solution is intended to be used in the Swiss Educhain service [2] to satisfy the main requirement of the various stakeholders identified in Section 4.1, which is the issuance and verification of digital certificates utilizing blockchain technology. This master's thesis provides a blockchain-based identity management for the individual user roles described in Section 4.4.5.

The **key goals** for this thesis are:

**Requirements Engineering:**
Elicit requirements by evaluating the current process of certificate issuance. Based on this process, propose improvements and a system that could be used by multiple educational institutions, recipients and verifiers with the new Swiss Educhain design.

**Investigate the suitability of multiple Blockchain platforms for blockchain-based identity management:**
The comparison and evaluation of different identity management and DLs [10], [11] should be documented and support the decision for a specific platform and architecture.

**Research the individual requirements with regards to Privacy, Security and Verifiability:**
With regard to the proposed Swiss Educhain IAM requirements and the publication of hashes, evaluate from a privacy, security and verifiability perspective, evaluate potential risks and problems, but also advantages.

**Design and Architecture:**
Design of identity management and application accounts with a good user experience (UX) in mind. Create an architecture fulfilling previously determined properties (e.g. identities owned by the Recipient).

**Proof-of-Concept (PoC) Evaluation:**
Evaluate the implemented approach considering its prior defined properties and desired functionality.

**Code Delivery and Testing:**
Source code needs to be well-documented, open-source and readable. The PoC is to be tested with appropriate methods.

**Documentation and Report:**
The steps of the initial analysis, its results, design decisions, prototyping, and the evaluation approach as well as its outcome are documented in this thesis report.

Furthermore, where possible the system design and implementation should try to follow an approach that allows re-usability, easy-of-use and globally available technologies.

## 1.2 Description of Work

The focus of this master thesis is the design and implementation of a suitable identity and access management (IAM) in a blockchain-based certificate issuance process [2]. This work is done in close cooperation with Simon Müller's master's thesis [12], that is focusing on the aspect of the verification process in the Swiss Educhain process.

In the first stage, research is conducted on the relevant related work on identity and access management and research work within academia. The already elicited requirements such as the ones described in Table 4.1 are evaluated from a technical perspective and complemented by new requirements identified belonging to the scope of the Swiss Educhain project. Moreover, the first stage includes the evaluation and possibilities to integrate existing legacy systems and identities with such a new Swiss Educhain service [2].

The second stage concerns the design and implementation of an application that can be used by the `Issuer` (e.g. an UZH employee) to generate academic certificates and publish them on a private and public blockchain. The architecture was discussed during periodical meetings with the advisor to examine the feasibility of the proposal; additionally a close contact was maintained continuously with the related master thesis on "Design and Implementation of a Data-Agnostic Structure for Blockchain Proof-of-Existence". The outcome is a working PoC that adheres to the designed solution with a detailed description and reasoning of any implementation decisions taken.

The final stage of this master thesis covers an evaluation with respect to its achieved properties and a discussion of the implemented PoC. The results are contrasted to the thesis goals. This report includes the motivation and problem description, background information, related work, design decisions, implementation details, evaluation, and conclusion.

## 1.3 Thesis Outline

The thesis report is structured as follows:

**Chapter 2** shortly visits fundamental blockchain concepts and explains in detail background identity concepts.

**Chapter 3** details related work, such as, the SWITCH identity federation, key concepts of the Corda distributed ledger, and the Corda Accounts library.

**Chapter 4** presents the design of the Swiss Educhain service, enumerating different options and the chosen solution. It also includes the requirements, stakeholders and roles identified, as well as the reasoning behind all the decisions made.

**Chapter 5** presents the implementation details for the identity and access management (IAM) of the Swiss Educhain service.

**Chapter 6** evaluates the individual requirements against the implemented solution and provides a high-level evaluation of the IAM solution.

**Chapter 7** concludes this work with final considerations and identifies future IAM work.

**Appendix A** provides installation and configuration guidelines for Swiss Educhain.

**Appendix B** presents a simplified directory tree structure of the Swiss Educhain code.

# Chapter 2

# Background

In this chapter, technical background of concepts needed to understand the work in this thesis is covered. Identity management concepts such as digital identity, identity access management, access control and access control models are explained.

## 2.1 Blockchain

**Blockchain** refers to the technology that was used by Satoshi Nakamoto in [13] as the underlying building block of Bitcoin. It is based on the proposed solution to the problem of time-stamping data as described in [14], which uses hashes of data and links them in a chain, later on referred to as a chain of blocks (or block chain) data structure introduced in 1991. Satoshi's paper and Bitcoin's implementation created an ever-growing ecosystem with multiple projects initially forked from Bitcoin as alternative cryptocurrencies (or altcoins). A lot of enthusiasts envisioned general-purpose blockchain platforms which could potentially be used in a variety of use cases other than cryptocurrencies, such as, tokenization of assets, e-voting, supply chain, notarization, intellectual property protection, peer-to-peer financial transactions or settlements, and digital evidence [15]. To enable the creation of diverse applications, several blockchain platforms have been developed, as either public platforms targeting end-users or private aiming to support the enterprise sector [16].

The terms public and private in relation to blockhain platforms refer to the ability of an entity to be able to participate freely in the network. Participation to a blockchain platform might refer to any of several activities, such as, accessing the blockchain network freely, running an independent node or keeping a copy of the distributed ledger, participating in the consensus mechanism and executing functionality.

A public blockchain is defined as a platform where anyone can participate and perform the activities stated above with no access control. Analogously, a private blockchain is defined as a platform where entities can only participate if they are granted access by the platform owner. The terms permissioned and permissionless, in relation to Swiss Educhain and the scope of this thesis, refer to the permission to write in the distributed ledger. Figure 2.1

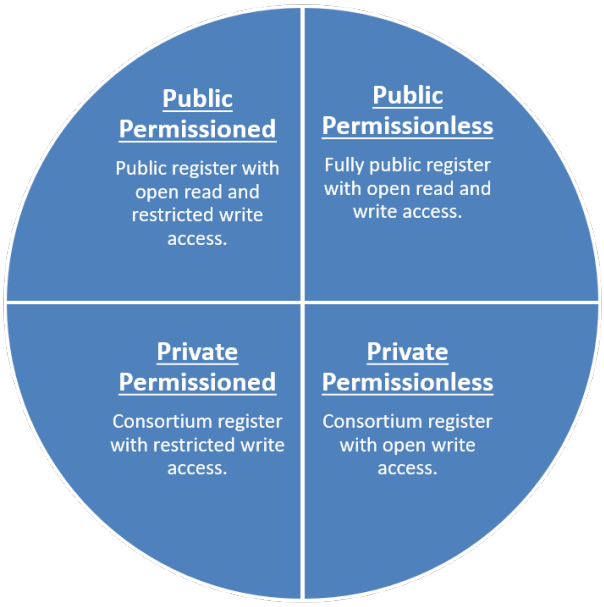


Figure 2.1: Blockchain deployment types, based on [17]

presents a high-level separation of the different blockchain platform types with respect to access control.

For the Swiss Educhain service, a hybrid approach has been chosen to combine a public permissionless blockchain with a private permissioned blockchain. The need for a private permissioned blockchain platform derives from the architectural requirements analysis in Section 4.2. For the scope of this thesis and the identity management of Swiss Educhain participants only the private permissioned blockchain is relevant.

## 2.2 Identity

Identity and access management (IAM) is a vast topic that consists of a plethora of processes, frameworks and technical implementation solutions. In this section, the foundational theoretical background of identity is explained with an emphasis only on the aspects that are relevant to this work. In Section 2.3 Identity and Access Management is explained in more detail.

### 2.2.1 Digital Identity

The notion of identity is a subject that precedes the digital era and can be defined as something different depending on the unique perspective from which it is examined [18], [19]. When identity is mentioned in this work it refers to the digital identity as seen from the prism of information technology and computer science.

An informal way to describe what is an identity would be the set of unique characteristics or attributes of an entity (human or not) which can uniquely identity this entity and

differentiate it from other ones. As stated in [20] by Grassi et al., "a single definition is widely debated internationally", therefore the following definition aims to explain the concept rather than provide a formal definition. According to [21] a **digital identity** is defined as:

> "... an online or networked identity adopted or claimed in cyberspace by an individual, organization or electronic device. These users may also project more than one digital identity through multiple communities. A digital identity is linked to one or more digital identifiers, like an email address, URL or domain name."

As it is inherently very hard to prove ownership of a digital identity and associate it with a specific entity (human or technical), especially over the Internet or any network, various processes and techniques have been developed as a potential solution to prevent identity impersonation and other attacks [20]. A brief high-level overview of such processes and frameworks is presented in Section 2.3.

### 2.2.2 Self-Sovereign Identity

A traditional identity, digital or not, is usually verified by an identity provider (in digital systems) or by an authority such as a government or a trusted third party to uniquely map an identity to an entity and prevent impersonation. The official documents or the digital accounts are issued on behalf of the entity, which is the identity owner, and handed over to them after a verification procedure is completed. An unconventional and different paradigm has emerged historically known as **self-ownership** or **individual sovereignty** through various political philosophies such as liberalism and anarchism [22]. Stemming from these, a more recent evolution of this notion is the **self-sovereign identity** (SSI).

Christopher Allen in [23] identifies four distinct phases in the evolution of identity:

1. **Centralized Identity** administrative control by a single authority or hierarchy,
2. **Federated Identity** administrative control by multiple, federated authorities,
3. **User-Centric Identity** individual or administrative control across multiple authorities without requiring a federation,
4. **Self-Sovereign Identity** individual control across any number of authorities.

According to Allen [23] a formal definition what is self-sovereign identity cannot be proposed, but he enumerates a set of principles that are meant to "... provoke a discussion about what is truly important.":

1. **Existence** - users must have an independent existence.
2. **Control** - users must control their identities.
3. **Access** - users must have access to their own data.

4. **Transparency** - systems and algorithms must be transparent.
5. **Persistence** - identities must be long-lived.
6. **Portability** - information and services about identity must be transportable.
7. **Interoperability** - identities should be as widely usable as possible.
8. **Consent** - users must agree to the use of their identity.
9. **Minimalization** - disclosure of claims must be minimized.
10. **Protection** - the rights of users must be protected.

Partial realizations of the self-sovereign identity vision have been technically possible utilizing, blockchain, decentralized systems, consensus algorithms and applied cryptography. Some popular self-sovereign identity platforms include Sovrin [24], uPort [25] and ShoCard [26]. A detailed explanation and analysis of the platforms is provided in [27].

## 2.3 Identity and Access Management

The existence of digital identities and their usage in computing systems, created the need to effectively manage digital accounts, verify their owners, provide fine-grained access to resources and allow for account lifecycle management via create, read, update and delete (CRUD) operations.

**Identity and access management (IAM)** is defined by Gartner [28] as:
"...the discipline that enables the right individuals to access the right resources at the right times for the right reasons."

In order for any IAM system to provide access to users to different resources, distinct steps need to be executed as part of the overall **access control** process. These steps can be identified in a high-level as *Identification*, *Authentication* and *Authorization*; a detailed explanation of the access control process is given in Section 2.3.1. There exist multiple ways to implement this functionality, resulting in different access control models, the more relevant ones to this thesis' work are explained in Section 2.3.2. How accounts or roles are created is not analyzed in this chapter since it is a subjective matter greatly affected by the system's design and requirements.

### 2.3.1 Access Control

**Access control** in the computer security context, can be described as the process through which users are granted access and certain privileges to systems, resources or information based on a set of credentials, assumed role or identity characteristics [29], [30], [31]. The access control process has three distinct steps:

**Identification**

In computing systems, identification can be informally described as the ability to uniquely identify a user's account and associate it with an entity, (e.g. with a unique username for digital systems or an access card for physical access), with several more formalized definitions listed in [32].

**Authentication**

Authentication "... refers to an electronic process that allows for the electronic identification of a natural or legal person. Additionally, authentication may also confirm the origin and integrity of data in electronic form, such as the issuance of a digital certificate to attest to the authenticity of a website." [33]. In general, authentication can be performed against three **authentication factors**:

**Knowledge** this includes passwords or passphrases, a personal identification number (PIN) or a response to a pre-selected security question.

**Ownership** this includes something that a user possesses, e.g. an access card, an one-time password (OTP) token or a specific phone number.

**Inherence** this could be something that a user is, e.g. biometric identifiers such as facial, fingerprint or retinal pattern recognition. [33]

Using one or more of the aforementioned factors, there are four main **types of authentication**, described below in order based on the increasing levels of security as explained in [33]:

**Single-factor** uses only one of the components, it is considered a method easily susceptible to impersonation or replay attacks and is not preferred in most modern systems.

**Two-factor (2FA)** combines two factors, e.g. something a user has and something a user knows, is considered safer and is available to the majority of computing systems.

**Multi-factor (MFA)** is similar to 2FA but combines more than two authentication factors to achieve enhanced security.

**Strong Authentication** was requested by the European Banking Authority (EBA) [34] to provide enhanced security for financial customers. It provides specific requirements such as, usage of **a minimum of two mutually independent factors** and at least one element that is **non-reusable** and **non-replicable**. A detailed opinion on the elements of strong authentication is provided in a report published by the EBA [35].

**Authorization**

Authorization "... refers to the process of granting privileges to processes and, ultimately, users." [36]. Authorization always succeeds authentication, after a user's claim of account ownership has been confirmed, the user is granted access to a set of resources based on either account information, assumed roles, membership in a group or organization, or other authorization rules [37].

### 2.3.2 Access Control Models

A variety of access control models exists to serve diverse purposes and implement unique custom access control management policies. Briefly defined are the models related to this work:

**Attribute-based Access Control (ABAC)** is defined as "An access control method where subject requests to perform operations on objects are granted or denied based on assigned attributes of the subject, assigned attributes of the object, environment conditions, and a set of policies that are specified in terms of those attributes and conditions." [38].

**Role-Based Access Control (RBAC)** is defined as "Access control based on user roles (i.e., a collection of access authorizations a user receives based on an explicit or implicit assumption of a given role). Role permissions may be inherited through a role hierarchy and typically reflect the permissions needed to perform defined functions within an organization. A given role may apply to a single individual or to several individuals." [39], [40].

# Chapter 3

# Related Work

Related work that had a direct or influencing impact on the design and implementation of this master's thesis' work is analyzed in this Chapter including topics such as, single sign-on (SSO), the Security Assertion Markup Language (SAML), the Shibboleth project, the SWITCH edu-ID identity provider, the Corda distributed ledger (DL) and the Corda accounts library [41], [42], [10], [43].

## 3.1 Swiss Educhain Previous Work

The work for Swiss Educhain is based on the preliminary work conducted by Jerinas Gresch in [44], where a detailed analysis of the diploma issuance process in UZH was performed. The high-level requirements, the stakeholders and a high-level architecture with a working proof-of-concept (PoC) was the produced outcome. Through this, the formalized proposal for the Swiss Educhain project came to fruition [2] with a more technical architecture [1]. The work for the Swiss Educhain service in this thesis, which is in close collaboration with the work in [12], considers previous work but follows a greenfield approach not constrained by any previous assumptions or decisions. The main aim is to design and create an end-to-end digital service that also encapsulates the digital issuance of a diploma, in contrast with previous work where diploma issuance is designed to occur in the legacy system. More information on the design and implementation of the Swiss Educhain service is provided in Chapters 4 and 5.

## 3.2 SWITCH

SWITCH is a Swiss foundation that provides a variety of information technology services to the academic community, mainly in the areas of network, security and identity management [45]. The identity management offerings, namely SWITCHaai (Authentication and Authorization Infrastructure) and SWITCH edu-ID, provide access to academic services in a secure manner [46]. The relationship between SWITCH, SWITCHaai and SWITCH edu-ID as part of the federation [47] is depicted in Figure 3.1.

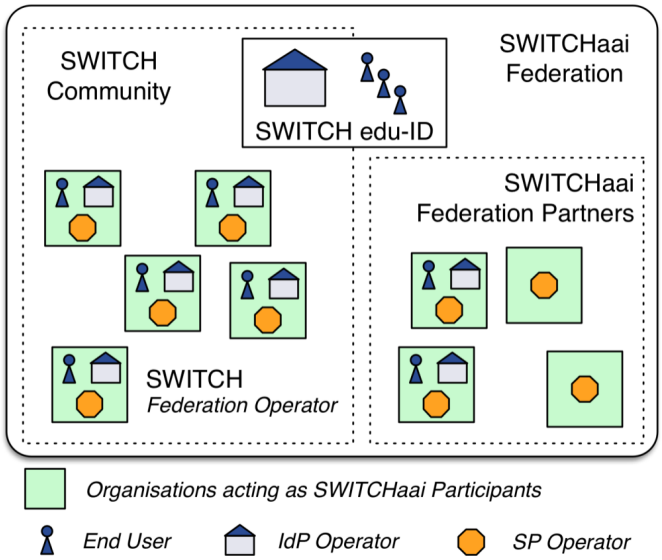


Figure 3.1: SWITCH identity federations [48].

### 3.2.1 SWITCH edu-ID

SWITCH edu-ID is the logical evolution of SWITCHaai and will eventually replace it as the single identity solution across academic organizations and services in Switzerland. SWITCH edu-ID builds upon existing infrastructure and leverages the SWITCHaai wide adoption and backwards compatibility. Several advantages for organizations, services and users make the adoption of SWITCH edu-ID attractive [49]:

**Organizations**

Organizations don't need to operate an own IdP (Identity Provider).
Covers also guests, not only regular students and collaborators.
Organizations can realize their one-identity-concept.
Organizations and their users keep the control over their data.
Organizations can use features implemented once and at one place. [50]

**Services**

High security (SWITCHaai basis, controlled guidelines and high-quality attributes).
Less administration effort.
Compatibility with SWITCHaai, Switzerland and internationally. [51]

**Users**

One identity for all academic services, lifelong, user-controlled and secure.
Simple and safe to use with transparent data quality and forwarding. [52]

From a technical point of view, SWITCH edu-ID aims to make a shift from a role-based to a **persistent identity** model, with one long-living digital identity for the user. A

user has more control over their data, and the responsibility to provide accurate data and to verify fields of the account individually, a visual representation of the user account in SWITCHaai and SWITCH edu-ID is shown in Figure 3.2. A user who is no longer member of a university and has no other affiliation(s) retains the private, user managed part of a SWITCH edu-ID identity [53].

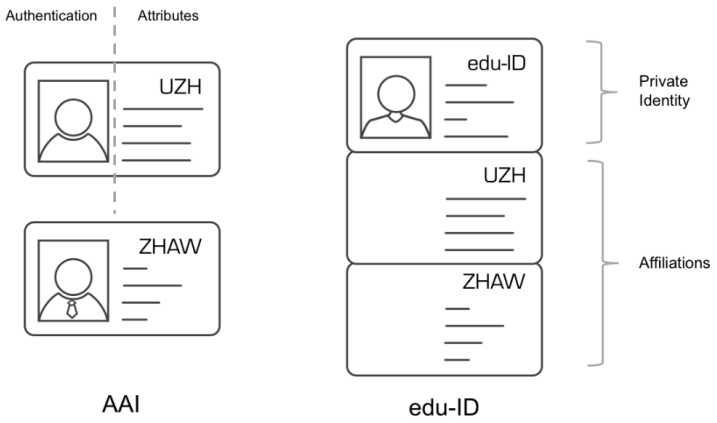


Figure 3.2: User account structure with two affiliations, compared in aai and edu-ID [53].

