

# BlockDiploma – Decentralizing the Norwegian Diploma Registry using Blockchain Technology

Thomas Reite

Master's thesis in Software Engineering

Department of Computer science, Electrical engineering and  
Mathematical sciences, Western Norway University of  
Applied Sciences

Department of Informatics, University of Bergen

Supervisors: Lars-Petter Helland and Svein Ølnes

June 2020



Western Norway  
University of  
Applied Sciences



## Abstract

Academic diplomas are being falsified and potentially resulting in unqualified individuals getting the job, or a better candidate being bypassed by a forger. Secure and reliable verification mechanisms for academic diplomas are needed. Norway has attempted to accomplish this by developing the Diploma registry, a digital solution for sharing academic results.

Our research reviews current diploma systems to identify challenges. Following the review, our research effort shifts focus from identifying challenges to attempting to find solutions using blockchain technology. The research is based on the hypothesis that there are challenges with the present solutions, and that those challenges can be resolved by decentralizing the diploma registry using blockchain and peer-to-peer technology.

The research is classified as computer research using the engineering method. The first step was to gather and aggregate information about current diploma systems and relevant blockchain proposed solutions. Based upon the information gathered we could identify challenges with the current solutions, and we started to formulate requirements for a blockchain-based one. After formulating our proposal in the form of written requirements, we started to explore how the challenges could be resolved using decentralized technology.

Following the exploration of decentralized technologies, we ended up with developing a decentralized application called BlockDiploma. BlockDiploma is built using smart contracts with the Ethereum blockchain, IPFS for decentralized storage and standard web technologies for the user interface. During and after the development we analyzed and evaluated how well it resolved the identified challenges and whether it introduces new challenges. Our conclusion is that there are several issues other than just falsification with the present diploma systems, and that a decentralized diploma registry can in the future be part of the solution to those challenges.

## Acknowledgements

I would like to express my deep gratitude to my supervisors Lars-Petter Helland and Svein Ølnes for their patient guidance, encouragement, and useful critiques of this research work. I sincerely appreciate the extra help at the end with finalizing my thesis. Thank you for taking the time.

I wish to acknowledge the help provided by Maren Skogland Nornes, Carl Fredrik Engholm Holen and Christian Aure Aannø for their assistance with proofreading this thesis. I would like to thank my fellow students at the lab for valuable discussion and distractions throughout greater parts of the work on this thesis.

Finally, I wish to thank my family for their support and encouragement throughout my study.

# Contents

<b>Abstract</b>	<b>i</b>
<b>Acknowledgements</b>	<b>ii</b>
<b>List of Figures</b>	<b>viii</b>
<b>List of Tables</b>	<b>ix</b>
<b>List of Terms</b>	<b>x</b>
<b>List of Abbreviations</b>	<b>xiv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Background . . . . .	2
1.2 Motivation . . . . .	6
1.3 Problem Description . . . . .	6
1.4 Thesis Outline . . . . .	8
<b>2 Research Methodology</b>	<b>10</b>
2.1 The Goal: Gaining insights and understandings through ex- ploration . . . . .	10
2.2 Methodology - Computer Research using the Engineering Method	10
2.2.1 Informational phase . . . . .	10
2.2.2 Propositional phase . . . . .	11
2.2.3 Analytical phase . . . . .	11
2.2.4 Evaluation phase . . . . .	11
2.3 Expected results . . . . .	12
2.4 Validity . . . . .	12
2.5 Literature . . . . .	12
<b>3 Theoretical Background</b>	<b>13</b>
3.1 The Beginning of Blockchain Technology . . . . .	13
3.2 Blockchain Fundamentals . . . . .	14
3.3 Ethereum: The World Computer . . . . .	15
3.3.1 Ethereum Compared to Bitcoin . . . . .	16
3.3.2 Cryptographic Hash Functions . . . . .	17
3.3.3 Keys and Addresses . . . . .	18
3.3.4 Transactions . . . . .	20
3.3.5 Consensus Methods . . . . .	22
3.3.6 Smart Contracts . . . . .	24



3.3.7	Oracles . . . . .	25
3.3.8	Decentralized Applications . . . . .	26
3.3.9	Ethereum Improvement Proposals and ERCs . . . . .	28
3.4	Wallets . . . . .	28
3.5	Permissioned and Unpermissioned Blockchains . . . . .	30
<b>4</b>	<b>Assessment of Current Diploma Solutions</b>	<b>32</b>
<b>5</b>	<b>Related Work</b>	<b>37</b>
5.1	Issuance of Diplomas . . . . .	39
5.1.1	Store the hash of diploma on a blockchain . . . . .	39
5.1.2	Storing the diploma on the blockchain . . . . .	40
5.2	Verification of Diplomas . . . . .	41
5.2.1	Verifying a diploma by hash . . . . .	41
5.2.2	Verify a diploma through cryptography and certificates	42
5.3	Revocations of diplomas . . . . .	42
5.3.1	Centralized revocation . . . . .	42
5.3.2	Re-issue all diplomas . . . . .	42
5.3.3	Extra transaction outputs per diploma . . . . .	43
5.3.4	Revocation is handled by an issuer-hosted CRL . . . . .	43
5.3.5	Use an additional decentralization layer for revocation	43
5.3.6	Blockchain specializing in diploma issuing . . . . .	43
5.3.7	Credentialing meta-protocol . . . . .	44
5.3.8	Revoke using cryptography . . . . .	44
5.4	Summary . . . . .	44
5.5	Assessment of Related Work . . . . .	45
<b>6</b>	<b>BlockDiploma - Design and Implementation</b>	<b>48</b>
6.1	Requirements . . . . .	48
6.1.1	Stakeholders . . . . .	48
6.1.2	Legal requirements . . . . .	49
6.1.3	Non-functional requirements (NFRs) . . . . .	50
6.1.4	Functional Requirements (FRs) . . . . .	52
6.1.5	User Interface Requirements . . . . .	54
6.1.6	Requirements Summary . . . . .	55
6.2	Choice of Technology . . . . .	57
6.2.1	Why Ethereum and not some other blockchain? . . . . .	58
6.2.2	Truffle Suite . . . . .	58
6.2.3	ReactJS . . . . .	59
6.2.4	Why IPFS and not Swarm? . . . . .	60
6.3	Architecture . . . . .	60

6.4	Implementation . . . . .	62
6.4.1	Authentication with Ethereum Address . . . . .	62
6.4.2	Issuing Diplomas . . . . .	63
6.4.3	Revoke a Diploma . . . . .	67
6.4.4	Verify a Diploma . . . . .	68
6.4.5	Student Portal . . . . .	69
6.4.6	University Portal . . . . .	71
6.4.7	Fighting Diploma Mills . . . . .	72
6.4.8	Customizing the solution . . . . .	74
6.4.9	Implementation of user interface requirements . . . . .	74
6.4.10	Implementation of non-functional requirements (NFRs) . . . . .	74
6.4.11	Summary of implemented requirements . . . . .	76
<b>7</b>	<b>Security Assessment</b>	<b>79</b>
7.1	Smart Contract Security Assessment . . . . .	79
7.1.1	Reentrancy . . . . .	79
7.1.2	Access Control . . . . .	80
7.1.3	Arithmetic . . . . .	80
7.1.4	Unchecked Low Level Calls . . . . .	80
7.1.5	Denial of Services . . . . .	81
7.1.6	Bad Randomness . . . . .	81
7.1.7	Front Running . . . . .	81
7.1.8	Time Manipulation . . . . .	82
7.1.9	Short Addresses . . . . .	82
7.1.10	Unknown Unknowns . . . . .	83
7.1.11	Summary . . . . .	83
7.2	Frontend Security Assessment . . . . .	84
7.2.1	Injection . . . . .	84
7.2.2	Broken Authentication . . . . .	85
7.2.3	Sensitive Data Exposure . . . . .	85
7.2.4	XML External Entities (XXE) . . . . .	85
7.2.5	Broken Access Control . . . . .	86
7.2.6	Security Misconfiguration . . . . .	86
7.2.7	Cross-Site Scripting XSS . . . . .	86
7.2.8	Insecure Deserialization . . . . .	87
7.2.9	Using Components with Known Vulnerabilities . . . . .	87
7.2.10	Insufficient Logging and Monitoring . . . . .	87
7.2.11	Summary . . . . .	88
7.3	Storage and communication Security Assessment . . . . .	89
7.4	Other known vulnerabilities and weaknesses . . . . .	89
7.5	Summary . . . . .	90

<b>8</b>	<b>Analysis and Assessment</b>	<b>91</b>
8.1	BlockDiploma – a Decentralized Diploma System . . . . .	91
8.1.1	Stakeholders and requirements . . . . .	91
8.1.2	BlockDiploma and the challenges with the present diploma solutions . . . . .	92
8.1.3	BlockDiploma at a larger scale . . . . .	95
8.2	Assessment of Ethereum for BlockDiploma . . . . .	97
8.3	Results and Learning Outcomes . . . . .	99
8.4	Research . . . . .	102
8.4.1	Own Contribution . . . . .	102
8.4.2	Validity . . . . .	103
8.4.3	The method used . . . . .	104
<b>9</b>	<b>Conclusion</b>	<b>105</b>
<b>10</b>	<b>Further Work</b>	<b>108</b>
<b>A</b>	<b>Appendix A: Smart Contract Implementation</b>	<b>116</b>
A.1	The University Registry Contract . . . . .	116
A.2	The Diploma Registry Interface . . . . .	119
A.3	The Diploma Registry Contract . . . . .	121
A.4	The University Interface . . . . .	123
A.5	The University Contract . . . . .	125
A.6	The ERC-173 Contract . . . . .	128
<b>B</b>	<b>Appendix B: BlockDiploma Screenshots</b>	<b>129</b>
B.1	Public Website . . . . .	129
B.2	University Registry . . . . .	132
B.3	University . . . . .	135
B.4	Student . . . . .	140

## List of Figures

1	Sharing a diploma with an employer the traditional way. . . .	4
2	Sharing a diploma with an employer using blockchain technology.	6
3	How a hash algorithm works. . . . .	17
4	Proof of inclusion using a merkle tree [16, p. 220]. . . . .	18
5	Process of creating an Ethereum address . . . . .	19
6	Process of creating and verifying a digital signature . . . . .	20
7	Simplified illustration of a transaction. . . . .	22
8	Higher Education Provider working as an oracle by providing diplomas . . . . .	26
9	Connecting Metamask to DApp and choosing network. . . . .	29
10	Transactions against a smart contract that changes state. . . .	30
11	Redacted email received from the Diploma registry. . . . .	33
12	Certificate proving that the websites is in fact the Diploma registry. . . . .	34
13	User groups and their use cases. . . . .	55
14	The Ganache GUI used for local development. . . . .	59
15	BlockDiploma interacting with IPFS (storage) and Ethereum (smart contracts) using Infura. . . . .	60
16	BlockDiploma Overview. . . . .	62
17	The currently flawed wallet Authentication and Authorization flow. . . . .	63
18	Process of issuing a diploma. . . . .	64
19	Screenshot of form used to issue diplomas using an existing file.	65
20	Screenshot of form used to issue diplomas manually. . . . .	66
21	Screenshot of diploma generated by BlockDiploma. . . . .	67
22	Screenshot showing the screen used to revoke diplomas. . . . .	68
23	Process of revoking a diploma. . . . .	68
24	Process of verifying a diploma. . . . .	69
25	Student Portal inner workings. . . . .	69
26	Screenshot of Student Portal. . . . .	70
27	How diploma mills are denied access to the diploma registry smart contract. . . . .	73
28	Screenshot of the form used to remove higher education providers from the university registry smart contract. . . . .	73
29	Diplomas sent directly to employer to prevent fraud. . . . .	93
30	Process of determining if a higher education provider is rep- utable. . . . .	96
31	BlockDiploma at a larger scale. . . . .	98
32	The public website for validating diplomas. . . . .	129