The authentication functionality is delegated to the SWITCH edu-ID identity provider (IdP), which manages the accounts' lifecycle, and delivers the requested subset of attributes to different services. Universities will no longer be the identity providers, but will act as **attribute authorities** (AAs) and will assign roles and access rights to users through attribute values. This enables attribute aggregation for users with multiple affiliations and the users keep a persistent account independent to duration of their affiliation with an organization. A detailed technical comparison of SWITCH edu-ID and SWITCHaai is available in [54].

SWITCH edu-ID is operated by SWITCH and is built upon multiple components that offer different pieces of functionality as shown in the architecture diagram in Figure 3.3. It is also tightly integrated with Swiss universities and academic institutions, but technically allows non-academic users and service providers to be part of the ecosystem as well. The most important notion in the SWITCH edu-ID ecosystem is the **affiliation** which determines the nature of the relationship between a user and one or more organizations. The user's edu-ID account can be created before or after an affiliation to an organization is linked and duplicate accounts are automatically detected (through the `matriculationNr` attribute which is unique per person across universities) and can be merged into one.

The Swiss Educhain is a Service Provider (SP) in the SWITCH edu-ID ecosystem and aims to provide a service initially to edu-ID users affiliated with UZH as students and later on to members of multiple organizations, details on the design and functionality are provided in Chapter 4. To access a service, a user authenticates through the central edu-ID IdP. All the user managed attributes and the linked affiliation organizational attributes are collected by the aggregator, and only the ones requested by the service are chosen. Finally, after a user has given consent, the attributes are transmitted to the service [53].

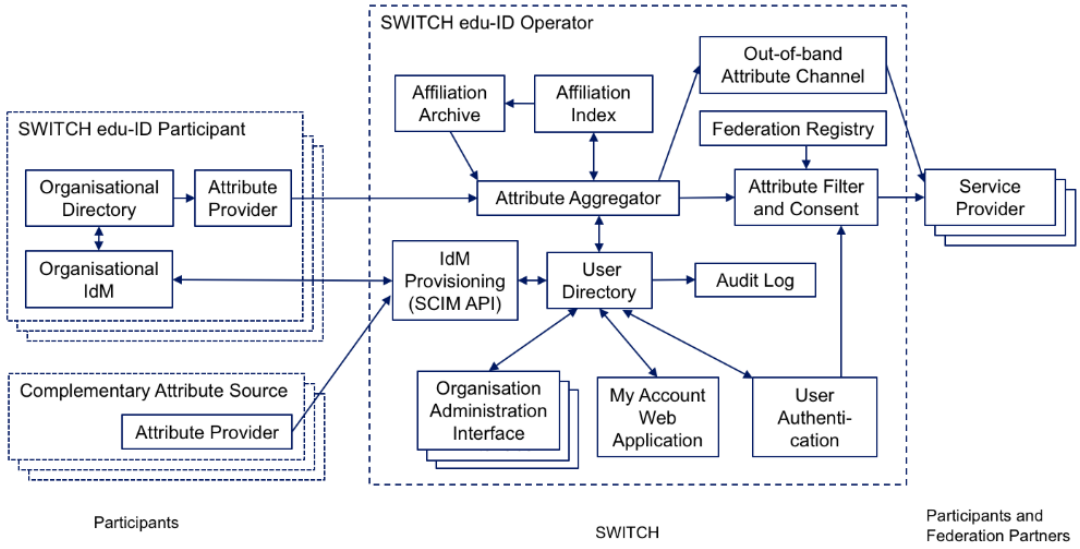


Figure 3.3: SWITCH edu-ID component architecture [53].

### 3.2.2 Shibboleth

Shibboleth is an open source software that implements the Security Assertion Markup Language (SAML) standard. It consists mainly of two major parts, an identity provider (IdP) and a service provider (SP). The most used scenario includes a third component, usually a web browser to complete a web single sign-on (SSO). A simple sequence diagram that shows the high-level steps of the web single sign-on process in Shibboleth is depicted in Figure 3.4.

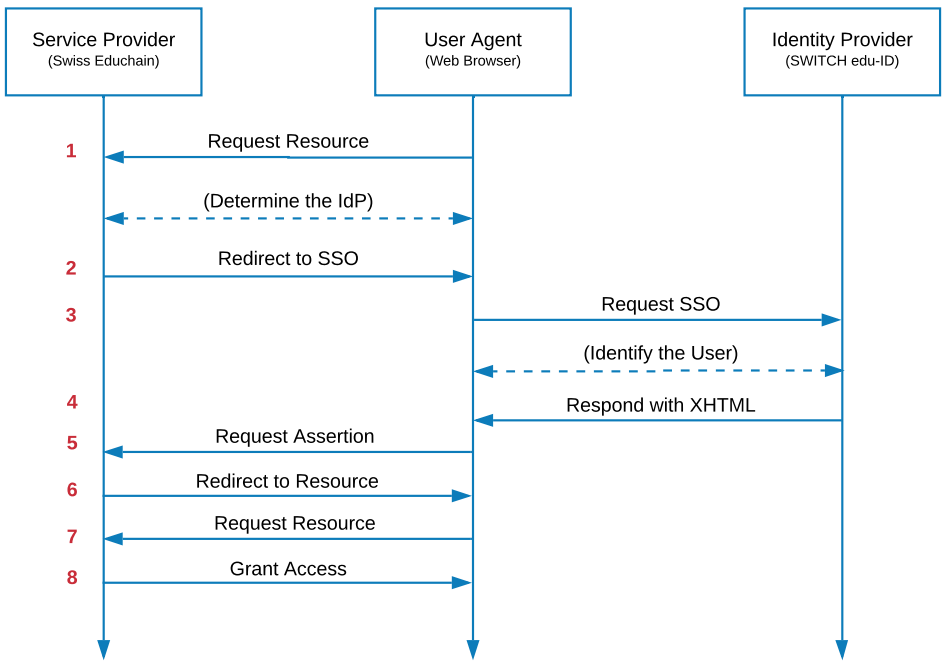


Figure 3.4: Web single sign-on sequence diagram.

Shibboleth IdPs and SPs exchange authentication, authorization and configuration information between them securely via an (usually encrypted) xml metadata file. The IdPs and SPs in the metadata file typically form a federation similar to the ones visualized in Figure 3.1. A federation is used to denote a trust relationship between the participating members. The security of messaging between IdP and SPs is handled through cryptography at various steps of the process. For example, SAML messages are usually digitally signed, and encrypted. [55]

## 3.3 Corda

Corda is a distributed ledger technology platform that can be categorized as a private permissioned blockchain (in this context the term blockchain refers to Corda's terminology and not the technical term) according to the classification described in Section 2.1. Corda is offered as open source software [56], developed and maintained mainly by R3 [57], a dedicated enterprise blockchain software firm, along with additional contributions from the community [58], [59].

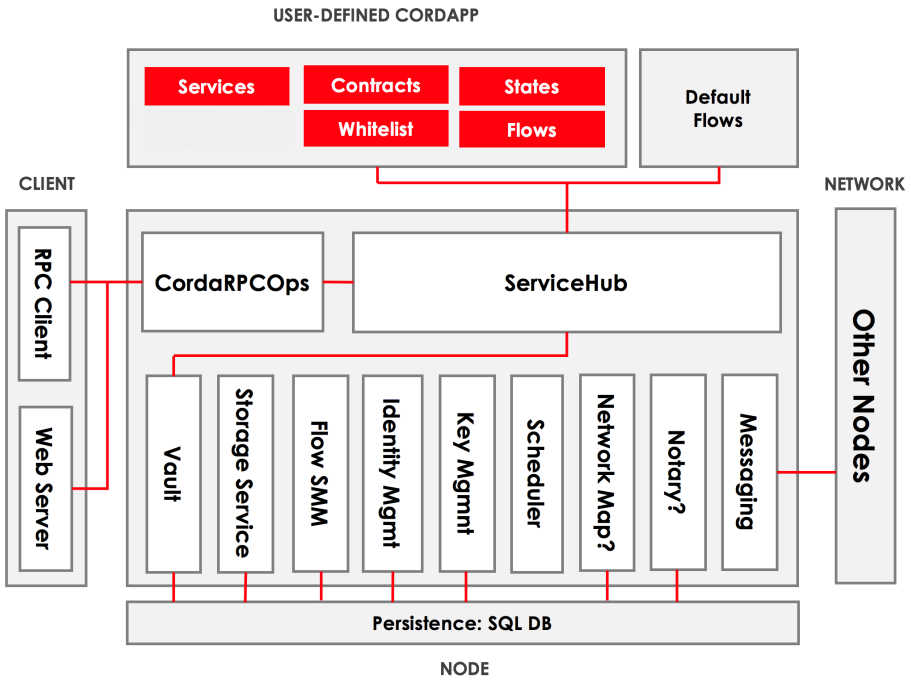


Figure 3.5: Corda node architecture [60]

The Corda key concepts as identified in [61] and which are relevant to Swiss Educhain are shortly described:

**The network** - the ecosystem that Corda exists in.

**The ledger** - the ledger, and how facts on the ledger are shared between nodes.

**States** - the states represent shared facts on the ledger.

**Transactions** - the transactions update the ledger states.

**Contracts** - the contracts govern the ways in which states can evolve over time.

**Flows** - the flows describe the interactions that must occur to achieve consensus.

**Whitelist** - the classes in the whitelist can be deserialized.

When one builds a custom Corda Decentralized Application (CorDapp), the CorDapp will have state, transaction, contract and flow classes. Nodes on the Corda network are instances that run the Corda DJVM (Deterministic JVM) [62] which can host one or more CorDapps simultaneously. Corda offers a lot of different pieces of functionality that run in the background in the form of services and are exposed to the CorDapps via the ServiceHub as depicted in Figure 3.5.

Initially, R3 targeted financial institutions and business-to-business (B2B) transactions as the main use case for the platform. This naturally influenced Corda's design and functionality resulting in what is the main differentiating factor between Corda and other public or private blockchain platforms, it being the lack of a broadcast mechanism. This technically is translated in the existence of multiple smaller ledgers and the lack of a single global ledger in the network [63]. Corda uses a point-to-point messaging system instead of a gossiping mechanism, transaction data are only shared to the participants of a transaction, and sharing is performed only on a need-to-know basis. Notary might also gain access to the the data in order to be able to validate a transaction if needed, transactions can also be validated without access to any of the participants' data, depending on the type of transaction and contract rules [64], [65].

### 3.3.1 Corda Accounts

Until recently inside the Corda network identities, were mapped to a single deployed node instance, this design came from the original perception that the Corda network would be used for business-to-business (B2B) transactions between entities belonging to a certain business network. This assumed that each entity would be able to deploy and operate an own node, something that is not the case in most applications. After high demand from the community and to be able to offer competitive functionality, R3 introduced the Corda Accounts SDK (Software Development Kit) library [66] [43].

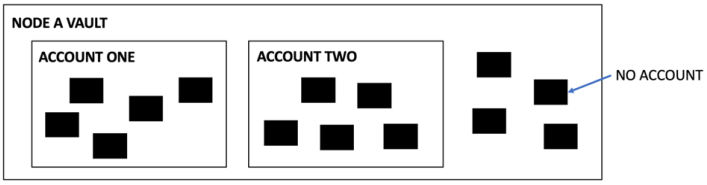


Figure 3.6: Account based node vault partition [43].

Accounts are a logical subset inside a node's vault as shown in Figure 3.6. The vault can be described as a node's encrypted storage. Unlike the nodes, the accounts do not have a

unique identifier at the network level, they inherit the node's CordaX500Name. Accounts need to be managed at the application level to make them identifiable and use them to represent different entities and transact. Figure 3.7 shows the different transaction options for accounts. Three different options exist, account-to-account transaction inside the same node, account-to-node transaction and account-to-account transaction between accounts that are hosted in different nodes.

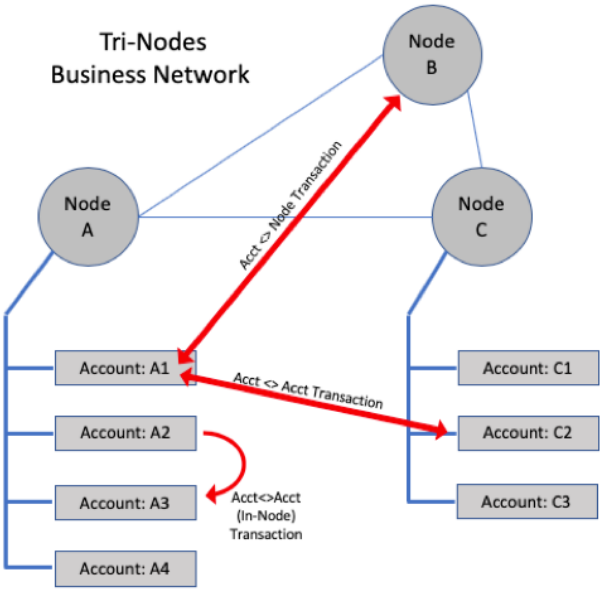


Figure 3.7: Account transaction types [43].

It is important to note that the Accounts library is not part of the main Corda release but can be optionally added as a dependency and used by a CorDapp. There is also the option to use accounts in some nodes of the network only, allowing for a hybrid setup. In the current Swiss Educhain service implementation only account-to-account transactions between accounts that reside in the same node are executed. A more detailed explanation on the design and implementation of application level accounts is provided in Chapters 4 and 5.

# Chapter 4

# System Design

The requirements elicitation and the system design of the Swiss Educhain service proved to be a challenging but quite rewarding process. Previous work conducted in [44] was taken into consideration, but a greenfield approach was chosen. The initial requirements were re-evaluated and complemented with new functional and non-functional requirements, which were refined as implementation progressed. Research, design and implementation phases, entailed valuable interactions with stakeholders (e.g. workshop with the SWITCH edu-ID program lead Christoph Graf, communication with UZH IdP (Identity Provider) administrator August Yannikis) and reaching out to the open source technical communities to comprehend how to best take advantage of the latest features, when documentation deemed to be insufficient [67], [68], [69].
In Section 4.1, the main stakeholders are identified, architectural requirements are elicited and produced, split into two distinct categories, functional and non-functional. Section 4.3 lists the different high-level design options for the Swiss Educhain service, the chosen solution, as well as a proposal for the overall governance model. Finally, Section 4.4 is entirely focused on the identity and access management (IAM) part of Swiss Educhain and enumerates the different candidate approaches complemented by their individual evaluation, a detailed presentation of the engineered solution and the reasoning behind each decision.

## 4.1 Stakeholders

*This is joint text with Simon Müller [12].*

Based on the stakeholder analysis conducted in [44], stakeholders are analyzed from a technical perspective and mapped to different roles of the Swiss Educhain system. The main stakeholders and participants identified are:

**Swiss Educhain Governance Body**
: The Swiss Educhain is governed from the UZH Blockchain Center [70] as a Free and Open Source Software (FOSS) project. The UZH Blockchain Center has the responsibility to onboard Issuing Organizations to the service.

**Issuing Organization**
Issuing Organization can be any educational institution or company that has been onboarded to the service and is given the right to issue diplomas, certificates of work history or other credentials that can be independently verified and included in a recipient's CV (Curriculum Vitae) or resume.

**Issuer**
The `Issuer` is an individual that is officially associated with an issuing organization and has been granted the right to issue digital certificates on behalf of this organization.

**Recipient**
`Recipient` is any individual that has an account in the Swiss Educhain service, receives educational or other credentials and can be associated with one or more Issuing Organizations. Each `Recipient` holds a single Swiss Educhain account that is mapped to only one real world identity.

**System Administrator**
The System Administrator is responsible for user administration, system maintenance and rollout of new system versions. The System Administrator has root access but has no right to issue credentials.

**Verifier**
Verifier is anyone that verifies a credential using the public permissionless blockchain. Verifiers are completely anonymous as they can verify any credential independently.

**Contributor**
Contributor is anyone that contributes to the project in a technical or non-technical fashion. Contributors include developers and extend to persons that add value to the project via any activity such as performing beta testing, writing documentation, reporting bugs, participating in discussions etc.

*This ends the text jointly written with Simon Müller [12].*

## 4.2 Requirements

*This is joint text with Simon Müller [12].*

In addition to the requirements identified in [44] the requirements in Sections 4.2.1 and 4.2.2 have been elicited.

### 4.2.1 Functional Requirements

Further functional requirements proposed for the Swiss Educhain project are:

Some of the functional requirements are described in more detail:

| Requirement | Description |
|---|---|
| RQ1 | Only authorized individuals are allowed to issue diplomas. |
| RQ2 | Diploma data should be confidential to its `Recipients`. |
| RQ3 | Process of issuing and verifying diplomas should abstract technical complexities. |
| RQ4 | Multiple diplomas should be processable in batch. |
| RQ5 | Verification capabilities should be accessible to anyone. |
| RQ6 | Diplomas should be verified autonomously. |
| RQ7 | Graduates should receive their diplomas in a digital format. |

Table 4.1: Initial Educhain Requirements based on [1]

| Requirement | Description |
|---|---|
| RQ8 | `Recipients` should have a unique identification. |
| RQ9 | `Recipients` should be the only ones that have the right to disclose issued credentials. |
| RQ10 | `Recipient's` account should persist over time and be independent of any association with an Issuing Organization. |
| RQ11 | Registration needs identity verification. |
| RQ12 | `Issuers` should be able to revoke diplomas. |
| RQ13 | The governance model of the Swiss Educhain system must be defined. |
| RQ14 | Issuing should create an unchangeable audit trail. |
| RQ15 | Data owner is responsible for data backup. System should provide an option for a participants' data to be exported. |
| RQ16 | Multisig transactions should be possible. |
| RQ17 | System processes data in a text-based format. |
| RQ18 | Allow for identity details to change (e.g. name, address). |
| RQ19 | The process to onboard Issuing Organisations to the Swiss Educhain service needs to be examined and defined. |
| RQ20 | User accounts need to be associated with one or more Issuing Organizations. |

Table 4.2: Swiss Educhain Functional Requirements

**RQ8**

Any `Recipient` should be uniquely identified and use a single account that is used to receive diplomas, certificates, certificate of employment etc. issued from multiple issuing organizations.

**RQ9**

Issued diplomas and other digital credentials should be entirely owned by the recipient and not the institution/company that issues them. This will allow for complete control of someone's data and enable a granular voluntary peer-to-peer read-only disclosure (temporary or permanent).

**RQ10**

The cease of operation of any Issuing Organization should not affect the `Recipient's` account, it must also be ensured that no individual other than the System Administrator has the ability to suspend or lock a `Recipient's` account.

**RQ11**

The user account creation process (or user registration) needs to be defined, and map one single real-world identity to a unique digital identity. As part of this process the need for a Know-Your-Customer (KYC) process or identity verification should be examined.

**RQ12**

`Issuers` of a digital credential must be able to revoke it if there is proof that the `Recipient` acquired it maliciously or by falsifying information.

**RQ16**

Multisig refers to the technical ability for multiple parties to sign a transaction.

**RQ17**

Any produced files such as PDFs (Portable Document Format) must be converted to a text-based format (i.e. using base64 conversion) and included in a JSON (JavaScript Object Notation) file/structure.

**RQ18**

`Recipient` should be able to request an update to his data.

**RQ20**

The association of user accounts to one or more Issuing Organizations must be defined, this includes the assignment of specific roles for members within a certain Organization such as credential `Issuer` and `Recipient`.

### 4.2.2 Non-Functional Requirements

The Swiss Educhain system is intended to provide a service to a plethora of organizations such as universities, government departments and employers of different sizes, in diverse jurisdictions and of varying technology maturity level. This creates the need for a system that fulfills these non-functional requirements:

| Requirement | Description |
|---|---|
| RQ21 | Verifier must be able to verify the diploma even when the private environment is not available. |
| RQ22 | Easy to use from a user perspective, with a simple UX/UI and straightforward functionality. |
| RQ23 | Easy to install, configure, deploy, operate, monitor and maintain from a System Administrator's perspective. |
| RQ24 | Uses technologies that are freely available, popular, well-established and mature. |
| RQ25 | Has as few as possible technology requirements and dependencies both in terms of hardware and software. |
| RQ26 | Can be easily integrated with existing IT infrastructure and is cross-platform compatible. |
| RQ27 | Is not dependent on state-of-the-art technologies such as Containers, Cloud etc. |
| RQ28 | System should support multiple issuing organizations. |
| RQ29 | High-level access control must be defined for the different kind of identities participating in the system. |
| RQ30 | Can be modularly enhanced by existing functionality. |
| RQ31 | Data that are disclosed peer-to-peer should not be broadcasted. |
| RQ32 | All transactions in the system should be signed and the identity of any action initiator should be verifiable. |

Table 4.3: Swiss Educhain Non-Functional Requirements

*This ends the text jointly written with Simon Müller [12].*

## 4.3 Architecture

This Section discusses the chosen high-level architecture for Swiss Educhain. This work was done in close collaboration with [12] using common decision criteria, the IAM specific architecture is explained in Section 4.4. Some important factors and goals taken into consideration during this design phase include:

**Requirements fulfillment**
The chosen solution should partially or completely fulfill the majority of the architectural requirements identified in Section 4.2.

**Real-world application**
The Swiss Educhain service is planned to be eventually deployed as a real-world service for members of academia, with the aspiration to serve additional use cases in the future. In contrast to other projects or proof-of-concepts that are created as part of a thesis or an academic project, this work will continue to be developed and improved upon until it is production ready.

**Privacy, Security and Verifiability**
Swiss Educhain must handle sensitive user data, thus, it is crucial that information is stored and transferred securely and disclosed strictly on a need-to-know basis only. Actions such as issuing or blacklisting diplomas must be auditable, with the identity of any action initiator easily verifiable.

**Simple development and operation**
As students are the main contributors and the service will be operated under the academic umbrella, it is essential that no specialized knowledge is required to be able to develop, maintain and operate the service. This would also mean that a strong preference to open source, well documented and widely adopted technologies should be given.

**Extensible modular architecture**
New features' development, easy integration with existing or future external components and uncomplicated co-hosting of the service with other systems should be enabled. To achieve this goal, a modular loosely coupled architecture should be designed, so components can interact using clearly defined interfaces that encapsulate and hide the chosen functionality implementation(s).

Based on the guidelines and goals mentioned above, an appropriate solution is chosen in Section 4.3.2 and the implementation details are mentioned in Chapter 5.

### 4.3.1 Candidate Solutions

After conducting initial research and taking into consideration possible technical solutions and available technologies, two major options were identified for the Swiss Educhain service which are explained in this Section.

#### Governance Model

*This is joint text with Simon Müller [12].*

The Swiss Educhain ecosystem is comprised of a variety of institutions and stakeholders as identified in 4.1. This creates challenges and different options for the governance model to be chosen.

Two different governance and operational models have been determined:

**Option 1: Global Network**
A global network is deployed, where different organizations are onboarded. A process to identify accredited institutions and to allow them to participate in the system as `Issuers` must be defined and explored both in technical and non-technical aspects. The system is operated and governed by the UZH Blockchain Center which is responsible for administration, support and new functionality.

**Advantages**

Only one user account for each `Recipient`.
Unified update and upgrade rollout.
Low administration and operational effort.

**Disadvantages**

Forced update and upgrade policy.
More complex identity management.

**Option 2: Per-institution Network**

Each institution deploys and operates an independent instance of the FOSS Swiss Educhain service. The service is entirely operated and governed by each institution which is responsible for administration, support and new functionality.

**Advantages**

More control over the system.
Independent update and upgrade rollouts.

**Disadvantages**

High administration and operational effort.
Users need new account for each institution.
Institution's cease of operation results in unexpected system termination.

After weighing advantages and disadvantages between option 1 and option 2, the Global Network governance and operational model was the preferred choice. A global network from a technical perspective simplifies the deployment and operation of the system. Assuming that the Swiss Educhain service will be adopted by multiple institutions a global standardized network will simplify the user administration and reduce significantly the overhead of operating multiple parallel instances of the system.

*This ends the text jointly written with Simon Müller [12].*

### 4.3.2 Architecture Solution

*This is joint text with Simon Müller [12].*

After the initial analysis of the previously elicited requirements the resulting Swiss Educhain high-level architecture is depicted in Figure 4.1.

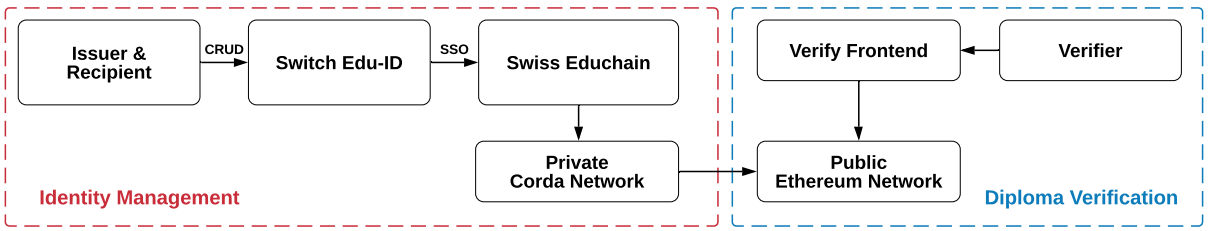


Figure 4.1: Swiss Educhain high-level architecture.

The high-level architecture as shown in 4.1 illustrates the logically separated steps and processes entailed in the end-to-end issuance and verification of a diploma.

Initially, the diploma `Issuer` and `Recipient` register to the Swiss Educhain service with their SWITCH edu-ID account which integrates through the Shibboleth web single sign-on (SSO) implementation. The Swiss Educhain service leverages Spring Boot and the Corda distributed ledger (DL) to execute core functionality, such as, executing signed transactions, storing account and diploma information in states, and permitting `Issuers` to issue diplomas to `Recipients`.
Corda was chosen as the private permissioned blockchain due to the excellent compatibility with other system components, such as Apache and the AJP connector. Futhermore, it is written in Kotlin same as the Spring Boot webserver simplifying development and it offers extensive up-to-date documentation. Then, the issued diplomas can be hashed and published, individually or in batches, to the public Ethereum ledger via a Solidity smart contract which offers additionally the option to blacklist an already published diploma. As a last step in the issuance process verification can occur independently and anonymously by any person or organization.

To achieve the minimum required functionality for a proof-of-concept implementation as described in Section 4.3.3, a plethora of components and different technologies were combined. A detailed architecture of the Swiss Educhain components is depicted below in Figure 4.2.

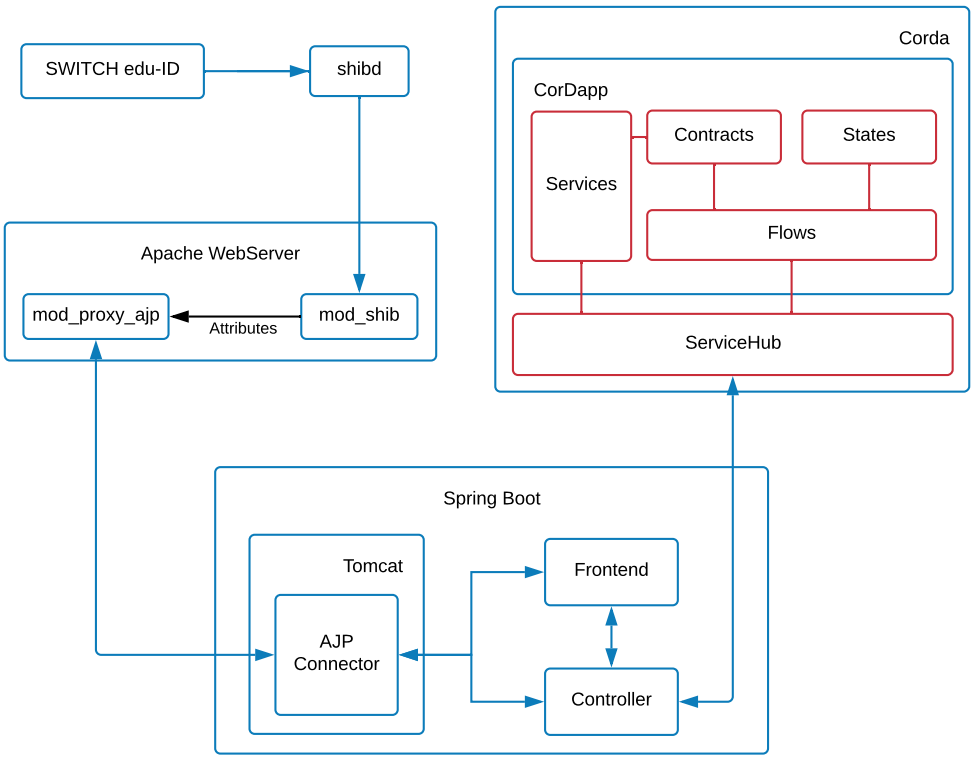


Figure 4.2: Swiss Educhain component architecture.

Figure 4.2 outlines the inner workings of the Swiss Educhain application component and depicts in detail all interactions down to the module level.

As it can be seen in Figure 4.2, the Swiss Educhain service is dependent on the account information provided by SWITCH edu-ID. SWITCH edu-ID relays attributes from a logged-in user to `shibd`, the Shibboleth daemon service, through the `mod_shib` the Apache Webserver Shibboleth module. With this configuration in place, Apache is able to disclose the user account attributes via the Apache JServ Protocol (AJP) over the `mod_proxy_ajp` module when requested by the Spring Boot application server. An embedded Tomcat server resides inside the Spring Boot application that is listening to and serving all call requests, over the custom AJP Connector, from and to the Apache Webserver. The embedded Tomcat server hosts the Frontend over which a logged-in user interacts with the Swiss Educhain service. Additionally, in the Spring Boot application relies the Controller which exposes various RESTful endpoints to the Frontend and the Tomcat server.
When the controller receives a request that requires a connection to Corda it calls the Corda ServiceHub via a Remote Procedure Call (RPC) operation. Within Corda, ServiceHub is the main orchestrating entity can be viewed as an entry point. It routes all requests to the appropriate service or flow that are part of the Corda Decentralized Application (CorDapp). Services and flows execute transactions based on the contract rules, store or update information on states and cryptographically ensure the overall system privacy, security and actions verifiability.

*This ends the text jointly written with Simon Müller [12].*

### 4.3.3 MVP Functionality

*This is joint text with Simon Müller [12].*

The following functionality has been identified as the minimum required for a PoC version of Swiss Educhain:

**Identity management**
- Two types of identities should be supported, `Issuers` and `Recipients`.
- Define process of creating a new account.
- Define data structures for the Educhain account data.
- Define access control rules for general access to the service.
- Define application level access control for `Issuers`.
- Fetch student details to create Corda identities.
- Detect student detail changes and update Educhain account automatically.

**Data Structures**
- Define an appropriate data structure for storing data related to a diploma.
- Allow digital diploma hashing and publishing on a public blockchain.
- Allow existing diplomas to be digitally signed and published.
- Publish diplomas in batch.
- Blacklist diplomas.

**Web Interface**
- Issue diploma by uploading JSON.

View received diplomas (all users).
View issued diplomas (only `Issuers`).
`Issuer` should be able to perform all actions from the frontend.
Provide a simple login and logout interface.

**Operations**
Define build, installation and deployment process.
Encryption for data in transit and data at rest.
Cross-platform compatibility.

*This ends the text jointly written with Simon Müller [12].*

## 4.4 Identity and Access Management

With Corda as the chosen private permissioned blockchain technology to be used, an end-to-end IAM system must be designed, implemented and tightly integrated to offer appropriate access controls. During the design process, several options were considered and evaluated against technical and non-technical criteria to achieve an optimal solution with little or no compromises.

As identified in Section 4.1, the Swiss Educhain service has only two distinct types of roles, an **Issuer** and a **Recipient**. The role of a user is determined through their current active scoped affiliation (student, staff etc.) with one or more organizations. It is also possible that a user is at the same time both an `Issuer` and a `Recipient` assuming the two linked affiliations are not in the same organization.

### 4.4.1 Identity Candidate Solutions

To ensure the requirements fulfillment and the Swiss Educhain service success, it is essential to choose the best way to implement an IAM solution. There are three main candidate implementation approaches:

1. Creation of a completely custom IAM solution using Corda, operating an own IdP.
2. Leveraging an existing CorDapp Identity solution, which integrates Corda with public third-party IdPs [71].
3. Integration with a federated IdP service, and mapping of IdP accounts with the Swiss Educhain CorDapp accounts.

Assessment of the different approaches' suitability for Swiss Educhain:

**Own IdP**
Creation and operation of an own IdP, is well-suited for complex IAM requirements of organizations that need to manage diverse roles, user groups and access rights.

As identified in Section 4.1 the Swiss Educhain service needs to only accommodate for two kind of accounts `Issuers` and `Recipients`. Designing, implementing and operating an own IdP, comes with a lot of overhead, such as user onboarding, KYC verification, account lifecycle management, sensitive data handling, regulatory compliance (e.g. GDPR) and generic technical maintenance activities.

**Third-party identity CorDapp**
A third-party CorDapp, offers out-of-the-box integration with one or more public IdPs. This choice caters best to an application that aims to easily acquire access to, and onboard as many users as possible, targeting a wide audience. Some considerations with this approach include technical dependencies, degree of adoption, whether the solution is Free and Open Source Software (FOSS), and what is the provided licensing or support amongst others.

**Integration with a federated IdP**
Integrating with a federated IdP solution offers a simple and straightforward way for a service provider to gain access to a special interest audience and users from multiple organizations, which participate in the federation. SWITCH edu-ID is the evolution of the sole identity provider (SWITCHaai) of the swiss academic community. A walkthrough of the detailed benefits for users, service providers and organizations using SWITCH edu-ID is given in Section 3.2.1. Possible drawbacks of using a federated IdP include a tight dependence on the quality and availability of the services provided by the IdP, lack of new feature implementation, no flexibility for customization, and the admission that provided user's data quality is accepted via a chain of trust [72].

For the Swiss Educhain service needs, integration with SWITCH edu-ID is the approach that provides most benefits with few to almost none significant drawbacks. The following advantages are particularly important for Swiss Educhain:

- One unique, long-lived and user-controlled identity for users.
- Sensitive user data are not stored on the Swiss Educhain service, data and fine-grained attributes are only disclosed on a need-to-know basis during login.
- Less administration, no need to onboard organizations or users and verify their details. Verification, access rights and data updates are performed by the organizations for affiliations and by edu-ID for personal user data.
- Swiss Educhain can be used by any user very easily through Web SSO.
- High security standards implemented and enforced from SWITCH centrally.
- Interoperability with SWITCHaai, Switzerland and internationally.

#### SWITCH edu-ID features

In addition to the traditional core IdP service functionality, SWITCH edu-ID offers a wide range of advanced features to enhance security, privacy and interoperability for all users, service providers and organizations that participate in the SWITCH community. The

most important features relevant to the current or future state Swiss Educhain service are:

**Advanced password policy**
Enforces minimum password strength, rejects compromised passwords and complies with NIST recommendations [73].

**Multi Factor Authentication**
Available in the form of SMS, or Time-based one-time passwords (TOTP) with the addition of one-time recovery codes [74].

**Attribute quality**
Individual level of assurance for each attribute with three distinct levels (low, medium, high), which are expressed in the meta-attribute `swissEduIDAssuranceLevel` [75].

**Extended Attribute Modes**
Potential to request attributes from the personal part of the identity, from linked current affiliations and group membership information [76].

**Technical Accounts**
Support for technical accounts [77].

**Link Composer**
Allows service providers to compose links for various flows such as the attribute completion flow and the login flow [78], [79], [80].

**Testing**
A test version is provided (`test.eduid.ch`) to run tests in an isolated environment, the federation is AAI Test (allows linking production SWITCHaai identities) [81].

### 4.4.2 Identity Chosen Solution

As the preferred solution, integration with the federated SWITCH edu-ID IdP is chosen. Swiss Educhain participates in the federation as a Service Provider (SP) under UZH as its Home Organization, with an appropriate service Resource Description in the SWITCH AAI Resource Registry [82].

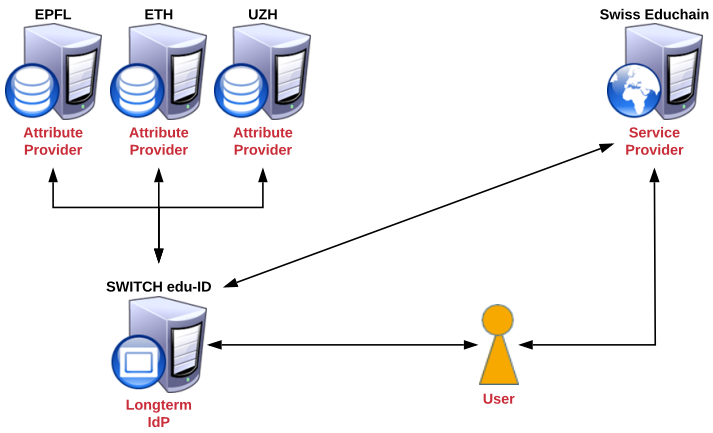


Figure 4.3: SWITCH-Swiss Educhain architecture based on [83].

In Figure 4.3 the high-level IAM architecture between Swiss Educhain, SWITCH edu-ID and the Attribute Providers is shown. The user only has a single unique long term identity, hosted and operated by SWITCH edu-ID, which can be used to create affiliation(s) with one or more Organizations. The Organizations act as attribute providers and attest that a certain individual has a specific role. Service Providers become members of the federation and register as Resources in the SWITCH AAI Resource Registry. Based on the approved Resource description and after the user's disclosure consent, only the required attributes are sent to the Service Provider by the edu-ID IdP. Attribute values are always fetched in real-time from all Attribute Providers and updated if needed before sent to the Service Provider. Figure 4.4 depicts an overview of the Resource Registry tool.