33	The public website when a diploma successfully validated. . . .	130
34	The public website when diploma could not be validated. . . .	130
35	Loading screen displayed when no wallet is connected to the DApp. . . . .	131
36	Page displayed when user is not authorized to perform any actions. . . . .	131
37	The University Registry Dashboard. . . . .	132
38	Page for adding a university to the university registry. . . . .	132
39	Page for removing a university from the university registry. . . .	133
40	Page for viewing event logs of actions in the University Registry.	133
41	Page for transferring ownership of the University Registry con- tract. . . . .	134
42	The University Portal Dashboard . . . . .	135
43	Page for issuing a Diploma. . . . .	136
44	Page for revoking a diploma from the diploma registry. . . . .	137
45	Page for viewing event logs of diploma and employee actions. . .	137
46	Page for giving an employee access to the university-contract. .	138
47	Page for revoking access for an employee from the university contract. . . . .	138
48	Page for transferring ownership of the university contract. . . .	139
49	Page for terminating the University contract. . . . .	139
50	The Student Portal . . . . .	140
51	Page showing the download of a diploma from IPFS. . . . .	140
52	Example of a BlockDiploma PDF-generated diploma. . . . .	141
53	Example of generated diploma supplement . . . . .	142
54	Private sharing using asymmetric encryption. . . . .	142
55	Sharing screen with link and QR-code that can be sent to an employer. . . . .	143

## List of Tables

1	Blockchain Types [22] . . . . .	31
2	Overview of related work . . . . .	45
3	Requirement Categories. . . . .	56
4	Non-functional Requirements. . . . .	56
5	UI-requirements. . . . .	56
6	Functional Requirements. . . . .	57
7	Status of Non-functional Requirements. . . . .	77
8	Status of UI-requirements. . . . .	77
9	Status of Functional Requirements. . . . .	78
10	Status of DASP Top 10 Threats Handling. . . . .	83
11	Status of OWASP Top 10 Threats Handling. . . . .	88

## List of Terms

The list of terms contains many of the terms used in this thesis. Many of the definitions are inspired and some are copied from the glossary in the Mastering Ethereum book by Antonopoulos and Wood [1]

**Account** An account can be either an Externally owned account or a contract account. An account is an object consisting of an address, balance and nonce and optionally storage and code.

**Address** Addresses can receive (destination address) or send (source address) transactions on the blockchain. Specifically, an address is the rightmost 160-bits of the Keccak hash of a public key.

**Asymmetric encryption** Encryption where a public-private key pair is used. The message is encrypted using the public key of the recipient and decrypted by the recipient using his private key.

**Block** A block is collection of required information about transactions. Blocks are appended to the blockchain by miners.

**Centralized** The term centralized will often refer to a single entity storing and/or controlling the information.

**Common Student System** Felles studentsystem (FS) can be translated to the common student system. FS is a study administration system developed for higher education providers. FS is developed by The Norwegian Directorate for ICT and Joint Services in Higher Education and Research (UNIT).

**Consensus** When most of the nodes in the network has the same blocks in their local validated blockchain, in other words the nodes have agreed upon which blocks are valid.

**Contract Account** Contract account is an account that executes its code when it receives a transaction (invocation) from another account. Contract account cannot initiate transactions on their own.

**Decentralized** The term decentralized will often refer to using peer-to-peer technology to control and store information. In other words, no single or central authority is in full control.

**Decentralized Application (DApp)** Decentralized application are applications built using decentralized protocols and techniques. Usually a DApp consist of a smart contract and a frontend developed using standard web technologies.

**Digital signature** Cryptographic technique where a private key is used to create a string of data that can be verified using the original data and the public key of the signer.

**Diploma** A diploma is a proof of academic achievement issued by a higher education provider.

**EOA** Externally owned accounts. Account that is controlled by a private key. This type of account can initiate transactions.

**Ether** The cryptocurrency in the Ethereum platform.

**Events** Events enable the developer to log smart contract actions. Based upon the fired events (log entries) listening DApps could get information and use that information to trigger a response.

**EVM** The Ethereum Virtual Machine(EVM) is a stack based virtual machine that executes bytecode. Smart contracts are compiled down to bytecode.

**Fork** A fork is a temporary or permanent split in the blockchain resulting in an alternative chain.

**Gas** Gas is a separate currency used for paying for computational power in Ethereum. Gas has its own exchange against ether and is used to protect against the volatile price of ether.

**Gas Limit** Gas limit is the maximum amount of gas that you are willing to use.

**Gas Price** Gas price is the price you are willing to pay for gas.

**Hash** A hash is a fixed-length output produced by a hash function.

**Higher Education Provider** University or other providers of higher education recognized as reputable by a government quality assurance body.



**IPFS** The InterPlanetary File System. A decentralized p2p storage service.

**Keccak-256** This is the cryptographic hash function used in Ethereum. Keccak-256 was the winning proposal for the new SHA-3, and is now implemented as SHA-3.

**Merkle Patricia Tree** A data structure used in Ethereum to efficiently store key-value pairs.

**Merkle Tree** Merkle tree is a data structure. The top of the tree is called the root and at the bottom the nodes or leaves are found.

**Node** A node is client participating in the network and running an implementation of the Ethereum software.

**NOKUT** The Norwegian Agency for Quality Assurance in Education (NOKUT) is an independent expert body under the Ministry of Education and Research. NOKUT is responsible for overseeing the quality of higher education in Norway.

**Nonce** Nonce is in the world of cryptography a value that can only be used once. The transaction nonce is used to make the account-based model work, and the proof-of-work nonce is used to fulfill some target property.

**Oracle** An oracle is an external data source for smart contracts.

**Private key** A private key is a secret key that corresponds to a public key. The private key can be used to prove the ownership of an EAO and to create digital signatures.

**Proof of Stake** Consensus method where the validator stakes money on the next valid block.

**Proof of Work** Consensus algorithm invented by Satoshi Nakamoto for Bitcoin. Proof of Work is a process where a value is changed to satisfy some target property.

**Public key** A public key is a number derived from a private key using a one-way function. The public key can be shared publicly. Public keys can be used to verify digital signatures.