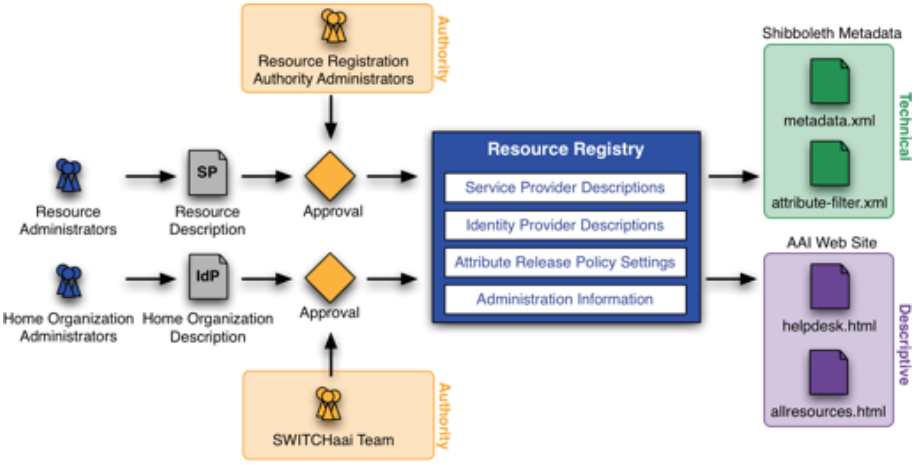


Figure 4.4: Resource Registry overview [84].

Swiss Educhain has registered as a Resource in the AAI Test federation with Home Organization the University of Zurich test IdP. For the architecture shown in Figure 4.3 to be operational a certain level of trust needs to exist between the federation participants [85]. Trust in this context refers to information or data released from one federation participant to another, and that an entity trusts another means that the data (attributes) received are accepted as correct, complete and previously verified either directly from the data source or through a chain of trust.

Figure 4.5 shows the trust relationships relative to the Swiss Educhain service. SWITCH edu-ID is the trusted root in the federation, thus trusted by everyone. There exists a two-way trust relationship between Attribute Providers and SWITCH edu-ID, a result of the gradually built trust during the onboarding of Organizations to the federation, a process that entails multiple steps and has a duration of several months [86]. The identity data transferred are logically structured in the form of attributes.

Attributes are the main building block of SWITCH edu-ID and SWITCHaai identities. They offer a comprehensive and standardized way to structure user information and assist in simple attribute-based access control (ABAC) policy implementation, a concept previously introduced in Section 2.3.2. A SWITCH edu-ID identity consists of the following parts [87]:

**Personal part** - mandatory for all accounts, it must contain at least first name, last

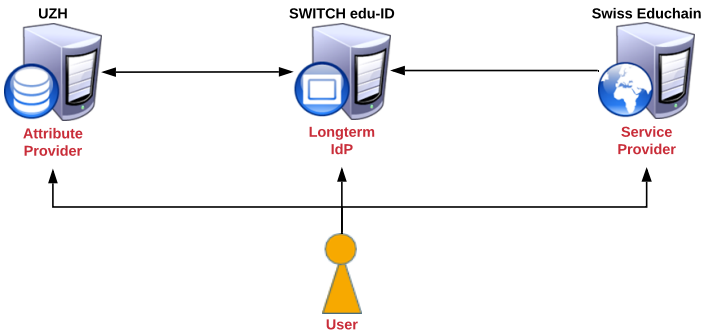


Figure 4.5: Swiss Educhain trust relationships.

name and an email address.

**Current affiliation** - added to an account when the user becomes a member of an organization (e.g. student or staff). May contain none, one or more current affiliations, and all current affiliations are created and managed by the respective organizations.

**Former affiliation** - a current affiliation is transformed into a former affiliation when an individual leaves an organization. The set of former affiliations acts as the affiliation history of an individual.

**Group memberships** - an identity's group memberships are represented in the `entitlement` attribute.

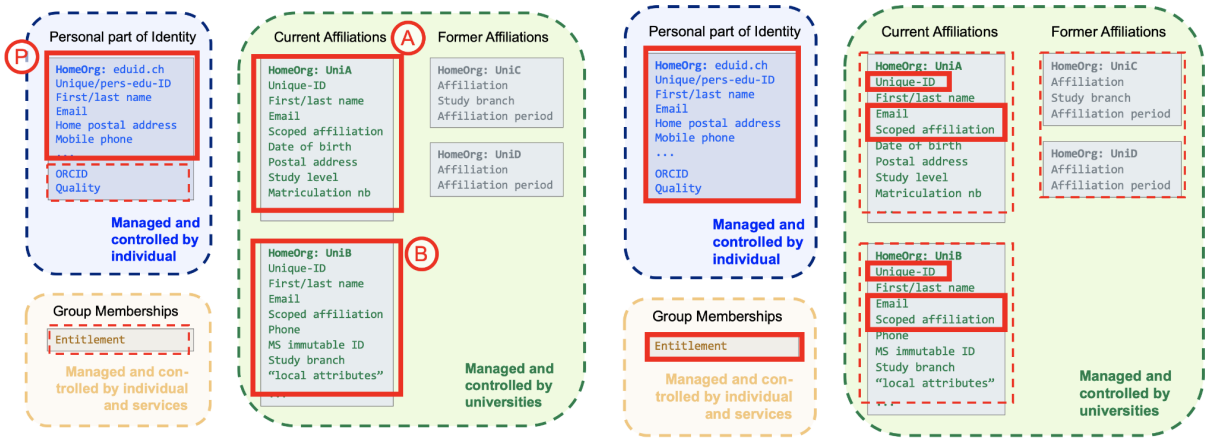


Figure 4.6: Classic and Extended Attribute models for Service Providers [88], [76].

The differences in the way a Service Provider can request, and access user attributes is shown in Figure 4.6. In the classic model, the service would either get the attribute assertion A or B, depending on the user's choice in the discovery service or the affiliation selection, a service can get only one part at a time. There may be cases where a service requires attributes from multiple home organizations simultaneously. This is possible in the extended model, where a service can potentially get a SAML assertion for attributes in the small bold red boxes in different parts of the identity. Swiss Educhain is using the extended model because it needs to fetch all current scoped affiliations

(`swissEduIDLinkedAffiliation` attribute) of the users. Users login with their edu-ID account and all the attributes that are requested by Swiss Educhain are updated with the current values and then disclosed. The extended model login flow is shown in Figure 4.7 and contrasted side by side with the classic model, where an affiliation is chosen before logging into a service.

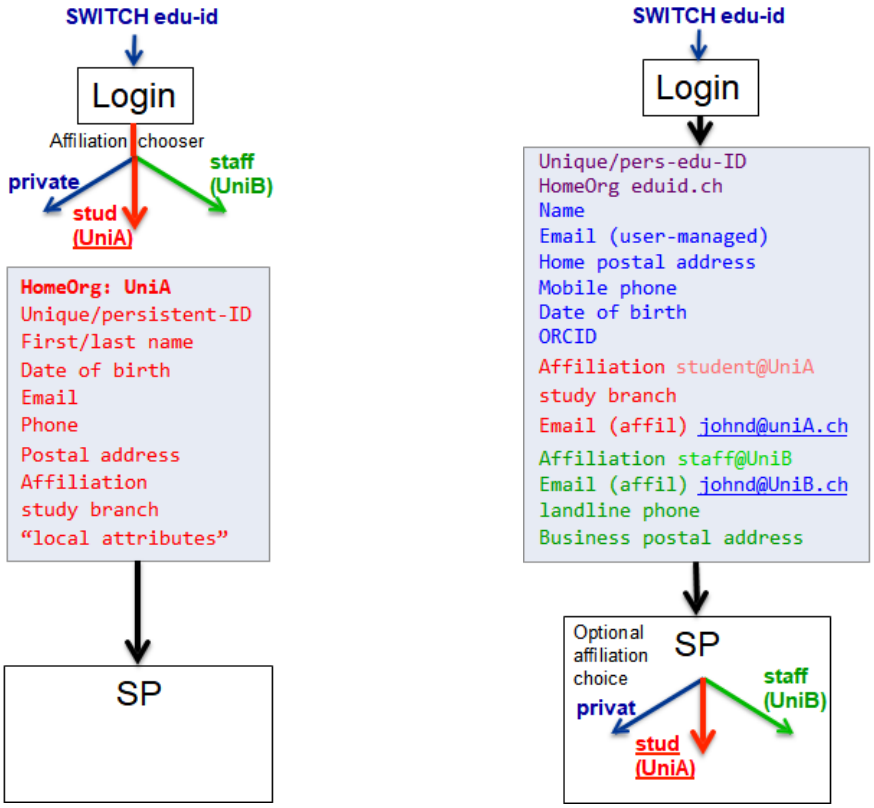


Figure 4.7: Classic and Extended models login flows [89].

The two distinct roles that can be assumed by the Swiss Educhain service users are **Issuer** and **Recipient**. In order to be able to distinguish amongst two kinds of users and issue a diploma the following attributes must be disclosed:

**commonName (cn)** - user's common name (first name, last name).

**mail** - preferred email address to be used to send messages to this person.

**matriculationNumber** - the unique long-lived matriculation number of a student.

**swissEduIDLinkedAffiliation** - a list of organizational scoped-affiliations (e.g. student@uzh.ch, staff@ethz.ch).

**persistentID** - a privacy-preserving user identifier shared between the Identity Provider (IdP) and the Service Provider (SP).

### 4.4.3 Persistent ID

The persistentID attribute is generated by the IdP when the user accesses a specific SP for the first time. It is stored in a relational database when the IdP is configured as the SWITCHaai deployment guides instruct. If no database is configured, a new value will be computed every time using the predefined salt. As it is persistent, the value remains the same for all further sessions between the same user and the same Service Provider. For different Service Providers, different Persistent IDs are generated for a given user. Therefore, the Persistent IDs cannot be used to correlate user data, even if several Service Providers tried to aggregate data. This results in better user privacy [90].

### 4.4.4 Target Audience

The target audience of the Swiss Educhain service is considered to be any edu-ID user which has at least one linked affiliation and a matriculation number. Linked affiliations are affilitations between edu-ID users and Organizations (acting as Attribute Providers), and should not be confused with the plain `affiliation` attribute which is the affiliation between a user and SWITCH edu-ID directly.

### 4.4.5 Role Assignment

The activities of role assignment, management and revocation are outside of the Swiss Educhain system boundary. Roles should be strictly assigned and managed by the organizations, it is their sole responsibility to ensure that only the correct users are assigned the corresponding affiliation(s) which are interpreted by Swiss Educhain into the two distinct roles of `Issuer` and `Recipient`. Through the edu-ID IdP, Swiss Educhain is always provided with the latest up to date values of all the user account attributes. The most important attribute is `swissEduIDLinkedAffiliation` which holds all the current active user affiliations with one or more organizations, the values should only be present while the affiliation lasts, as soon as a user leaves an organization the affiliation should be marked as a former affiliation by edu-ID and the value removed from the attribute.

### 4.4.6 User Access Control

A user has different interactions and relationships with the **Service Provider** (Swiss Educhain), the **Identity Provider** (SWITCH edu-ID) and the **Attribute Provider(s)** (Organization(s)). In Figure 4.8 the login sequence steps are shown for a new session, this includes the different levels of access controls with the SP and the IdP. Even though the User Experience is excellent and the Web SSO works seamlessly, there are a lot of prerequisites, established processes and well-defined steps happening in the background to ensure privacy, security and verifiability.

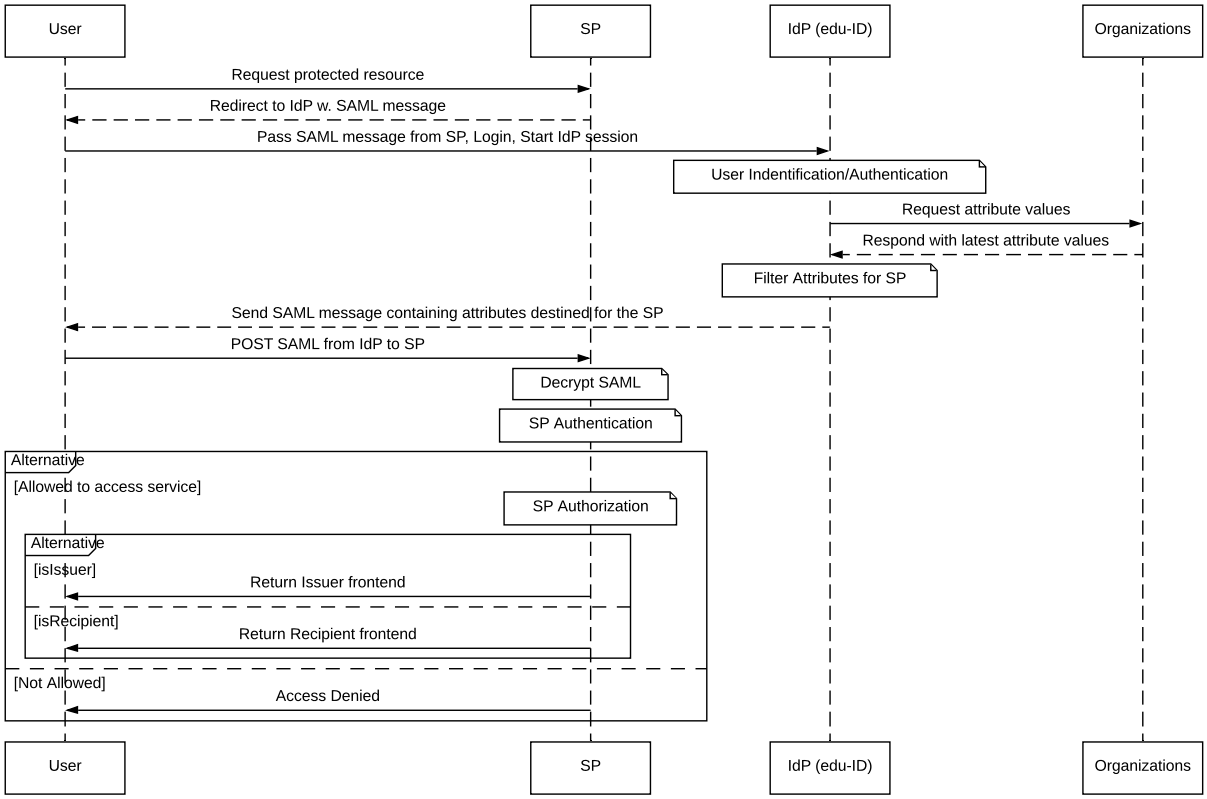


Figure 4.8: Swiss Educhain login sequence diagram based on [91].

Initially, the user tries to visit the Swiss Educhain service (`https://educhain.csg.uzh.ch/app/`), the `mod_shib` Apache module in the SP checks if a valid session exists to access the protected location relative to the base url (e.g. **/app/**), details on the implementation are provided at Chapter 5. Since there is no active session the user is redirected to the IdP to login and authenticate, carrying also a SAML message from the SP which requests certain attributes. In other federated Web SSO scenarios the SP conducts a WAYF (Where Are You From) step to discover the IdP which holds the user's identity, while in the SWITCH edu-ID scenario the IdP is unique and already known to the SP (`https://login.test.eduid.ch/`).
The user reaches the IdP and needs to perform the identification and authentication steps (usually using a username and password). After the user successfully authenticates, the IdP updates all the attributes and linked affiliations of the user by sending requests to the Organizations that act as Attribute Providers. Once all the attributes have been received, the IdP performs any necessary updates internally and then consults the Resource Registry, to filter the attributes based on the SP Resource Description and the active Attribute Release Policy Settings. Then, the SAML response is encoded and sent back to the user to be forwarded to the SP, this response only contains the attributes that the SP is allowed to request and that are available. The user is shown a comprehensive message of which attributes will be disclosed to the SP. After providing consent, the SAML response is forwarded to the SP together with a new request to access the protected resource.
The SP decrypts the message, verifies the IdP signature and depending on the ABAC policy in effect, allows or denies access to the resource. User identification is done with the `persistentID` to preserve privacy, and authentication is inherited from the authenti-

cation statement produced by the IdP (possibly also stating if it was 1FA, 2FA or MFA). Swiss Educhain has two levels of authorization, the service-level authorization which determines if a user should be granted access to the service in general and the application-level authorization which determines what actions the user will be allowed to perform. A detailed description is provided below.

### 4.4.7 Authorization Policy

**Service-Level Authorization**

The service level policy is stored in the Apache Webserver configuration and uses the `mod_shib` module (which integrates the local Shibboleth daemon with Apache) to check the received values and enforce the policy.

**Rules:** (Require All)

**Valid Session**

A valid shibboleth session needs to be active between the SP and the user, this means the user has authenticated with the IdP, the SP has validated the received authentication statement and a new session was created.

**Linked Affiliation exists**

At least one linked affiliation exists, checked through the `swissEduIDLinkedAffiliation` attribute which needs to have at least one value.

**Matriculation number exists**

The user's matriculation number needs to be present and valid; the validity and non-duplication is ensured by SWITCH edu-ID. Attribute `matriculationNr` holds the value. In Swiss academia the matriculation number is only generated once and is used across organizations when needed.

If all the above conditions are true, Apache creates a session and sends a request to the Spring Boot embedded Tomcat server through the AJP protocol as shown in Figure 4.2. If any of the conditions is not satisfied access is denied and the user is redirected to an error page by the Shibboleth handler.

**Application-Level Authorization**

The application level policy only determines the role the user will assume. The two distinct application roles are **Issuer** and **Recipient** as defined in Section 4.1. It must be clarified, as mentioned in detail in Section 4.4.5, that Swiss Educhain does not manage any user accounts nor is able to assign access rights or affiliations on behalf of any Organization.

**Rules:**

**Recipient**

`Recipient` is the role that is assigned by default to all the users that have access to the service. There is no additional check to verify that a user should assume the role of a `Recipient`, this is ensured from the check performed by Shibboleth and Apache.

**Issuer**

An `Issuer` has elevated access rights and is able to issue diplomas to one or more `Recipients` individually or in batch. To identify someone as an `Issuer` the CorDapp checks the `swissEduIDLinkedAffiliation` attribute for the values `staff@uzh.ch` or `faculty@uzh.ch`. For the Swiss Educhain to be released to production and Organizations to be able to assign the role of `Issuer` a new value should be available in the `swissEduIDLinkedAffiliation` attribute, `issuer` (e.g. `issuer@uzh.ch, issuer@epfl.ch`). The values of `staff` and `faculty` are used for the purpose of the MVP implementation.

A view of the frontend interface is shown in Figure 5.4 in Chapter 5.

### 4.4.8 Application Accounts

The extensive integration with SWITCH edu-ID has been described and the solution design has been presented. It is essential to define the mapping amongst edu-ID identities with Swiss Educhain application accounts. Figure 4.9 shows the information flow and granular identity mapping of parts for the various entities.