**Satoshi Nakamoto** The individual or group that invented Bitcoin.

**SHA** Secure Hash Algorithm. Hash function standard published by National Institute of Standards and Technology. In our thesis we will sometimes refer to SHA-256 which is a specific hash algorithm in the SHA family.

**Smart contract** In this thesis it refers to a computer program that runs on the Ethereum platform.

**Solidity** A procedural (imperative) programming language with a JavaScript like syntax.

**Swarm** Swarm is a decentralized p2p storage solution developed by the Ethereum foundation.

**Symmetric encryption** Encryption technique where the same key is used for encrypting and decrypting the message.

**The European Credit Transfer and Accumulation System (ECTS)**

ECTS is a system designed to make it easier for student to move between countries and to have their diploma recognized. ECTS allows credits taken at one higher education institution to be counted towards a qualification from another institution.

**Transaction** A transaction is data that is committed to the Ethereum blockchain. The data is sent by one address and targets another specific address. Transactions contains metadata such as gas limits, gas price, etc.

**Turing-complete** Turing-complete means that the system can simulate any Turing-machine.

**Vitnemålsportalen (Diploma Registry)** The Diploma Registry is a Norwegian service allowing students to share their academic results with an employer.

**Wallet** A wallet is software used as a container to protect and hold the private keys of accounts.

**Zero-address** Sending a transaction to the zero-address containing the smart contract is equal to say, “*create this contract*” .

## List of Abbreviations

**API** Application Programming Interface.

**BIP** Bitcoin Improvement Proposal.

**CPU** Central Processing Unit.

**CRL** Content Revocation List.

**CSS** Cascading Style Sheets.

**CSV** Comma-separated values (file format).

**CV** Curriculum Vitae.

**DApp** Decentralized Application.

**DASP** Decentralized Application Security Project.

**DNS** Domain Name Service.

**EBP** European Blockchain Partnership.

**EBSI** European Blockchain Services Infrastructure.

**ECDSA** Elliptic Curve Digital Signature Algorithm.

**ECTS** European Credit Transfer and Accumulation System.

**EIP** Ethereum Improvement Proposal.

**ENS** Ethereum Name Services.

**EOA** Externally Owned Account.

**EQAR** The European Quality Assurance Register for Higher Education.

**ERC** Ethereum Request for Comments.

**EU** European Union.

**EVM** Ethereum Virtual Machine.

**FR** Functional Requirement.

**FS** Felles Studentsystem.

**GDPR** EU's General Data Protection Regulation.

**GUI** Graphical User Interface.

**HTML** Hypertext Markup Language.

**HVL** Western Norway of Applied Sciences.

**IPFS** InterPlanetary File System.

**JSON** JavaScript Object Notation (file format).

**JSX** JavaScript Extended.

**MOOC** Massive open online course.

**NFR** Non-Functional Requirement.

**NIST** National Institute of Standards and Technology.

**NOKUT** Nasjonalt Organ for Kvalitet i Utdanningen.

**OWASP** Open Web Application Security Project.

**P2P** Peer-to-peer.

**RQ** Research Question.

**SAIT** Southern Alberta Institute of Technology.

**SHA** Secure Hash Algorithm.

**UI** User Interface.

**UK** United Kingdom.

**UNESCO** The United Nations Educational, Scientific and Cultural Organization.

**URL** Uniform Resource Locator.

**UTXO** Unspent transaction output.

# 1 Introduction

Every day we review documents, articles, and blog posts without a second thought. In general, we trust the sender of the information, but should we always do that? What if the sender of the information has an interest in giving false information? Another problem arises when the information is sent over an insecure network. Is it possible that someone has tampered with the content in transit? In general, trust is a good thing, but what if we could make the world trustless instead? Imagine that instead of trusting the sender of the information or that the content has not been tampered with in transit, you can simply check and verify it yourself.

Falsification of information that serves someone's personal interest is a problem. In 2015 the Chief Executive Officer of Telenor, Sigve Brekke, admitted giving false information in his CV [2]. In the United Kingdom (UK) a woman, called Zholia Alemi, from New Zealand falsely claimed to have a medical degree from Auckland University when she registered in the UK. She practiced psychiatry for 22 years before she was exposed [3]. The case with Sigve Brekke and Zholia Alemi are only two examples of people being hired using fraudulent information.

According to Khrono, a newspaper for higher education in Norway, thirteen students were registered as banned from admission to higher education in the registry of banned students because of falsified information [4]. The students in question had been registered for using fraudulent documents such as a fake diploma, other fake documents or even a diploma from a fake institution [4].

Important documents like driver licenses, birth certificates, identification papers, and academic diplomas needs a secure way of being transmitted and verified. Today this is often a burdensome process. In a report by the Risk Advisory Group from 2017, called CV Lies 2017, they state that 12 percent of candidates falsified their degree and 57 percent had discrepancies [5]. The report statistics are based upon a screening of 5,000 CVs [5].

The Norwegian Police Security Service, the National Security Authority, the Norwegian Police, and the Business and Industry Security Council [6] state in a joint report, named Personnel Security, that there are a need for background checks. The report points out that inside knowledge and trust means that current and former employees have a wide opportunity to do harm to the employer. The harm can be intended or unintended since they are not qualified for the job. The consequences could be financial loss, business secrets and sensitive information falling into the wrong hands, and reputation

damage [6]. For a higher education provider, like a university or college, the reputation damage could be huge if a student is hired using a fraudulent diploma appearing to be from that provider, especially if that person makes an error causing harm to the employer.

Academic diplomas are in this thesis defined as a diploma issued by a college or university. The solutions discussed in this thesis could probably be used on all types of important documents, but for this thesis the focus and use-case will be on the handling of academic diplomas.

Diplomas issued by a university are usually final and only subject to change if cheating or a mistake has been discovered. Documents that never or rarely changes, such as diplomas, could be suitable for use with blockchain technology. Documents that are subject to many changes or edits are probably not suited for blockchain technology. We will learn more about this later in the thesis.

Blockchain-based solutions and proposals for creating a diploma system already exist. We will in this thesis explore some of those solutions to figure out how a complete decentralized diploma system can be developed.