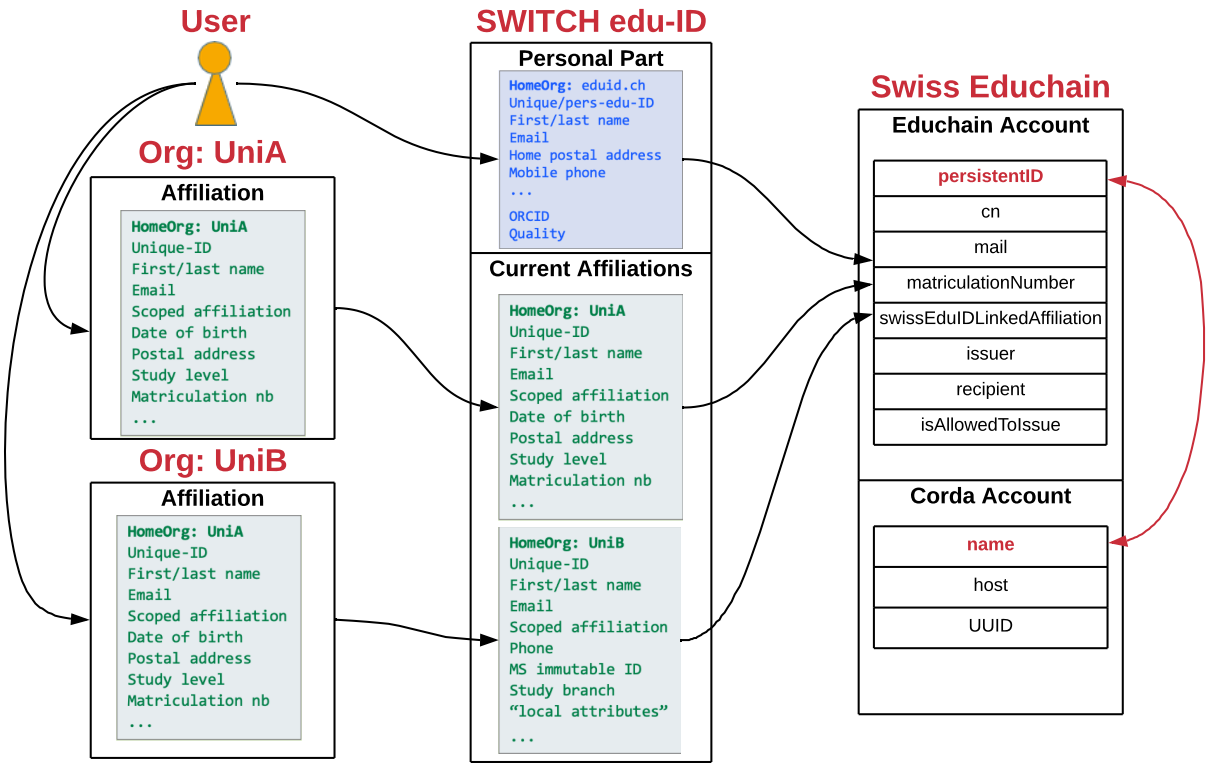


Figure 4.9: Information flow and identity mapping.

**User**

The user provides personal and contact information to the affiliated Organizations and the edu-ID IdP. Information provided by the user is not trusted by default and must be always verified, either digitally or via in-person verification. The user

owns and manages the personal part of the edu-ID account, following a user-centric approach.

**Organizations**

Organizations no longer act as a complete identity provider which hosts and manages the users' accounts. In the edu-ID architecture, they host only the affiliations that exist with users, entrusting the user identity management responsibility to edu-ID. Thus, they act as Attribute Providers to edu-ID, by assigning individual roles and access rights using the attribute values. In other use cases, the relationship with edu-ID can be two way, but from Swiss Educhain's perspective the flow of information is only unidirectional.

**edu-ID IdP**

Edu-ID is the central root of trust of the SWITCH federation and is the sole IdP, serving all the other entities. Edu-ID acts as a data provider and a source of trust for Swiss Educhain. It provides only the relevant (and approved for release) parts or attributes of a user's identity to the service. Apart from the information depicted, edu-ID holds a wide variety of metadata and a history of all the former affiliations of a user.

**Swiss Educhain**

Swiss Educhain acts solely as a consumer of information from a single data source, the edu-ID IdP. A strong trust relationship is assumed, to treat all user data disclosed as authentic, complete and valid. Internally, specific attributes are used to provide two levels of authorization and identify a user (persistentID). A Corda account is created to be used in transactions and is logically mapped one-to-one with the EduchainAccount. The Educhain account is stored as a Corda state with specific attributes and is updated if needed after every login. The `persistentID` attribute acts as the primary key and is used to perform all account related activities. More information on the implementation of accounts is provided at Chapter 5.

# Chapter 5

# Implementation

As described in Chapter 4 several design decisions were made to address the core functionality requirements. With Corda chosen as the private permissioned blockchain platform, an appropriate IAM solution was engineered as described in Section 4.4. Section 5.1 goes through the high-level necessary technical steps for Swiss Educhain to integrate with SWITCH edu-ID. Section 5.2 describes the parts of the codebase relevant to Swiss Educhain's IAM solution. Section 5.3 demonstrates how the identity management part of Swiss Educhain is implemented in the CorDapp code. Lastly, Section 5.4 demonstrates the implementation details of the Spring Boot essential sub-components. High-level information and guidelines on how to install and configure the Swiss Educhain are provided in Appendix A.

## 5.1 Integration with SWITCH edu-ID

As already analyzed in Section 4.4.2 the chosen IAM solution is participation in the SWITCH federation and integration with SWITCH edu-ID. Swiss Educhain participates as a Service Provider (SP) and needs to implement certain technical integration steps as described in the next Sections.

### 5.1.1 Shibboleth Installation and Configuration

The Shibboleth Service Provider software needs to be installed to the server that hosts the Swiss Educhain service. The Swiss Educhain service is hosted on an Ubuntu Server provided by the Communications Systems Research Group (CSG) at the Department of Informatics. The service can be accessed at `https://educhain.csg.uzh.ch/app/`. Figure 5.1 shows how the Shibboleth SP daemon integrates with Webservers.

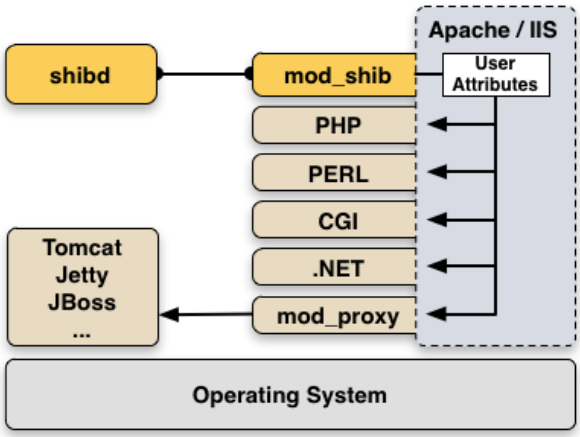


Figure 5.1: Shibboleth daemon integration [92].

The Shibboleth Service Provider consists of a daemon `shibd` running on all major operating systems and a web server module `mod_shib` which is natively supported by the Apache HTTPD server. The Service Provider can protect any web server content by enforcing user authentication [92]. Detailed step-by-step instructions for the installation and configuration of the Shibboleth Service Provider (SP) 3.0, as well as instructions on how to register an SP at the Resource Registry are provided by SWITCH [93], [94].

### 5.1.2 HTTPS Configuration

To provide integrity, security and confidentiality the Swiss Educhain service uses HTTPS traffic. This requires an SSL/TLS (Secure Sockets Layer, Transport Layer Security) Certificate by a trusted CA (Certificate Authority). Swiss Educhain uses **certbot** to create and renew automatically certificates signed by **Let's Encrypt** [95], [96].

```
1 RewriteEngine on
2 RewriteCond %{SERVER_NAME} =educhain.csg.uzh.ch
3 RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI} [END,NE,R=permanent]
```

Listing 5.1: Excerpt of the `apache2/sites-enabled/educhain.conf` file.

Once the certificate has been created, Apache is configured to redirect all `http` requests to `https`. Listing 5.1 shows the `Rewrite` directives which are defined inside the port 80 `VirtualHost` configuration.

```
1 SSLEngine On
2 SSLProxyEngine On
3 SSLCertificateFile /etc/letsencrypt/live/example.com/fullchain.pem
4 SSLCertificateKeyFile /etc/letsencrypt/live/example.com/privkey.pem
5 Include /etc/letsencrypt/options-ssl-apache.conf
```

Listing 5.2: Excerpt of the `apache2/sites-enabled/educhain-le-ssl.conf` file.

In Listing 5.2 Apache is configured to turn on the SSL engine, the absolute paths are set for the generated certificate and private key, and the `options-ssl-apache.conf` file is imported from the `/etc/letsencrypt/` directory. This configuration file allows for further customization of the SSL protocol, the ciphersuite and enabling or disabling compression amongst other options.
To ensure that communication is secure between Apache and Spring Boot, SSL needs to be configured for the AJP Connector as seen in Figure 4.2 and explained in detail at Section 5.4.1. As mentioned in the guidelines provided in Appendix A, a PKCS12 (Public-Key Cryptography Standards 12) certificate needs to be generated based on the valid SSL certificate, instructions on how to create it are given in [97]. The variables in Listing 5.3 need to be set in the `application.properties` file of the `clients` module, so Spring Boot is able to use the keystore.

```
server.ssl.key-store = /path/keystore.p12
server.ssl.key-store-password = notapassword
server.ssl.keyStoreType = PKCS12
server.ssl.key-alias = mytomcat
```

Listing 5.3: Spring Boot embedded server SSL properties.

### 5.1.3 Shibboleth Access Control

When configuration is in place for all the components to communicate securely and Swiss Educhain has been registered and integrated as a Service Provider, access controls can be defined through Shibboleth.

```
<Location /app/>
  AuthType shibboleth
  ShibRequestSetting requireSession true
  ShibUseEnvironment On
  <RequireAll>
        Require shib-attr swissEduIDLinkedAffiliation ~ .*@.*
        Require shib-attr matriculationNumber ~ .*
  </RequireAll>
</Location>
```

Listing 5.4: Access control for `/app/` location based on Shibboleth attributes.

The configuration in Listing 5.4, stored in `apache2.conf`, sets the following requirements to allow access to the protected resource in `https://educhain.csg.uzh.ch/app/`:

- A valid active Shibboleth session.
- Shibboleth should use the environment to disclose attributes instead of the headers. This affects the way Spring Boot retrieves the already disclosed attributes which reside in Apache. More details on this process are provided in Section 5.4.1.
- Each user should have at least one linked affiliation with an Organization **and** the matriculation number should be present. If only one of the two is true, then access is denied.

The above access control is an implementation of the Service-level authorization as defined in Section 4.4.7. More examples of Shibboleth Service Provider access control rules are provided in [98].

### 5.1.4 Attributes

As already described in detail in Section 4.4.2 several attributes are necessary for Swiss Educhain to function. Swiss Educhain states in the Resource Description in the SWITCHaai Resource Registry (RR) which attributes are required and which are desired. Required are the core attributes which are available from all Organizations and desired are other attributes. If an attribute is neither required nor desired by the SP, then it will not be disclosed at all. Swiss Educhain attributes:

**commonName (cn)** - core, required.

**mail** - core, required.

**persistentID** - core, required.

**matriculationNumber** - other, desired.

**swissEduIDLinkedAffiliation** - other, desired.

Shibboleth checks only if `matriculationNumber` and `swissEduLinkedAffiliation` exist and have the proper value, because these attributes are not guaranteed to be present in a user's account. The Spring Boot module can request the value of one or more attributes by calling the `request.getAttribute()` method and specifying the attribute to be fetched by `name`. An example of all the attributes disclosed in a session are shown in Figure 5.2.

Attributes
**Meta-displayName**: SWITCH edu-ID [Test]
**Meta-informationURL**: https://projects.switch.ch/eduid/
SWITCH
edu-ID
**Meta-largeLogo**:
**Meta-organizationURL**: http://www.test.eduid.ch/
**Meta-smallLogo**: ID
**affiliation**: affiliate
**cn**: Vasileios Koukoutsas
**displayName**: Vasileios Koukoutsas
**eduPersonUniqueId**: 0000991096437499@test.eduid.ch
**givenName**: Vasileios
**homeOrganization**: test.eduid.ch
**homeOrganizationType**: others
**mail**: vasileios.koukoutsas@uzh.ch
**matriculationNumber**: 16718991
**persistent-id**: https://test.eduid.ch/idp/shibboleth!https://educhain.csg.uzh.ch/shibboleth!so3WNOJLA+Fp/SuC8E7KbK6wgFc=
**preferredLanguage**: en
**principalName**: 0000991096437499@test.eduid.ch
**schacHomeOrganization**: test.eduid.ch
**schacHomeOrganizationType**: urn:schac:homeOrganizationType:ch:others
**scoped-affiliation**: affiliate@test.eduid.ch
**surname**: Koukoutsas
**swissEduIDAssociatedMail**: vasileios.koukoutsas@uzh.ch
**swissEduIDAssuranceLevel**: swissEduPersonMatriculationNumber:https://eduid.ch/def/loa2;mail:https://eduid.ch/def/loa2;giv
**swissEduIDLinkedAffiliation**: member@uzh.ch;student@uzh.ch
**swissEduIDLinkedAffiliationMail**: vasileios.koukoutsas@uzh.ch

Figure 5.2: SSO session attributes disclosed to Swiss Educhain.

## 5.2 Code Structure

The Swiss Educhain service in its core is developed as a CorDapp (Corda Decentralized Application). The codebase has been based on the CorDapp Kotlin template [99] and was expanded to meet the MVP functionality needs both for the identity management and the verification process.

The main code modules of the service are:

**clients**
This is the `Spring Boot` component and it contains:

- the web `frontend` code, which is written in `HTML` and `AngularJS`,
- the embedded `Tomcat Server` (with the attached `AJP` connector),
- the `Controller` exposing `RESTful` endpoints to interact with the `CorDapp`.

**contracts**
This is a `CorDapp` component and contains the `Contracts` and `States` definitions.

**verification_frontend**
Contains the verification frontend files, it is written in plain `HTML` and `JavaScript`.

**workflows**
This module contains the majority of the `CorDapp` functionality:

- the **flows** for `Account` and `Diploma` functionality,
- `RPC` startable **queries** to retrieve data from the node `Vault`,
- the `Identity` and `Ethereum` node **services**,
- the `Solidity` **smart contract**.

The CorDapp consists of two code modules, namely `contracts` and `workflows`. There are two reasons behind this decision. Firstly, the contract JAR is attached to a transaction and independent upgrades, so producing it separately reduces its size significantly. Secondly, contracts have `constraints` and upgrading is complex, therefore decoupling contract code from flow code allows flows to be upgraded independently.
A high-level file structure of the code is presented in Appendix B, and the complete Swiss Educhain source code can be found in the contents of the accompanying CD. In the following Sections, only the functionality strictly related to identity and access management is analyzed, a detailed analysis on the functionality related to the verification process is provided in Simon Müller's work [12].

## 5.3 CorDapp

The application logic and core functionality of the Swiss Educhain is implemented as a CorDapp. Same as any other CorDapp, it needs to have a few building blocks to be complete such as states, contracts, flows and (optionally) services.

### 5.3.1 Swiss Educhain Application Accounts

As previously explained in Section 4.4.2 to create an Educhain application account some fields are retrieved through attributes, some fields are populated inside the CorDapp and a one-to-one mapping to Corda technical accounts must be defined. Figure 4.9 demonstrates the different fields and the exact data flow. To provide debugging information in the web frontend the sections of `Educhain Accounts` and `Corda Accounts` are displayed as shown in Figure 5.3.

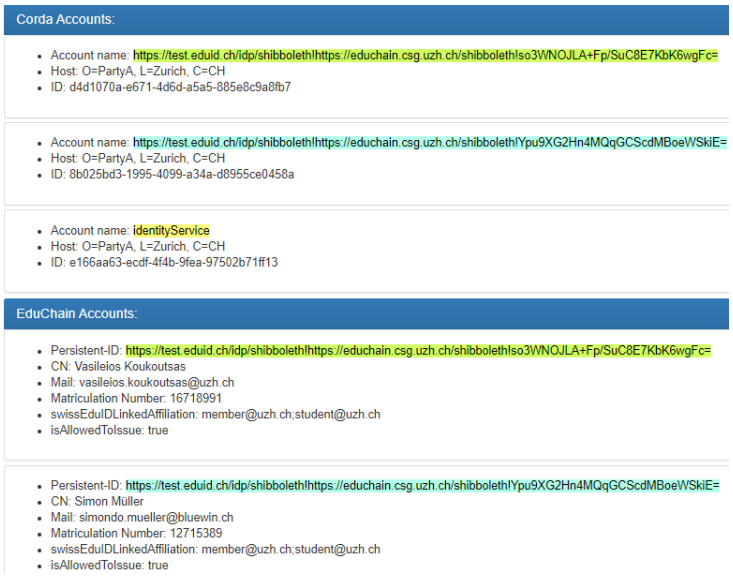


Figure 5.3: Educhain and Corda accounts frontend sections.

### 5.3.2 Corda Accounts & Node Identity Service

As analyzed in Section 3.3.1 the Corda Accounts library offers the possibility to create technical Corda accounts with only three fields:

**name** - can be set during the account creation (type `String`),

**host** - is of type `Party` and is the identity of the node hosting the account,

**identifier** - is a UUID (Universally Unique Identifier) value automatically generated of type `UniqueIdentifier`.

To create new accounts the `createAccount(name: String)` function is used which is defined in the `KeyManagementBackedAccountService` class [43]. Listing 5.5 shows the Corda account state and creation function. After a Corda account has been created it can no longer be updated. The Corda accounts need to be leveraged from the application to create any custom accounts lifecycle or identity and access management solutions.

```
@BelongsToContract(AccountInfoContract::class)
data class AccountInfo(
  val name: String,
  val host: Party,
  val identifier: UniqueIdentifier
)
@Suspendable
override fun createAccount(name: String): CordaFuture<StateAndRef<
    AccountInfo>> {
  return flowAwareStartFlow(CreateAccount(name))
}
```

Listing 5.5: Corda `AccountInfo` state and `createAccount` function.

To facilitate a standardized and easily accessible way to create Corda accounts an identity service was created. Corda services run on a single node and offer functionality inside the node, they are initialized automatically when the node boots up, and they can only be called from within a flow or from another service through the `serviceHub` interface [100].

```
@CordaService
class EduChainIdentityService(private val serviceHub: AppServiceHub):
    SingletonSerializeAsToken() {
 @Suspendable
 fun createIdentityServiceAccount() : StateAndRef<AccountInfo> {
  val name: String = "identityService"
  try { // Check if account already exists
   require(serviceHub
    .cordaService(KeyManagementBackedAccountService::class.java)
    .accountInfo(name).none
    {serviceHub.myInfo.legalIdentities.contains(it.state.data.host)})
  } catch (ex: Exception){
   println(ex.message)
   return serviceHub.cordaService(KeyManagementBackedAccountService::
       class.java).accountInfo(name).get(0)
  } // Creates and returns the identityService account
  return serviceHub.accountService.createAccount(name)
      .toCompletableFuture().getOrThrow()
 }
}
```

Listing 5.6: Educhain Identity Service.

Listing 5.6 shows the service definition and the only service method `createIdentityServiceAccount`. This method is used to create the hardcoded Corda identity service account if does not exist. The `identityService` account is in turn used by the `(Create|Update)EduChainAccountFlow` to create the Educhain accounts during the flow's execution.