## 1.1 Background

Falsification of important documents presents the need for a secure and reliable background check. The purpose of the background check is to verify the information provided by the applicant, thus reducing the risk of hiring someone unqualified for the job [6]. A complete background check of an applicant could include verifying identity, educational background, checking credit, business interests, and previous employments. Background checks with all those elements would be invasive for the applicant and costly for the organization [6]. Using only an educational background check the case with Sigve Brekke and the woman from New Zealand could probably have been avoided.

Employers wanting to perform a background check in Norway needs consent. The report from the Norwegian Authorities states that consent needs to be in writing and that the consent form should contain information about [6]:

1. *Who will perform the check.*
2. *What the purpose of the check is and what the results will be used for.*
3. *What information will be verified.*

4. *What sources will be contacted.*
5. *Relevant data protection legislation and the applicant's rights.*
6. *Where and how will the information be stored.*

The above points could be answered like this when using a digital solution for checking the educational background of the applicant:

1. The company representative responsible for hiring you.
2. The purpose is to verify that your educational background is correct.
3. Your academic diploma will be verified.
4. A digital verification system will be used to verify your diploma. If discrepancies are found, then the issuer could be contacted.
5. Relevant legislate is the Norwegian law and GDPR. Therefore to perform the verification we need your consent. Information will be stored only if needed and will comply with privacy laws.
6. The information will be stored partially or fully in the company system for applications. The verification system will store the information required for making the verification work.

Norwegian students can use Vitnemålsportalen <sup>1</sup> (Diploma registry) to share their academic results with whoever they want. The service allows a student to download a PDF-version of his results or to send a link to an employer through the Diploma registry service. The PDF-file is signed digitally increasing the trust in the file. When an employer receives the file, he or she can easily see that the file is issued by the Diploma registry, but they have limited access to verify the diplomas beyond the digital signature. The Diploma registry website states that to verify that the results came from the Diploma registry a recipient must check the following [7]:

1. *Emails with links from the diploma registry will have `ikkesvar@vitnemalsportalen.no` as the sender.*
2. *The link address (the URL) will start with `https://vitnemalsportalen.no..`*
3. *After clicking the link, an icon showing a padlock will appear next to the link address.*

Forging a diploma from the Diploma registry is possible by spoofing the sender of the email. The spoofed email contains a link to a fake website

---

<sup>1</sup>Vitnemålsportalen, diploma registry <https://www.vitnemalsportalen.no/english/>

showing the falsified diploma. Another forgery option is to manipulate the PDF-file, using tools like Photoshop, while making sure that the digital signature looks intact. Regardless of forgery method used the sender face the risk that the recipient exposes the fraud. Discovery of attempted fraud could result in the applicant not getting the job, becoming blacklisted by the company, or even be reported to the police.

Availability of diploma sharing services, such as the Diploma registry, is important for student applying for jobs. If an attacker takes down the Diploma registry website or service, the result could be that students are prevented from receiving and sharing their diplomas.

Universities has traditionally issued diplomas on paper, and valid copies had to be obtained from the university itself. Paper versions contribute to deforestation, making digital versions a more sustainable solution. Digital versions can be sent without the mailing costs and are more time-efficient than the mailing option. Verification of paper versions are hard; a student might copy and alter the paper version prior to sending it. High-quality falsification of the paper versions might not be discovered. In figure 1 the process of sharing a diploma the traditional way is illustrated. The process is the same for both digital and paper versions. The risk of falsification, regardless of paper or digital version, is greatest when the student functions as an intermediary.

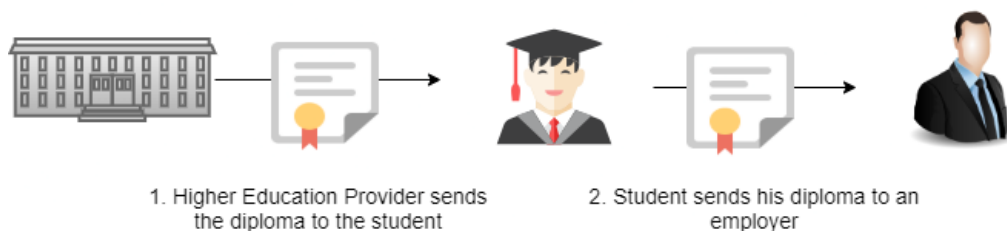


Figure 1: Sharing a diploma with an employer the traditional way.

The Diploma registry mitigates diploma fraud within Norway committed by Norwegians. Diploma fraud from foreigners remains a challenge. How can we trust diplomas from abroad? How can we verify them? Student mobility increases the need for diplomas that can be easily verified. Standardized verification of education background across borders are needed to fight diploma fraud globally.

Distinguishing reputable from disreputable higher education providers can be a challenge [8]. The European Quality Assurance Register for Higher Education (EQAR) defines a disreputable higher education provider as one offering



no real assessment of knowledge or skill [8]. EQAR defines these providers as “*diploma mills*” or “*degree mills*” [8]. The European Quality Assurance Register for Higher Education maintains a list of trustworthy quality assurance agencies [8]. All trustworthy quality assurance agencies in Europe is on the list. Disreputable quality assurance agencies on the other hand will not. EQAR defines a fake quality assurance agency as an “*accreditation mill*” [8]. Accreditation mills claim to perform quality assurance task for disreputable higher education providers to help them look legitimate [8]. The Norwegian Agency for Quality Assurance in Education (NOKUT) is on the list of trustworthy agencies [9]. Reputable higher education providers based in Norway will be accredited by NOKUT. Thus, a provider claiming to be based in Norway, but are unknown to NOKUT should be assumed to be a diploma mill.

The University of Nicosia, as the first university, started issuing their graduates diplomas on the blockchain in 2017 [10]. Later the same year, Malta announced a pilot project exploring blockchain for academic diplomas [11]. The pilot project is managed by Learning Machine, a company working closely with MIT [11].

Norway and Lichtenstein together with 27 member states of the European Union (EU) formed The European Blockchain Partnership (EBP) in 2018 [12]. EBP and The European Commission created the European Blockchain Service Infrastructure (EBSI). The purpose of EBSI is to deliver cross-border public services using blockchain technology within the EU [12]. Academic diplomas is one of four use-cases EBSI is working on in 2019, and one of their objectives is to create a prototype early in 2020 [12].

Blockchain-based projects attempts to make the administration of diplomas more secure and transparent. The idea is that everyone can verify the authenticity of the diplomas. This is like going back to one of the first ideas of a blockchain presented in a paper by Haber and Stornetta in 1991 [13]. They proposed a solution where all documents were timestamped, allowing anyone to correctly verify when a document came into existence.

Figure 2 presents a high-level overview of using blockchain technology to increase trust in academic diplomas. The transformation from the current solutions to a blockchain-based solution could help fight diploma fraud, and this will be furthered explored in this thesis.

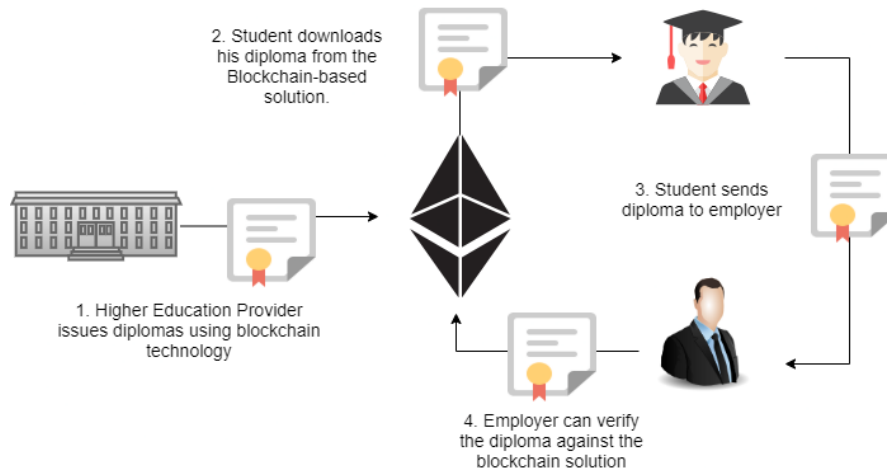


Figure 2: Sharing a diploma with an employer using blockchain technology.

## 1.2 Motivation

The motivation behind using blockchain technology is to increase trust in academic diplomas. Fraud and manipulation in higher education can severely damage reputable degrees awarded to capable people. Exploring ways of mitigating fraud with diplomas will help the correct applicant to get the job.

Imagine a world where everyone can share their diploma in a secure way with anyone in the world. Individuals with reputable degrees from their home country being recognized and easily verified in other countries. For a refugee this could mean that his education can be verified in the country he fled to, allowing him to get a relevant job and potentially ease integration.

## 1.3 Problem Description

Falsified diplomas could be a massive problem for the reputation of the issuers and for companies that potentially hires unqualified people. Higher education providers, such as universities and colleges, still issue diplomas on paper. In Norway, the Diploma registry has been a huge step in making diplomas digital. The digital versions are digitally signed and provides a certain level of trust. Paper versions of diplomas uses different measures and stamps to prevent fraud. Despite the measures taken we believe that there are challenges with our present solutions for academic diplomas. If we are

going to build a better diploma system, we need to know more about the challenges. Therefore we formulated the following hypothesis:

- H1: There are challenges with our present solutions for diplomas.

Simply pointing out challenges is not constructive. Earlier in the introduction chapter, we learned about universities exploring blockchain technology to create better diploma systems. Since we already know that falsification is a challenge regarding diplomas, one could ask if using blockchain technology could help fight falsification. Norway already provides digital diplomas through the Diploma registry, perhaps blockchain technology could improve that registry? That question can only be answered by looking into existing solutions and attempt to build a decentralized Diploma registry. Using only blockchain technology we can probably not replace the entire Diploma registry. Decentralized technologies other than blockchains might be needed. Think of decentralized technology as peer-to-peer (P2P) technology, just like the technology used for sharing pirate copies of movies, music etc. This leads to our second hypothesis:

- H2: Challenges with the present diploma solutions can be resolved by decentralizing the diploma registry using blockchain and peer-to-peer technology.

The following questions will help us to determine whether H2 is strengthened or weakened.

1. Can the proposed solution do the same as the Diploma registry?
2. Can the proposed solution match or supersede the Diploma registry in terms of availability?
3. Can the proposed solution be a better way of verifying academic diplomas?

For this thesis we will therefore map challenges with our present diploma systems and find out whether the Diploma registry can be fully decentralized or not. Based upon our hypothesis we derive the following research questions (RQ):

- RQ-1: What are the challenges with our present solutions for academic diplomas?
- RQ-2: Can the challenges with our present diploma solutions be resolved by decentralizing the diploma registry using blockchain and peer-to-peer technology?

## 1.4 Thesis Outline

The thesis is organized into ten chapters.

**Chapter 1: Introduction** The first chapter gives an introduction to the problem domain. The research questions for the thesis are presented.

**Chapter 2: Research Methodology** Describes how the research in this thesis is performed and how we can be sure that the results are valid and reliable.

**Chapter 3: Theoretical Background** Describes how blockchain technology and decentralized storage works.

**Chapter 4: Assessment of Current Diploma Solutions** This chapter aims at identifying challenges with our present diploma solutions.

**Chapter 5: Related Work** This chapter presents relevant work to our research.

**Chapter 6: BlockDiploma - Design and Implementation** Walk-through of all the requirements for our proof-of-concept. Explains the design choices and the implementation of our solution using smart contracts and frontend frameworks.

**Chapter 7: Assessment of Security** Assessment of our proposed solution based upon selected security frameworks.

**Chapter 8: Analysis and Assessment** Presents the analysis of our solution and an assessment of the results.

**Chapter 9: Conclusion** Presents the conclusion of this thesis and the answer to the research questions.

**Chapter 10: Further Work** Suggest further work and new research efforts.

## 2 Research Methodology

### 2.1 The Goal: Gaining insights and understandings through exploration

The goal for our research is to gain insights and understandings that contributes to the development of blockchain-based systems for handling academic diplomas.