### 5.3.3 Educhain Account State

The `EduchainAccountState` holds the information of a user's account. The account is created based on a user's unique identifier which is the `persistentID` attribute generated

by edu-ID to enable Swiss Educhain to identify a user. The code of this state can be seen in Listing 5.7.

```
1  @BelongsToContract(EduChainAccountContract::class)
2  data class EduChainAccountState(val persistentID: String,
3    val cn: String,
4    val mail: String,
5    val matriculationNumber: String,
6    val swissEduIDLinkedAffiliation: String,
7    val issuer: AnonymousParty,
8    val recipient: AnonymousParty,
9    val isAllowedToIssue:  Boolean) : ContractState {
10   override val participants get() = listOf(issuer, recipient)
11 }
```

Listing 5.7: Code of `EduChainAccountState`.

The annotation in Line 1 signals that any modification of the `EduChainAccountState` by a flow must obey the rules defined in the `EduChainAccountContract`. The contract's rules and code is provided in Listing 5.8. The first five attributes disclosed by SWITCH edu-ID have been described in Section 4.4.2. A description for the fields that are generated by the CorDapp is given:

**issuer** - the Corda account of the issuer of the Educhain account, of type `AnonymousParty` is a public key representation of the actual account.

**recipient** - the Corda account of the recipient of the Educhain account, of type `AnonymousParty` is a public key representation of the actual account.

**isAllowedToIssue** - determines if a user is an `Issuer` or not, it is generated in the `(Create|Update)EduChainAccountFlow`. It is always checked for validity by the `load-account` endpoint during each new session or webpage refresh.

### 5.3.4 Educhain Account Contract

The `EduChainAccountContract` is used by the `EduChainAccountState` and defines the possible actions that can modify an `EduChainAccountState`. The contract defines two possible commands `Create` and `Update`, which is derived from the high-level solution design that only allows for accounts to be created or updated. Currently, Educhain account deletion is not implemented to ensure there is transparency, traceability and verifiability of actions in the diploma issuance and verification process.

```
1  class EduChainAccountContract : Contract {
2   interface Commands : CommandData {
3    class Create : TypeOnlyCommandData(), Commands
4    class Update : TypeOnlyCommandData(), Commands
5   }
6   @Throws(IllegalArgumentException::class)
7   override fun verify(tx: LedgerTransaction) {
8    val command = tx.commands.requireSingleCommand<Commands>()
```

```
9   val output = tx.outputsOfType<EduChainAccountState>().single()
10
11  when (command.value) {
12   is Commands.Create -> requireThat {
13    "No inputs should be consumed when creating an EduChain account."
         using (tx.inputs.isEmpty())
14    "Only one output state should be created when creating an EduChain
         account." using (tx.outputs.size == 1)
15    val eduChainAccount = tx.outputStates.single() as
         EduChainAccountState
16    "The account creator and account owner cannot have the same
         identity." using
17     (output.participants[0] != output.participants[1])
18    "Only account creator and account owner may sign the account create
         transaction." using
19     (command.signers.toSet() == output.participants.map { it.owningKey
          }.toSet())
20   }
21   is Commands.Update -> requireThat {
22     //Same as Create but allows one input state.
23   }
24  }
25 }
26 }
```

Listing 5.8: Code of the `EduChainAccountContract`.

There are four rules defined by the contract for the `Create` command as seen in Listing 5.8:

- No inputs should be consumed when creating an Educhain account.
- Only one output state should be created when creating an Educhain account.
- The account creator and account owner cannot have the same identity.
- Only account creator and account owner may sign account create transaction.

The same rules apply to the `Update` command with the only difference being one input state is expected, which will be consumed and marked as historic, to produce one output (updated) state.

### 5.3.5 Educhain Account Flows

Corda flows are the mechanism that encapsulates the core business logic of a CorDapp. Simon Müller in [12] provides an excellent skeleton of the Swiss Educhain flows in the form of pseudo-code, including a list of the main flow functionality. The pseudo-code is provided in Listing 5.9.

```
@InitiatingFlow
@StartableByRPC
class Flow(private val exampleName: String,
private val exampleId: UUID) : FlowLogic<ReturnObject>() {
 companion object {
  /* ProgressTracker steps are defined here
   * They can be used during flow execution to track progress. */
  object FIRST_STEP : ProgressTracker.Step("First step")
  object SECOND_STEP : ProgressTracker.Step("Second step")
  fun tracker() = ProgressTracker(
      FIRST_STEP,
      SECOND_STEP )
 }
 override val progressTracker = tracker()
 @Suspendable
 override fun call(): ReturnObject {
  progressTracker.currentStep = FIRST_STEP
  /* Business logic of the flow is contained here.
  *  - Checking requirements,
  *  - Choosing the notary,
  *  - Requesting the public keys for the accounts,
  *  - Creating the transaction (input, output, command, attachment),
  *  - Gathering the signatures from all parties for the transaction,
  *  - Finalising the transaction. */
  progressTracker.currentStep = SECOND_STEP
  return ReturnObject()
 }
}
@InitiatedBy(Flow::class)
class FlowResponder(val counterPartySession: FlowSession) : FlowLogic<Unit>() {
 @Suspendable
 override fun call() {
 /* The response flow called as part of finalising a transaction.
  * Not every flow uses a response flow.
  * Code here typically involves verifying the transaction and recording the states. */ }
}
```

Listing 5.9: Pseudo-code of a typical Corda flow [12].

### CreateEduChainAccountFlow

```
// Retrieve the notary identity from the network map.
val notary = serviceHub.networkMapCache.notaryIdentities[0]
// Create or Fetch identityService account's UUID
val issuingAccountId: UUID = serviceHub.cordaService(EduChainIdentityService::class.java)
    .createIdentityServiceAccount().state.data.identifier.id
// Retrieve the issuing/receiving accounts and their public keys
val issuingAccount = accountService.accountInfo(issuingAccountId)
?: throw FlowException("No account for ID $issuingAccountId found in vault.")
val receivingAccount = accountService.createAccount(name = persistentID)
    .toCompletableFuture().getOrThrow()
val issuingAccountAnonParty = subFlow(RequestKeyForAccount(issuingAccount.state.data))
val receivingAccountAnonParty = subFlow(RequestKeyForAccount(receivingAccount.state.data)
    )
// Check if an account has diplomas to be issued in the Waiting List
val diplomasIssuedFromWaitingList = mutableListOf<StateAndRef<DiplomaState>>()
val diplomaList = subFlow(CheckDiplomaWaitingListForMatriculationNumber(
    matriculationNumber))
for (diploma in diplomaList) {
 val diplomaIssuer = accountService.accountInfo(diploma.state.data.issuer.owningKey)
 if (diplomaIssuer?.state?.data?.identifier?.id == null) {
  // should not happen
  continue
 }
 val diplomaState=subFlow(IssueDiplomaToAccountFromWaitingList(diploma,
 diploma.state.data.diplomaAttachment,
 diploma.state.data.diplomaHash,
 diplomaIssuer.state.data.identifier.id,
```

```
24  receivingAccount.state.data.identifier.id))
25  diplomasIssuedFromWaitingList.add(diplomaState)
26 }
27 System.out.println("Diplomas issued from waiting list: ${
       diplomasIssuedFromWaitingList.size}")
28 // Determine if the account owner is allowed to issue
29 val issuersAffiliation=listOf<String>("faculty@uzh.ch","staff@uzh.ch")
30 val hardCodedIssuers=listOf<String>("16718991", "12715389", "12345678")
31 var isAllowedToIssue=false
32 for (it in issuersAffiliation) {
33  if (swissEduIDLinkedAffiliation.contains(it)) {
34   isAllowedToIssue = true }
35 }
36 if(hardCodedIssuers.contains(matriculationNumber)) isAllowedToIssue=true
37 // Create the transaction components.
38 val outputState = EduChainAccountState( persistentID, cn, mail, matriculationNumber,
       swissEduIDLinkedAffiliation, issuingAccountAnonParty, receivingAccountAnonParty,
       isAllowedToIssue)
39 val command = Command ( EduChainAccountContract.Commands.Create(), listOf(
       issuingAccountAnonParty.owningKey, receivingAccountAnonParty.owningKey))
40 // Create a transaction builder and add the transaction components
41 val txBuilder = TransactionBuilder(notary = notary)
42  .addOutputState(outputState).addCommand(command)
43 // Sign the transaction.
44 val locallySignedTx = serviceHub.signInitialTransaction(
45  txBuilder, listOfNotNull(issuingAccountAnonParty.owningKey,
        receivingAccountAnonParty.owningKey))
46 // Create a session with the other party.
47 val counterPartySession = initiateFlow(receivingAccount.state.data.host)
48 // Obtain the counterparty's signature and add to locally signed transaction
49 val receiverSignature = subFlow(CollectSignatureFlow(locallySignedTx, counterPartySession
       , receivingAccountAnonParty.owningKey))
50 val signedByCounterParty = locallySignedTx.withAdditionalSignatures(receiverSignature)
51 // Return fully signed transaction
52 return subFlow(FinalityFlow(signedByCounterParty, listOf(counterPartySession).filter
53  { it.counterparty != ourIdentity }))
```

Listing 5.10: Code snippet of the `CreateEduChainAccountFlow`.

As shown in detail in Listing 5.10 the following steps are executed in order with synchronous calls:

1. The notary's identity is retrieved from the network map.
2. `identityService` account's UUID is created or fetched.
3. Issuing and receiving accounts are retrieved and their public keys.
4. It is checked if an account has diplomas to be issued in the waiting list.
5. It is determined if the account owner should be allowed to issue.
6. Transaction components are created.
7. A transaction builder is created, and the transaction components are added.
8. A session is created with the counterparty.
9. Counterparty's signature is obtained and added to the locally signed transaction.
10. Fully signed transaction is returned.

### UpdateEduChainAccountFlow

The `UpdateEduChainAccountFlow` flow is quite similar to the `CreateEduChainAccount-Flow` flow, with a few distinct differences in the steps involved. The code implementation

of `UpdateEduChainAccountFlow` is omitted as Listings 5.9 and 5.10 provide a quite thorough walkthrough. The `UpdateEduChainAccountFlow` high-level steps are:

1. The notary's identity is retrieved from the network map.
2. The Educhain account to be updated and its state are retrieved.
3. `identityService` account's UUID is created or fetched.
4. Issuing and receiving accounts are retrieved and their public keys.
5. It is checked again if the account owner should be allowed to issue.
6. Transaction components are created, including an input state to be consumed.
7. A transaction builder is created and the transaction components are added.
8. A session is created with the counterparty.
9. Counterparty's signature is obtained and added to the locally signed transaction.
10. Fully signed transaction is returned.

## 5.4 Spring Boot

The Spring Boot component acts as a bridge between the end user and the CorDapp backend as shown in Figure 4.2. The AJP connector acts as a highway to serve the frontend and fetch attributes exposed using the Shibboleth environment. The frontend is a single webpage serving both `Issuers'` and `Recipients'` needs, and the controller exposes well-defined backend CorDapp functionality to the frontend via REST API (Application Programming Interface) endpoints. These three sub-components are analyzed further in this Section from the IAM perspective.

### 5.4.1 AJP Connector

The AJP connector is normally used in high-traffic scenarios to optimize performance between one or more webservers behind an Apache instance. In the Swiss Educhain service use case, it is preferred to provide a safer way to disclose the attributes through the environment instead of using http headers. It is defined based on custom properties and then added to the existing Tomcat connectors. Listing 5.11 shows the properties defined and the method that creates a new `Context` and adds it to the Tomcat connectors.

```
1 @Bean
2 open fun servletContainer(): ServletWebServerFactory? {
3  val tomcat: TomcatServletWebServerFactory = object :
      TomcatServletWebServerFactory() {
4   override fun postProcessContext(context: Context) {
5    val securityConstraint = SecurityConstraint()
6    securityConstraint.userConstraint = "CONFIDENTIAL"
7    val collection = SecurityCollection()
```

```
8       collection.addPattern("/*")
9       securityConstraint.addCollection(collection)
10      context.addConstraint(securityConstraint)
11    }
12  }
13  tomcat.addAdditionalTomcatConnectors(redirectConnector())
14  return tomcat
15 }
16 var maxSize = 50000000
17 open fun redirectConnector(): Connector? {
18   val connector = Connector("AJP/1.3")
19   connector.scheme = "https"
20   connector.port = 8009
21   connector.secure = true
22   connector.uriEncoding = "UTF-8"
23   connector.allowTrace = false
24   connector.maxPostSize = maxSize
25   connector.maxSavePostSize = maxSize
26   connector.redirectPort = 8443
27   return connector
28 }
```

Listing 5.11: Code of the `AJP Connector`.

The connector uses the `port 8009` following common practice. The scheme is set to `https` and requires that the connection is `secure`. Furthermore, the `maximum POST` size is increased to allow for multiple attributes to be disclosed simultaneously. Apache configuration also needs to be updated to serve traffic through the `ajp` protocol. Listing 5.12 shows the directive added to the `educhain-le-ssl.conf` file. It should be noted that the `ajp` protocol is used instead of the `http(s)` and that it is not possible to use both simultaneously.

```
1 ProxyPass /app/  ajp://localhost:8009/app/
```

Listing 5.12: Apache `AJP` proxy directive.

### 5.4.2 Controller

The Controller is used as an intermediate to expose REST API endpoints to the web frontend and communicate with the CorDapp backend. The base path for all REST requests is `https://educhain.csg.uzh.ch/app/api/`. The main endpoints relevant to the identity and access management functionality are:

#### load-account

- Runs every time a new session is initiated. Checks if an Educhain account exists with the edu-ID user's `persistentID`, if it does not exist it is created, or if it exists but the attributes need to be updated they are updated. The endpoint calls `(Create|Update)EduChainAccountFlow`.
- Method: `GET`

**my-account**

- Returns the user's `EduchainAccountState`.
- Method: `GET`

**edu-accounts**

- Returns a list of all the existing Educhain accounts' states.
- Method: `GET`

**own-accounts**

- Returns a list of all the existing Corda accounts.
- Method: `GET`

**me**

- Returns the node's identity.
- Method: `GET`

All account related API calls use the `GET` method, no account related fields need nor should be sent to the controller by the user or the frontend. Account information are fetched directly from edu-ID in the form of attributes for each session.

### 5.4.3 Frontend Interface

*This is joint text with Simon Müller [12].*

The Swiss Educhain frontend is the main interactive part of Swiss Educhain. As soon as the user logs in via the edu-ID login screen, he is greeted by the Educhain frontend. It is a simple one-page website built with AngularJS [101] and styled with Bootstrap [102].
In the top left corner, the name of the node of the current connection is shown. Next to the node name the button 'Issue diploma' is placed. If the user that is logged in is not allowed to issue, it will be grayed out. The same applies to the 'Unsigned Diplomas' button next to it. The 'Logout' button in the top right logs out of the Swiss Educhain service and SWITCH edu-ID. Figure 5.4 shows the frontend with one issued diploma and two received diplomas.

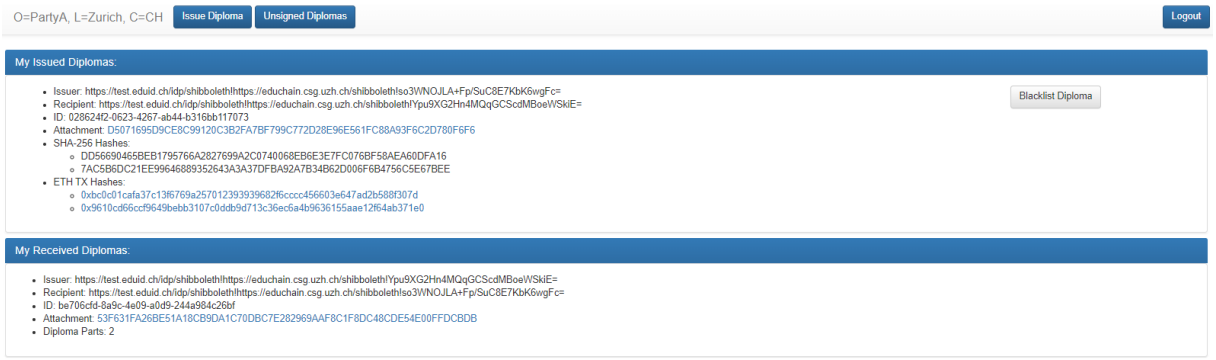


Figure 5.4: Swiss Educhain frontend.

The view of the frontend is dedicated to showing information about the diplomas and the accounts available on the Corda node. The accounts are mainly shown for demonstration and debugging purposes. In a productive environment, only diplomas should be shown. The frontend has the following four main parts:

**My Issued Diplomas**
: Shows a list of all issued diplomas by the logged-in account. This also includes diplomas for which the Ethereum transactions have not yet been signed. Selected information contained in the `DiplomaState` is shown for each entry in the list. Additionally, for every list entry that has been fully signed (i.e. broadcasted on the Ethereum network) there is also a button to blacklist the diploma, which will also exit the corresponding `DiplomaState`.

**My Received Diplomas**
: Shows a list of all received diplomas by the logged-in account. Selected information contained in the `DiplomaState` is shown for each entry in the list.

**Corda Accounts**
: Shows a list of all Corda accounts on the node. Information about each account is provided within the list entry. This is only for demonstration and debugging purposes.

**Educhain Accounts**
: Shows a list of all Educhain accounts created on the node. Information about each account is provided within the list entry. This is only for demonstration and debugging purposes.

Clicking on the 'Issue Diploma' button will open a popup from where the `Issuer` can choose between a simple diploma issuance, extending an already existing diploma, issuing a batch of diplomas or broadcasting a transaction that was signed offline. The 'Unsigned Diplomas' button allows an `Issuer` to download a CSV (Comma-Separated values) file containing all the unsigned transactions that were generated by issuing a diploma air-gapped.

*This ends the text jointly written with Simon Müller [12].*

# Chapter 6

# Evaluation

The goal of this chapter is to evaluate the Swiss Educhain requirements and their fulfillment in terms of functionality, security, privacy and verifiability. A short assessment of the Identity and Access Management (IAM) solution design and implementation is also provided.

## 6.1 Requirements Fulfillment

Based on the produced Proof-of-Concept, the individual requirements are evaluated for both the identity and verification part of Swiss Educhain. The PoC was developed to satisfy as many requirements as possible and most importantly to provide the Minimum Viable Product (MVP) functionality as defined in Section 4.3.3.

This is joint text with Simon Müller [12].

Evaluation of individual requirements:

**RQ01** - Access controls are in place that read the `swissEduIDLinkedAffiliation` attribute and allow issuing certificates if the value is `staff@uzh.ch` or `faculty@uzh.ch`.

**RQ02** - Only the `Issuer` and the `Recipient` of a diploma have access to the data.

**RQ03** - Frontend abstracts all technical complexities included in the issuance process.

**RQ04** - There is a dedicated button for batched diploma issuance.

**RQ05** - The smart contract is on a public blockchain and the verification frontend is accessible by anyone.

**RQ06** - Anyone can verify a diploma without the use of a specific frontend, only by hashing the file and calling the contract.

**RQ07** - Any `Recipient` can download their issued diploma(s).

**RQ08** - Each Educhain user is issued a unique identifier for the Swiss Educhain service from SWITCH edu-ID, which is the `persistentID` attribute.

**RQ09** - This requirement is partially fulfilled because the `Issuers` could disclose the `Recipients'` issued credentials without their approval. Even if the diploma state would be leaked, there is no way to know to whom the diploma was issued. It should be technically enforced that the `Recipients` are the only one that can disclose their credentials.

**RQ10** - Users' accounts are managed from the SWITCH edu-ID identity provider and not from the issuing organizations.

**RQ11** - When a user links their account through an issuing organization's identity provider the issuing organization account can only be created if the identity verification has been completed which includes registering in person. This ensures there is a chain of trust through the linked affiliations.

**RQ12** - Diplomas can be revoked by using the blacklisting functionality for the diploma in question.

**RQ13** - This requirement is partially fulfilled because in Section 4.3.1 a proposed governance process is described. This model has not been reviewed or accepted by all the stakeholders participating in the Swiss Educhain project, it might be updated in the future to meet their needs.

**RQ14** - In Corda DLT all transactions are recorded in the ledger and cannot be individually deleted or tampered with by any party.

**RQ15** - Any `Recipient` can download their issued diploma(s).

**RQ16** - Corda and Ethereum both technically enable multi-sig transactions.

**RQ17** - The system processes data at all levels in a text format.

**RQ18** - This requirement is partially fulfilled because Swiss Educhain accounts are updated automatically each time a user logs in with the latest values of the SWITCH edu-ID account attributes. The nature and the extent of the possible changes to a user's account are dependent on and limited by the account management capabilities offered by SWITCH edu-ID.

**RQ19** - The process for an issuing organization to utilize Swiss Educhain is defined and dependent on the organization's integration with the SWITCH edu-ID identity management system.

**RQ20** - Only users associated with at least one issuing organization are allowed to use the Swiss Educhain service.

**RQ21** - Verification is performed over a public blockchain and is accessible by everyone.

**RQ22** - The user can easily register, access and use the service via a single web interface.

**RQ23** - This requirement is partially fulfilled because it uses open source technologies with widely available documentation, but improvements can be implemented to make the building and deployment process simpler.

**RQ24** - The main technologies used by Swiss Educhain (Apache Webserver, Shibboleth, Spring Boot, Java) are freely available, popular, well-established and mature.

**RQ25** - The main technologies used by Swiss Educhain (Apache Webserver, Shibboleth, Spring Boot, Java) can be operated in most Unix distributions or Windows versions.

**RQ26** - This requirement is partially fulfilled because the application can be easily integrated and is cross-platform compatible as it is built on Java. But, the identity part of the Swiss Educhain service needs to be manually configured and it requires a Service Provider onboarding to the SWITCH edu-ID registry.

**RQ27** - The Swiss Educhain service does not require any specific state-of-the-art technology to be operational.

**RQ28** - This requirement is fulfilled, but as already mentioned the organizations need to onboard to SWITCH edu-ID.

**RQ29** - The high-level access control is defined through Shibboleth and provides access to the service only to either `Issuers` or `Recipients` based on their accounts' disclosed attributes from SWITCH edu-ID.

**RQ30** - Both Corda and Spring Boot can be easily extended with existing functionality.

**RQ31** - Corda does not broadcast by default. Data are disclosed only on a need-to-know basis.

**RQ32** - All transactions are signed, and the public key of the signers is known. Further steps are needed to map the public key to the Educhain account.

This ends the text jointly written with Simon Müller [12].

As seen in the list above, only five out of thirty-two requirements were not completely fulfilled and only partially satisfied. The non-fulfillment of RQ18 and RQ26 has been accepted as a technical limitation deriving from the choice of SWITCH edu-ID as the identity provider for Swiss Educhain. Future work to fulfill requirements RQ9, RQ13 and RQ23 is proposed in Section 7.2.

## 6.2 MVP Evaluation

The MVP functionality defined in Section 4.3.3 was fully implemented in the PoC implementation. An evaluation of the identity management relevant parts is provided, the data structures part is omitted as it is evaluated thoroughly in [12]. Functionality implemented and its relation to the MVP requirements:

**Identity Management**

- **Two types of identities should be supported, Issuers and Recipients.**
  The identities are supported and the role distinction is based on the `isAllowedToIssue` field in the `EduChainAccountState`.
- **Define process of creating a new account.**
  A new SWITCH edu-ID can be easily created by a user in `(test).eduid.ch`.
- **Define data structures for the Educhain account data.**
  The Educhain account data are represented as a Corda state in `EduChainAccountState` and is logically mapped to a Corda account as shown in Figure 4.9.

- **Define access control rules for general access to the service.**
  Service-level access control has been defined using attributes `swissEduIDLinkedAffiliation` and `matriculationNumber`. It is enforced by Apache using the configuration shown in Listing 5.4.
- **Define application level access control for Issuers.**
  By default, all users are `Recipients`, elevated access rights are provided to `Issuers` by checking the `swissEduIDLinkedAffiliation` attribute for the values `staff@uzh.ch` or `faculty@uzh.ch` inside the CorDapp flow execution. The `isAllowedToIssue` value is updated accordingly in the `EduChainAccountState` state.
- **Fetch Student details to create Corda identities.**
  The necessary attributes `commonName`, `mail`, `matriculationNumber`, `persistentID` and `swissEduIDLinkedAffiliation` are retrieved from SWITCH edu-ID.
- **Detect student detail changes and update Educhain account automatically.**
  Detail changes are automatically detected and updated, if needed, by the `load-account` endpoint which is called every time a new session is initiated, or the webpage is refreshed.

**Web Interface**

- **Issue diploma by uploading JSON.**
  Feature has been implemented and can be executed through the `Issue` button.
- **View received diplomas (all users).**
  Received diplomas can be viewed by all users in the `My Received Diplomas` part of the frontend as shown in Figure 5.4.
- **View issued diplomas (only Issuers).**
  Issued diplomas can be viewed only by `Issuers` in the `My Issued Diplomas` part of the frontend as shown in Figure 5.4.
- **Issuer should be able to perform all actions from the frontend.**
  All actions can be executed from the web interface through the single webpage and the provided buttons.
- **Provide a simple login and logout interface.**
  Login is provided by `test.eduid.ch` where the user is redirected automatically after visiting `https://educhain.csg.uzh.ch/app/` and logout is provided by the `Logout` button on the top right corner of the web interface.

**Operations**

- **Define build, installation and deployment process.**
  The process was defined, and the guidelines are provided in Appendix A.
- **Encryption for data in transit and data at rest.**
  Encryption is used end-to-end in all the system components (as shown in Figure 4.2) for data in transit or at rest.
- **Cross-platform compatibility.**
  Cross-platform compatibility is achieved partially for all the components except integration with SWITCH edu-ID which is an accepted limitation.

## 6.3 IAM Evaluation

The chosen IAM solution is for Swiss Educhain to participate as a Service Provider (SP) in the SWITCH identity federation as explained in detail in Section 4.4.2. Users do not assume any role explicitly, they use the attributes of their account to gain the access required to assume the conceptual role (`Issuer` or `Recipient`). A pure ABAC (Attribute Based Access Control) model is followed with two levels of authorization service-level and application-level which determine who should access the service and once authorized to access what actions are allowed to be performed respectively as defined in Section 4.4.7.
The integration with SWITCH and the implementation of application-level accounts was a complex process that required well-defined scenarios, clear requirements and a solution architecture consisting of tightly coupled components. Long term benefits, security, privacy and verifiability were prioritized over faster solutions such as creating a custom application level IdP or using a third-party service to integrate with public IdPs. Significant advantages of the participation in the SWITCH federation and advanced features that can be utilized are listed in Section 4.4.1.
Apart from the advantages certain limitations need to be taken into consideration. Swiss Educhain can only consume data that is available, can be produced in the form of attributes from the Attribute Providers and are supported by SWITCH edu-ID. To create a new value for existing attributes (e.g. `issuer` for `swissEduIDLinkedAffiliation`) the service must inform the Organization and the edu-ID to adapt for such a change. There is a strong dependency on the SWITCH federation for new feature(s) implementation, for the quality of provided services and interfederation interoperability adoption. An important drawback of the Swiss Educhain implementation is the difficulty to add new roles and adapt if the basic use case scenario changes in the future. While the presented solution fulfills entirely the requirements and implements all the necessary functionality for the MVP, adding a new role requires several adaptations in the Resource Registry, the Shibboleth and Apache configuration as well as in the Spring Boot and CorDapp source code.

# Chapter 7

# Conclusion & Future Work

## 7.1 Conclusion

The work conducted in this thesis and the produced outcome was done in close collaboration with Simon Müller [12]. The Swiss Educhain service built upon the foundational analysis conducted in [44]. A greenfield approach was taken to define the stakeholders, strict functional and non-functional requirements, as well as research to design and implement the Swiss Educhain service from scratch. While the foundational parts of the Swiss Educhain were researched, designed and implemented together with [12], this thesis focused on the identity and access management (IAM) part and Simon Müller's thesis on the diploma issuance and verification process [12].
The work conducted to create the Swiss Educhain service was of an exploratory nature. As such, the goals were initially defined only on a high-level with the first target being to define the requirements, research possible technologies or approaches, and most importantly, assess the feasibility of such a use case implementation. The requirements, stakeholders, governance and MVP (Minimum Viable Product) functionality were all derived and refined through consequent iterations of conceptual testing against basic and corner use cases.
With the requirements well-defined the search for appropriate technical designs and solutions followed. A significant challenge was that any potential design needed to not only satisfy the individual requirements but also allow for tight integration between the IAM and verification functionality of Swiss Educhain. Two major decision points were choosing Corda as the private permissioned blockchain and SWITCH edu-ID as the Identity Provider.
The resulting PoC (Proof-of-Concept) implementation satisfied all the MVP functionality requirements providing a robust set of features. Only a few requirements were not completely fulfilled mainly due to two reasons, either lack of time to further advance development, or due to the dependence of the Swiss Educhain service to external entities such as the University of Zurich and SWITCH edu-ID. In the next Section, the identified future work relative to the IAM part of the Swiss Educhain service is discussed.

## 7.2 Future Work

In the scope of this thesis and the implementation of the PoC, the main goals can be considered as completed. A few of the requirements were only partially or not fulfilled as analyzed in Section 6.1. Future work for requirements not completely fulfilled:

**RQ9: Recipients are the only ones that can disclose issued credentials.**
Could be implemented in the future, but a feasibility study needs to be conducted first, to assess if enforcement is possible from a technical standpoint.

**RQ13: The governance model of the Swiss Educhain system must be defined.**
A candidate model has been proposed in Chapter 4 but it needs to be reviewed, updated if needed, and approved by the stakeholders.

**RQ23: Easy to install, configure, deploy, operate, monitor and maintain from an System Administrator's perspective.**
Appendix A provides simple guidelines around installation, configuration and deployment. Monitoring and maintenance were not examined as part of the PoC, best practices could be easier identified after Swiss Educhain is released and tested by users. Room for improvement exists in the installation, configuration and deployment process, automation of the various steps could be beneficial.

During the system design and implementation phases different ideas on how the service could be improved came across. Due to time limitations and prioritization of implementing the MVP functionality they were not examined in depth or not at all. Valuable future work for Swiss Educhain in the field of IAM includes:

**UZH onboarding to SWITCH edu-ID.**
It is of essential importance that UZH is onboarded to SWITCH edu-ID, as this will unlock further possibilities, the features listed in Section 4.4.1 and the high-level benefits for organizations described in [50].

**Creation of `issuer` value for the affiliation attribute.**
Because the `issuer` value is not available, as `Issuers` are identified all members of an organization with the `staff` or `faculty` affiliation. `Issuer's` access rights should be explicitly appointed to an individual with a pre-defined expiration date.

**Four eye principle implementation.**
The system as implemented in the PoC allows all `Issuers` to issue one or more diplomas without any check. The four-eye principle could be used to require an approval before a diploma is issued. This would minimize human errors and prevent malicious behavior from an Issuer.

**Diploma issuance on behalf of a specific organization.**
Further functionality must be implemented to restrict `Issuers` to be able to issue diplomas only on behalf of a certain organization. Identifying the organization can be done from the `swissEduIDLinkedAffiliation` attribute value which is of the form `<affiliation>@<organization>` (e.g. `staff@uzh.ch`).

**Improved audit trail.**
All actions performed in the Swiss Educhain service are traceable through the Corda distributed ledger. The identities of the users are represented by their public keys (new ones generated for each transaction). An automated process which creates a human-readable audit trail log should be implemented.

**Attribute quality.**
SWITCH edu-ID offers an assurance level for each attribute via the `swissEduIDAssuranceLevel` attribute. The option to require a minimum level of assurance for attributes used by Swiss Educhain should be also explored.

**MFA enforcement.**
Multi-factor authentication (MFA) can be enforced by SWITCH edu-ID if requested from the Service Provider, this feature should be utilized to increase security.

# Bibliography


[1] Jerinas Gresch et al. "The Proposal of a Blockchain-Based Architecture for Transparent Certificate Handling". In: *Business Information Systems Workshops*. Ed. by Witold Abramowicz and Adrian Paschke. Cham: Springer International Publishing, 2019, pp. 185–196. ISBN: 978-3-030-04849-5.

[2] Christian Killer. "EduChain - Proposal of Requirements and Architecture". In: Zürich, Switzerland, June 2019.

[3] Nicholas Mwaniki Musee. "An academic certification verification system based on cloud computing environment". Thesis. 2015. URL: http://erepository.uonbi.ac.ke/handle/11295/90179 (visited on 02/24/2020).

[4] *Diploma Mills 9 Strategies for Tackling One of Higher Education's Most Wicked Problems*. WENR. Dec. 12, 2017. URL: https://wenr.wes.org/2017/12/diploma-mills-9-strategies-for-tackling-one-of-higher-educations-most-wicked-problems (visited on 02/24/2020).

[5] *Digital diplomas | MIT Registrar*. URL: https://registrar.mit.edu/transcripts-records/digital-diplomas (visited on 02/24/2020).

[6] *Blockchain Certificates*. Institute For the Future. URL: https://www.unic.ac.cy/iff/blockchain-certificates/ (visited on 02/24/2020).

[7] *Blockcerts Wallet - Apps on Google Play*. URL: https://play.google.com/store/apps/details?id=com.learningmachine.android.app&hl=en (visited on 02/24/2020).

[8] Amy Castor. *Cardano Blockchain's First Use Case: Proof of University Diplomas...* Bitcoin Magazine. Jan. 2, 2018. URL: https://bitcoinmagazine.com/articles/cardano-blockchains-first-use-case-proof-university-diplomas-greece (visited on 02/24/2020).

[9] *Home Page | Trust :: Data*. URL: https://trust.mit.edu/ (visited on 02/24/2020).

[10] *Corda | Open Source Blockchain Platform for Business*. Corda. URL: https://www.corda.net/ (visited on 02/08/2020).

[11] *Hyperledger Wiki*. URL: https://wiki.hyperledger.org/ (visited on 02/24/2020).

[12] Müller Simon. *Design and Implementation of a Data-Agnostic Structure for Blockchain Proof-of-Existence*. Mar. 2, 2020.

[13] Satoshi Nakamoto. "Bitcoin: A Peer-to-Peer Electronic Cash System". In: (), p. 9.

[14] Stuart Haber and W. Scott Stornetta. "How to Time-stamp a Digital Document". In: *J. Cryptol.* 3.2 (Jan. 1991), pp. 99–111. ISSN: 0933-2790. DOI: 10.1007/BF00196791. URL: http://dx.doi.org/10.1007/BF00196791 (visited on 10/07/2019).

[15] *20 Blockchain Use Cases for 2018 You Should Know*. URL: https://hackernoon.com/20-blockchain-use-cases-for-2018-you-should-know-f7d2919c191d (visited on 10/08/2019).

[16] *Top Blockchain Platforms to watch out in 2019*. URL: https://hackernoon.com/top-blockchain-platforms-to-watch-out-in-2019-aa80e336a426 (visited on 10/08/2019).

[17] Bruno Rodrigues et al. "A Technology-driven Overview on Blockchain-based Academic Certificate Handling". In: *Blockchain Technology Applications in Education*. Ed. by Ramesh Sharma, Hakan Yildirim, and Gulsun Meric. Pensilvania, U.S.A: IGI Global, Jan. 2020, pp. 1–290. ISBN: 978-1-522-59478-9. DOI: 10.4018/978-1-5225-9478-9. URL: https://www.igi-global.com/book/blockchain-technology-applications-education/221313.

[18] *Identity - Sociology, Oxford Bibliographies*. URL: https://www.oxfordbibliographies.com/view/document/obo-9780199756384/obo-9780199756384-0025.xml (visited on 02/04/2020).

[19] *Identity - APA Dictionary of Psychology*. URL: https://dictionary.apa.org/identity (visited on 02/04/2020).

[20] Paul A Grassi, Michael E Garcia, and James L Fenton. *Digital identity guidelines: revision 3*. NIST SP 800-63-3. Gaithersburg, MD: National Institute of Standards and Technology, June 22, 2017, NIST SP 800–63–3. DOI: 10.6028/NIST.SP.800-63-3. URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63-3.pdf (visited on 02/04/2020).

[21] *What is a Digital Identity? - Definition from Techopedia*. Techopedia.com. URL: https://www.techopedia.com/definition/23915/digital-identity (visited on 02/04/2020).

[22] Lawrence Frederick Kohl. *The Politics of Individualism: Parties and the American Character in the Jacksonian Era*. Google-Books-ID: sY2iEA6piHcC. Oxford University Press, Feb. 7, 1991. 279 pp. ISBN: 978-0-19-536183-4.

[23] *The Path to Self-Sovereign Identity*. URL: http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html (visited on 02/05/2020).

[24] *Home - Sovrin*. URL: https://sovrin.org/ (visited on 02/05/2020).

[25] *uPort - Tools for Decentralized Identity and Trusted Data*. URL: https://www.uport.me/ (visited on 02/05/2020).

[26] *ShoCard: The Premier Blockchain-Based Mobile Identity Platform*. URL: https://shocard.com/ (visited on 02/05/2020).

[27] Paul Dunphy and Fabien A.P. Petitcolas. "A First Look at Identity Management Schemes on the Blockchain". In: *IEEE Security & Privacy* 16.4 (July 2018), pp. 20–29. ISSN: 1540-7993, 1558-4046. DOI: 10.1109/MSP.2018.3111247. URL: https://ieeexplore.ieee.org/document/8425607/ (visited on 02/05/2020).

[28] *Identity And Access Management (iam)*. Gartner. URL: https://www.gartner.com/en/information-technology/glossary/identity-and-access-management-iam (visited on 02/08/2020).

[29] *What is Access Control? - Definition from Techopedia*. Techopedia.com. URL: https://www.techopedia.com/definition/5831/access-control (visited on 01/29/2020).

[30] James A. Martin. *What is access control? A key component of data security*. CSO Online. Aug. 21, 2019. URL: https://www.csoonline.com/article/3251714/

what-is-access-control-a-key-component-of-data-security.html (visited on 02/08/2020).

[31] *What is access control? - Definition from WhatIs.com*. SearchSecurity. URL: https://searchsecurity.techtarget.com/definition/access-control (visited on 02/08/2020).

[32] *identification - Glossary | CSRC*. URL: https://csrc.nist.gov/glossary/term/identification (visited on 02/08/2020).

[33] *Digital Authentication - the basics*. URL: https://www.cryptomathic.com/news-events/blog/digital-authentication-the-basics (visited on 02/05/2020).