### 2.2 Methodology - Computer Research using the Engineering Method

This study is classified as computing research, following the principle approach of the engineering method as described in [14]. The engineering method is to perform the following tasks: “*observe existing solutions, propose better solutions, build or develop, measure and analyze, and repeat until no further improvements are possible*”. The report categories the research into four phases: informational phase, propositional phase, analytical phase, and evaluation phase. Below, we give a short description on how each phase fits into our work.

#### 2.2.1 Informational phase

The report defines the informational phases as: “*gathering or aggregating information via reflection, literature survey, people/organization survey, or poll*” [14].

In this thesis, we survey related work on developing blockchain systems for academic diplomas. The purpose is to figure out what has been done right, where the possibilities for improvements are and how such a system should work. Information regarding legal requirements and existing standards within the EU will be gathered and used to keep our research relevant. Present diploma systems will be reviewed to figure out how they work and if there are any issues with them.

### 2.2.2 Propositional phase

The report defines the propositional phases as: “*proposing and/or formulating a hypothesis, method or algorithm, model, theory, or solution*” [14].

Based upon the information collected and aggregated in the informational phase we formulated the hypothesis and research questions. The phase involves analyzing and assessing present diploma solution to formulate requirements for a decentralized blockchain-based solution.

### 2.2.3 Analytical phase

The report defines the analytical phases as: “*analyzing and exploring a proposition, leading to a demonstration and/or formulation of a principle or theory*” [14].

The analytic phase consists of finding and analyzing solutions to the requirements formulated in the propositional phase. The goal is to develop a complete proof-of-concept demonstrating how the Diploma registry can be decentralized. This phase could be described as the development phase for our solution.

### 2.2.4 Evaluation phase

The report defines the evaluation phases as: “*evaluating a proposition or analytic finding by means of experimentation (controlled) or observation (uncontrolled, such as a case study or protocol analysis), perhaps leading to a substantiated model, principle, or theory*” [14].

After the analytic/development phase has completed, we started to assess our implemented proof-of-concept. The objective of the analysis and assessment is to present findings describing challenges with present diploma systems and whether those challenges can be resolved by decentralizing the Diploma registry or not. Findings and lessons learned are presented in a similar manner to the MIT Media Lab post: “*What we learned from designing an academic certificates system on the blockchain*” [15].

In this research we have little to no numerical data that can be used to validate or gain knowledge. Therefore, the underlying data will be of the qualitative type. The assessment will be based upon lessons taken from other

projects, lessons learned during the development, and using frameworks to analyze the security of the solution.

## **2.3 Expected results**

We expect to have a completed proof-of-concept demonstrating how a decentralized diploma system could work. The results are expected to contain a set of principles or findings describing how a complete decentralized system could be implemented. We expect the findings of this research effort to contain useful knowledge for succeeding projects.

## **2.4 Validity**

Research bias could impact our results. Relevant work is expected to become important for keeping our findings relevant and as a reference. Assessments of security is expected to give an indication on whether our decentralized proof-of-concept could survive the real world. Lessons learned and findings are presented based upon experiments on our proposed solution. If our solution could demonstrate how centralized solutions could be decentralized, then our hypothesis is strengthened. Our solution and our findings could be reviewed by others and if they arrive at the same conclusion it would be proof of the validity of our research.

Decentralized diploma systems building upon our work would be the ultimate proof of validity. However, creating a complete system ready for real world use is out of the scope of this thesis.

## **2.5 Literature**

Literature in the field of blockchain consist of mainly white papers and articles that is not peer-reviewed, although the number of scientific articles are growing fast. These documents are fundamental in the development of blockchain technology and therefor used as sources in our work. Relevant books and scientific literature will be used where they are found to be relevant for our research. Sources are discovered by looking up the references in whitepapers written by blockchain developers and relevant books. The research field of blockchain technology are in terms of research still new.



## 3 Theoretical Background

The theoretical background chapter is meant for individuals with some technical background. Readers without technical background might be able to follow, but this is not the ultimate guide to blockchain technology. The objective of this chapter is to make the reader comprehended enough to understand the proposed solution.

### 3.1 The Beginning of Blockchain Technology

The white paper "*Bitcoin: A Peer-to-Peer Electronic Cash System*", written by Satoshi Nakamoto, released in 2008 marks the beginning of blockchain technology. Satoshi Nakamoto is an alias for the individual or group that invented Bitcoin. The identity of Satoshi Nakamoto is not known [16, p. 4]. The invention of Bitcoin is important because it is a solution to a distributed computing problem known as the Byzantine Generals Problem. The Byzantine Generals problem was first presented in a paper by Lamport [17]. The problem is an abstraction of the problem with computer systems that needs to be reliable and able to cope with component failures [17]. In our case the explanation by Antonopoulos is more precise: "*the problem consists of trying to agree on a course of action or the state of a system by exchanging information over an unreliable and potentially compromised network*" [16, p. 4].

Satoshi Nakamoto's solution with Proof-of-Work to achieve consensus solves the Byzantine Generals problem if most nodes in the Bitcoin system remains honest. Honest nodes will continue to build on the chain with the most blocks in it, since that chain represents the correct information. Honest nodes will reject invalid information keeping the information in the chain valid. The result is a mechanism that can be used to achieve consensus on decentralized networks. Antonopoulos explains it like this: "*Satoshi Nakamoto's solution, which uses the concept of Proof-of-Work to achieve consensus without a central trusted authority, represents a breakthrough in distributed computing and has wide applicability beyond currency*" [16, p. 4]. The proof-of-work solution could be used to prove the fairness of elections, lotteries, asset registries, digital notarization, and more [16, p. 4-5].

All the credit for inventing blockchain technology cannot be accredited to Nakamoto since he combined previous work to create Bitcoin. The first proposed usage of a cryptographic proof of computational expenditure was

presented in a paper by Dwork and Naor in 1992 [18] as a measure to fight spam. Another important paper in regards to Proof-of-Work was Hashcash proposed in a paper by Back in 2002 [19], and this paper is cited in the Bitcoin whitepaper [20]. Previously, in the introduction, we mentioned Haber and Stornetta's work which is important for the creation of Bitcoin. Satoshi Nakamoto has listed three papers by Stornetta and Haber in the references in the Bitcoin whitepaper.