[34] *Payment Services Directive (PSD2): Regulatory Technical Standards (RTS) enabling consumers to benefit from safer and more innovative electro*. European Commission - European Commission. URL: https://ec.europa.eu/commission/presscorner/detail/en/MEMO_17_4961 (visited on 02/08/2020).

[35] *EBA publishes an Opinion on the elements of strong customer authentication under PSD2*. European Banking Authority. June 21, 2019. URL: https://eba.europa.eu/eba-publishes-an-opinion-on-the-elements-of-strong-customer-authentication-under-psd2 (visited on 02/08/2020).

[36] Barbara Y. Fraser. *Site Security Handbook*. URL: https://tools.ietf.org/html/rfc2196#section-4.4 (visited on 02/08/2020).

[37] *IBM Knowledge Center - Authorization Rules*. URL: https://www.ibm.com/support/knowledgecenter/SSPREK_9.0.0/com.ibm.isam.doc/base_admin/concept/con_tamauthorule.html (visited on 02/08/2020).

[38] Vincent C. Hu et al. *Guide to Attribute Based Access Control (ABAC) Definition and Considerations*. NIST SP 800-162. National Institute of Standards and Technology, Jan. 2014, NIST SP 800–162. DOI: 10.6028/NIST.SP.800-162. URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-162.pdf (visited on 02/08/2020).

[39] *role-based access control (RBAC) - Glossary | CSRC*. URL: https://csrc.nist.gov/glossary/term/role_based-access-control (visited on 02/08/2020).

[40] Joint Task Force Transformation Initiative. *Security and Privacy Controls for Federal Information Systems and Organizations*. NIST SP 800-53r4. National Institute of Standards and Technology, Apr. 2013, NIST SP 800–53r4. DOI: 10.6028/NIST.SP.800-53r4. URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf (visited on 02/08/2020).

[41] *Shibboleth Consortium Privacy Preserving Identity Management*. URL: https://www.shibboleth.net/ (visited on 02/02/2020).

[42] *SWITCH edu-ID - SWITCH*. URL: https://www.switch.ch/edu-id/ (visited on 02/01/2020).

[43] *Corda Accounts Library*. URL: https://github.com/corda/accounts (visited on 10/17/2019).

[44] Jerinas Gresch. "An Educational Blockchain for the University of Zurich (UZHBC)". MA thesis. Zurich, Switzerland: Universität Zürich, Aug. 2018. URL: https://files.ifi.uzh.ch/CSG/staff/rodrigues/extern/theses/ma-jerinas.pdf.

[45] *Foundation - About us - SWITCH*. URL: https://www.switch.ch/about/foundation/ (visited on 02/09/2020).

[46] *Authentication - Services - SWITCH*. URL: https://www.switch.ch/services/aai/ (visited on 02/09/2020).

[47] Christoph Graf. *The SWITCH identity federation - a look beyond its borders*. SWITCH Identity Blog. Apr. 24, 2018. URL: https://identityblog.switch.ch/2018/04/24/the-switch-identity-federation-a-look-beyond-its-borders/ (visited on 02/14/2020).

[48] *Service Description - SWITCH edu-ID - SWITCH*. URL: https://www.switch.ch/edu-id/about/terms/ (visited on 02/09/2020).

[49] *SWITCH edu-ID - SWITCH Benefits*. URL: https://www.switch.ch/edu-id/#benefits (visited on 02/09/2020).

[50] *For organisations - SWITCH edu-ID - SWITCH*. URL: https://www.switch.ch/edu-id/organisations/ (visited on 02/09/2020).

[51] *For services - SWITCH edu-ID - SWITCH*. URL: https://www.switch.ch/edu-id/services/ (visited on 02/09/2020).

[52] *For users - SWITCH edu-ID - SWITCH*. URL: https://www.switch.ch/edu-id/users/ (visited on 02/09/2020).

[53] *Architecture - For organisations - SWITCH edu-ID - SWITCH*. URL: https://www.switch.ch/edu-id/organisations/architecture/ (visited on 02/09/2020).

[54] *Swiss edu-ID vs. SWITCHaai - About - Swiss edu-ID - SWITCH Projects*. URL: https://projects.switch.ch/eduid/about/swiss-edu-id-vs.-switchaai/ (visited on 02/09/2020).

[55] *SAML and Shibboleth*. URL: http://sites.utoronto.ca/security/projects/shibboleth.htm (visited on 02/09/2020).

[56] *corda/corda*. original-date: 2016-10-06T08:46:29Z. Feb. 8, 2020. URL: https://github.com/corda/corda (visited on 02/09/2020).

[57] *R3 | Blockchain & DLT Software Development Company*. R3. URL: https://www.r3.com/ (visited on 02/09/2020).

[58] *Advance Corda by Contributing Code*. Corda. URL: https://www.corda.net/contribute-code/ (visited on 02/09/2020).

[59] *Connect with Our Blockchain Community*. Corda. URL: https://www.corda.net/community/ (visited on 02/09/2020).

[60] *Nodes – R3 Corda Master documentation*. URL: https://docs.corda.net/key-concepts-node.html (visited on 02/11/2020).

[61] *Key concepts – R3 Corda Master documentation*. URL: https://docs.corda.net/key-concepts.html (visited on 02/11/2020).

[62] *Deterministic JVM – R3 Corda Master documentation*. URL: https://docs.corda.net/key-concepts-djvm.html (visited on 02/11/2020).

[63] *The network – R3 Corda Master documentation*. URL: https://docs.corda.net/key-concepts-ecosystem.html (visited on 02/11/2020).

[64] *The ledger – R3 Corda Master documentation*. URL: https://docs.corda.net/key-concepts-ledger.html (visited on 02/11/2020).

[65] *Contracts – R3 Corda Master documentation*. URL: https://docs.corda.net/key-concepts-contracts.html (visited on 02/11/2020).

[66] *Unlocking New Opportunities with Accounts on Corda*. Corda. Nov. 18, 2019. URL: https://www.corda.net/blog/unlocking-new-opportunities-with-accounts-on-corda-2/ (visited on 02/11/2020).

[67] *Shibboleth - Users - Shibboleth 3 attributes not exposed from Apache 2.4 as environment variables | Threaded View*. URL: https://shibboleth.1660669.n2.

nabble.com/Shibboleth-3-attributes-not-exposed-from-Apache-2-4-as-environment-variables-tt7644981.html (visited on 02/15/2020).

[68] *kotlin - Corda Service Failed to Instantiate*. Stack Overflow. URL: https://stackoverflow.com/questions/59433434/corda-service-failed-to-instantiate (visited on 02/15/2020).

[69] *corda - How to Query if an Account exists by name*. Stack Overflow. URL: https://stackoverflow.com/questions/59431400/how-to-query-if-an-account-exists-by-name (visited on 02/15/2020).

[70] *UZH Blockchain Centre*. URL: http://blockchain.uzh.ch/ (visited on 10/12/2019).

[71] *R3 Marketplace | Identity Solutions*. URL: https://marketplace.r3.com (visited on 02/13/2020).

[72] *The Beer Drinker's Guide to SAML*. Duo Security. URL: https://duo.com/blog/the-beer-drinkers-guide-to-saml (visited on 02/14/2020).

[73] Lukas Haemmerle. *Secrets of the edu-ID passwords*. SWITCH Identity Blog. Aug. 13, 2019. URL: https://identityblog.switch.ch/2019/08/13/secrets-of-edu-id-passwords/ (visited on 02/14/2020).

[74] *Two-Step Login - For services - SWITCH edu-ID - SWITCH*. URL: https://www.switch.ch/edu-id/services/two-step-login/ (visited on 02/14/2020).

[75] *Attribute Quality - Attribute Model - For services - SWITCH edu-ID - SWITCH*. URL: https://www.switch.ch/de/edu-id/services/attributes/quality-levels/ (visited on 02/14/2020).

[76] *Extended Model - Attribute Model - For services - SWITCH edu-ID - SWITCH*. URL: https://www.switch.ch/de/edu-id/services/attributes/extended-model/ (visited on 02/14/2020).

[77] Rolf Brugger. *Technical Accounts*. SWITCH Identity Blog. Feb. 1, 2019. URL: https://identityblog.switch.ch/2019/02/01/technical-accounts/ (visited on 02/14/2020).

[78] *Link Composer - For services - SWITCH edu-ID - SWITCH*. URL: https://www.switch.ch/de/edu-id/services/link-composer/ (visited on 02/14/2020).

[79] *Attribute Completion Flow - Link Composer - For services - SWITCH edu-ID - SWITCH*. URL: https://www.switch.ch/de/edu-id/services/link-composer/attribute-completion-flow/ (visited on 02/14/2020).

[80] *Login flows - Link Composer - For services - SWITCH edu-ID - SWITCH*. URL: https://www.switch.ch/de/edu-id/services/link-composer/login-flows/ (visited on 02/14/2020).

[81] *Testing - APIs and technical guides - For organisations - SWITCH edu-ID - SWITCH*. URL: https://www.switch.ch/edu-id/organisations/tech/testing/ (visited on 02/14/2020).

[82] *SWITCH - AAI Resource Registry - Home*. URL: https://rr.aai.switch.ch/menu.php (visited on 02/15/2020).

[83] *The Swiss edu-ID - The persistent Swiss academic digital identity*. URL: https://projects.switch.ch/export/sites/projects/eduid/.galleries/documents/1_20140813_Swiss_edu-ID_V2.pdf (visited on 02/15/2020).

[84] *Resource Registry - Tools - Support - SWITCHaai - SWITCH*. URL: https://www.switch.ch/aai/support/tools/resource-registry/ (visited on 02/15/2020).

[85] *Federation - About - SWITCHaai - SWITCH*. URL: https://www.switch.ch/aai/about/federation/ (visited on 02/15/2020).

[86] *Adoption - For organisations - SWITCH edu-ID - SWITCH*. URL: https://www.switch.ch/edu-id/organisations/adoption/ (visited on 02/15/2020).

[87] *Attribute Model - For services - SWITCH edu-ID - SWITCH*. URL: https://www.switch.ch/edu-id/services/attributes/ (visited on 02/15/2020).

[88] *Classic Model - Attribute Model - For services - SWITCH edu-ID - SWITCH*. URL: https://www.switch.ch/edu-id/services/attributes/classic-model/ (visited on 02/15/2020).

[89] Rolf Brugger. *Services and Swiss edu-ID*. June 29, 2017. URL: https://projects.switch.ch/export/sites/projects/eduid/.galleries/documents/1_20140813_Swiss_edu-ID_V2.pdf (visited on 02/15/2020).

[90] *Using Persistent ID as a User Attribute - Service Provider - Guides - SWITCHaai - SWITCH*. URL: https://www.switch.ch/aai/guides/sp/persistentid/ (visited on 02/16/2020).

[91] *How Shibboleth Logins Work | NC State Shibboleth*. URL: https://docs.shib.ncsu.edu/docs/shibworks.html (visited on 02/16/2020).

[92] *Service Provider - Guides - SWITCHaai - SWITCH*. URL: https://www.switch.ch/aai/guides/sp/ (visited on 02/22/2020).

[93] *Shibboleth SP Installation - Service Provider - Guides - SWITCHaai - SWITCH*. URL: https://www.switch.ch/aai/guides/sp/installation/?os=ubuntu (visited on 11/10/2019).

[94] *Shibboleth SP Configuration - Service Provider - Guides - SWITCHaai - SWITCH*. URL: https://www.switch.ch/aai/guides/sp/configuration/ (visited on 11/10/2019).

[95] *Certbot - Ubuntubionic Apache*. URL: https://certbot.eff.org/lets-encrypt/ubuntubionic-apache (visited on 02/22/2020).

[96] *Let's Encrypt - Free SSL/TLS Certificates*. URL: https://letsencrypt.org/ (visited on 02/22/2020).

[97] *KeyChest - Let's Encrypt certificate into Java JKS*. URL: https://keychest.net/stories/lets-encrypt-certificate-into-java-jks (visited on 02/20/2020).

[98] *SP Access Rules - Service Provider - Guides - SWITCHaai - SWITCH*. URL: https://www.switch.ch/aai/guides/sp/access-rules/ (visited on 02/25/2020).

[99] *corda/cordapp-template-kotlin*. original-date: 2017-07-05T15:59:56Z. Feb. 4, 2020. URL: https://github.com/corda/cordapp-template-kotlin (visited on 02/20/2020).

[100] Dan Newton. *Corda Services 101*. Aug. 19, 2018. URL: https://lankydan.dev/2018/08/19/corda-services-101 (visited on 02/23/2020).

[101] *AngularJS - Superheroic JavaScript MVW Framework*. URL: https://angularjs.org/ (visited on 02/22/2020).

[102] *Bootstrap*. URL: https://getbootstrap.com/ (visited on 02/22/2020).

# Abbreviations

| | |
|---|---|
| 2FA | Two-Factor Authentication |
| AAI | Authentication and Authorization Infrastructure |
| ABAC | Attribute Based Access Control |
| AJP | Apache JServ Protocol |
| API | Application Programming Interface |
| CA | Certificate Authority |
| CorDapp | Corda Decentralized Application |
| CRUD | Create Read Update Delete |
| CSG | Communications Systems Research Group |
| CSV | Comma-Separated values |
| CV | Curriculum Vitae |
| DJVM | Deterministic Java Virtual Machine |
| DL | Distributed Ledger |
| DLT | Distributed Ledger Technology |
| EBA | European Banking Authority |
| FOSS | Free and Open Source Software |
| GDPR | General Data Protection Regulation |
| GRNET | Greek Research and Technology Network |
| GUID | Globally Unique Identifier |
| IAM | Identity and Access Management |
| IFI | Department of Informatics |
| JAR | Java ARchive |
| JSON | JavaScript Object Notation |
| KYC | Know Your Customer |
| MFA | Multi Factor Authentication |
| MIT | Massachusetts Institute of Technology |
| MVP | Minimum Viable Product |
| NTP | Network Time Protocol |
| NIST | National Institute of Standards and Technology |
| OSS | Open Source Software |
| OTP | One Time Password |
| PDF | Portable Document Format |
| PIN | Personal Identification Number |
| PDF | Portable Document Format |
| PKCS12 | Public-Key Cryptography Standards 12 |
| PoC | Proof-of-Concept |

| | |
|---|---|
| PoE | Proof-of-Existence |
| RBAC | Role Based Access Control |
| REST | Representational State Transfer |
| RR | Resource Registry |
| RQ | Requirement |
| RPC | Remote Procedure Call |
| SAML | Security Assertion Markup Language |
| SDK | Software Development Kit |
| SHA | Secure Hash Algorithm |
| SSI | Self-Sovereign Identity |
| SSL | Secure Sockets Layer |
| SSO | Single Sign On |
| SP | Service Provider |
| TLS | Transport Layer Security |
| TOTP | Time-based one time password |
| UI | User Interface |
| UUID | Universally Unique Identifier |
| UNIC | University of Nicosia |
| UX | User Experience |
| UZH | University of Zurich |
| WAYF | Where Are You From |

# List of Figures

# List of Tables

# Appendix A

# Installation and Configuration Guidelines

*This is joint text with Simon Müller [12].*

## A.1 System Requirements

To facilitate the development of the Swiss Educhain service a server was set up in the internal CSG (Communications Systems Group) infrastructure of the department of Informatics. Swiss Educhain has several dependencies and system requirements, which are explained below:

**OS Requirements**
A medium-sized Ubuntu server (implementation tested on 4 CPUs 2.4 GHz with 4 Gb of memory), which is capable of running multiple Java instances.

**Software Dependencies**

Zulu Java OpenJDK 8 (tested on version 1.8.0_232)
Apache HTTP Server (tested on version 2.4.29)
Spring Boot (tested on version 2.0.2.RELEASE)
Shibboleth (tested on version 3.0.4)
Valid SSL certificate (tested with LetsEncrypt)

## A.2 Deployment

The code of Swiss Educhain is based on the Kotlin CorDapp example template provided in [99]. To build the Swiss Educhain software components, the Gradle build agent is used. Before deployment a PKCS12 (Public-Key Cryptography Standards - 12) certificate needs

to be generated based on the valid SSL certificate, instructions on how to create it are given in [97]. The instructions on how to build the Swiss Educhain code are as follows:

- Corda Nodes:
  1. Fill the configuration file `additional.conf` with the correct information. The file is found in the root directory.
  2. Execute in root directory: `./gradlew deployNodes`
  3. JAR, configuration and database files will be created inside `/build/nodes` folder.
- Frontend:
  1. Fill the configuration file `application.properties` with the correct information. The file is found in the `clients/src/main/resources` directory.
  2. Execute in root directory: `./gradlew runTemplateServer`
  3. The `clients-VERSION.jar` file will be created inside the `/clients/build/libs` folder.

After successfully building the code, the service can be deployed by following these steps:

- Copy the produced `clients-VERSION` JAR file to the server.
- Copy the `nodes` folder containing the two `corda.jar` files for `PartyA` and `Notary` to the server together with the configuration files produced (`node.conf`, `persistence.mv` etc.)
- Execute command `nohup java -jar corda.jar &` from the same directory where it was copied to, inside the `Notary` folder.
- Execute command `nohup java -jar corda.jar &` from the same directory where it was copied to, inside the `PartyA` folder.
- Execute command `nohup java -jar clients-VERSION.jar &` from the same directory where it was copied to. The frontend will exit automatically if it doesn't detect any running Corda nodes.

*This ends the text jointly written with Simon Müller [12].*

# Appendix B

# Code Repository Structure

*This is joint text with Simon Müller [12].*

The following directory tree represents a simplified structure of the Swiss Educhain code that is included with the CD. For the sake of readability, folders and files that are not directly relevant have been left out.

```
educhain-code
├── clients
│   ├── src/main
│   │   ├── kotlin/ch/educhain
│   │   │   ├── bean
│   │   │   └── webserver ..... Folder contains Kotlin code for frontend.
│   │   └── resources
│   │       ├── static
│   │       │   ├── app.js ..... Frontend JavaScript code.
│   │       │   └── index.html ..... Frontend HTML code.
│   │       └── application.properties ..... Properties file for frontend.
│   └── resources
├── contracts
│   └── src/main/kotlin/ch/educhain
│       ├── contracts ..... Folder contains the Swiss Educhain contract code.
│       └── states ..... Folder contains the Swiss Educhain states.
├── verification_frontend ... Folder contains the code for the verification frontend.
├── workflows
│   └── src/main
│       ├── java/ch/educhain/solidity ..... Contains the contract wrapper code.
│       └── kotlin/ch/educhain
│           ├── flows ..... Contains all the Swiss Educhain flows.
│           └── services ..... Contains all the Swiss Educhain services.
├── additional.conf ..... Configuration file for Corda node.
└── educhainSign.sh ..... Shell script for offline signing.
```

*This ends the text jointly written with Simon Müller [12].*

# Appendix C

# Contents of the CD

**abstract.txt** - abstract in English.

**educhain-configuration** - contains Educhain configuration files.

**educhain-code** - contains the Swiss Educhain code.

**intermediate-presentation.pdf** - contains the slides for the intermediate presentation.

**thesis.pdf** - this thesis report as PDF.

**report** - contains LaTeXsources of this thesis, including all figures.

**zusammenfassung.txt** - abstract in German.