## 3.2 Blockchain Fundamentals

What is a blockchain? There are many definitions of a blockchain, and it is hard to decide what definition is the correct one. Merriam-Webster has the following definition on their site of a blockchain: "*a digital database containing information (such as records of financial transactions) that can be simultaneously used and shared within a large decentralized, publicly accessible network*" [21]. According to the Merriam-Webster website the first use of the term blockchain was in "*2011, in the meaning defined above*" [21]. If that information is correct that means that the term blockchain was first used approximately 2-3 years after Nakamoto released his whitepaper.

Dr Garrick Hileman and Michel Rauchs explains a blockchain as a type of database that is replicated over a peer-to-peer network. A blockchain differs from distributed databases by the fact that the participants in a blockchain network reach consensus about changes (transactions amongst participants) to the state of the shared database without needing to trust the integrity of any of the network participants [22]. The consensus mechanism ensures that every participant has the same view of the shared database.

Blockchains usually have the following five components [22]:

- **Cryptography:** including cryptographic hash functions, merkle trees and asymmetric encryption (private-public key pairs).
- **P2P network:** used to discover peers and sharing information in peer-to-peer manner.
- **Consensus mechanism:** an algorithm that determines the ordering of transactions in an open and potentially adversarial environment where not every participant can be assumed to be honest.
- **Ledger:** is a list of transactions grouped together in blocks that are cryptographically linked.

- **Validity rules:** are the common rules of the network used to determine what transaction are valid and how appending to the ledger works, etc.

Considering open public blockchains Antonopoulos and Wood adds the following components to the list [1, p. 2]:

- A state machine that process transactions according to the validity rules.
- One or more open source software implementations of client software for that blockchain.
- A game-theoretically sound incentivization scheme that secures the state machine while operating in an open environment.

### 3.3 Ethereum: The World Computer

Vitalik Buterin outlined the idea behind Ethereum in a paper shared with a few in December 2013 [1, p. 3]. The paper described a Turing-complete, general purpose blockchain that can function like a world computer [1, p. 3]. Alan Turing defined a system to be Turing-complete if it can be used to simulate any Turing-machine [1, p. 8]. General purpose blockchain means that a developer should be able to develop their solution without having to implement the underlying mechanism of a blockchain such as peer-to-peer network, consensus algorithms, etc [1, p. 4]. The world computer refers to Ethereum being an open source, globally decentralized computing infrastructure that executes computer programs called smart contracts. The blockchain is used to synchronize state changes, combined with a cryptocurrency called ether to meter the use of computational resources [1, p. 1].

Ethereum consists of several components, some already explained above in “Blockchain Fundamentals”. In Ethereum cryptographical techniques like digital signatures, asymmetric encryption, cryptographical hash functions and merkle trees are used. They are covered in the section 3.3.3 and 3.3.2. Ethereum has a P2P network operating a protocol called DEVp2p<sup>2</sup>, but is not covered in detail in this thesis. Consensus mechanisms is covered in section 3.3.5. The ledger with the state of Ethereum is stored locally on each node as a database, the ledger consists of all the state changes (transactions) that are considered valid. The validity rules of Ethereum is explained in

---

<sup>2</sup>DEVp2p: <https://github.com/ethereum/devp2p/blob/master/rlpx.md>

the yellow paper<sup>3</sup>. Economic security is ensured in Ethereum through the consensus algorithm and covered by section 3.3.5. The state machine is the world computer that handles transactions and transform the state based upon those transactions. There are plenty of client implementations of the Ethereum protocol like Parity<sup>4</sup> and Geth<sup>5</sup>, that handles interaction with the Ethereum blockchain. Implementation of clients is not covered in detail in this thesis and if one needs to know more, we recommend reading the footnotes.

### 3.3.1 Ethereum Compared to Bitcoin

Bitcoin and Ethereum is similar in the fact that they are both open blockchains sharing many of the same features [1, p. 1]. However, their purpose is different. Ethereum is a project meant for creating a general-purpose blockchain [23] while Bitcoin was created to become a peer-to-peer electronic cash system [20]. Having different purposes, they also use different languages. Bitcoin uses a stack-based scripting language, called Script, without loops and complex flow controls. Ethereum runs a virtual machine capable of executing code of arbitrary and unbound complexity [1, p. 2].

The developer culture of Bitcoin and Ethereum is also quite different. Ethereum's culture is described as "*move fast, and break things*", while bitcoin study every change in detail [1, p. 11]. The development of Ethereum is planned over several stages resulting in hard forks [1, p. 5]. A fork is a change in protocol causing an alternative chain, hard forks are not backward compatible so the chains cannot be joined together again. Forks could impact solutions building upon that blockchain, exactly how is hard to predict.

Another difference worth mentioning is the account model. Ethereum uses an account-based model with a counter, called nonce, to keep track of transactions. Bitcoin uses a model called Unspent Transaction Model (UTXO), and sending and receiving bitcoin means transferring control of the UTXO [16, p. 119].

---

<sup>3</sup>Validity rules (the yellow paper): <https://ethereum.github.io/yellowpaper/paper.pdf>

<sup>4</sup>Parity: <https://www.parity.io/>

<sup>5</sup>Geth: <https://geth.ethereum.org/>

### 3.3.2 Cryptographic Hash Functions

Cryptographic hash functions are essential for blockchains, and they are one-way functions that maps data of arbitrary size to a fixed-size string of bits [1, p. 71]. A cryptographic hash function needs to be deterministic meaning a given input always produces the same hash output. Easy verifiable meaning that computing the hash message is efficient. A small change to the input should change the hash output extensively, ensuring that it cannot be correlated to the original message. Irreversible in the way that it is not possible to go from hash to message without doing a brute-force search [1, p. 71]. Hash collision resistance is an important feature for avoiding digital signature forgery in Ethereum [1, p. 72]. A hash function is illustrated in figure 3.

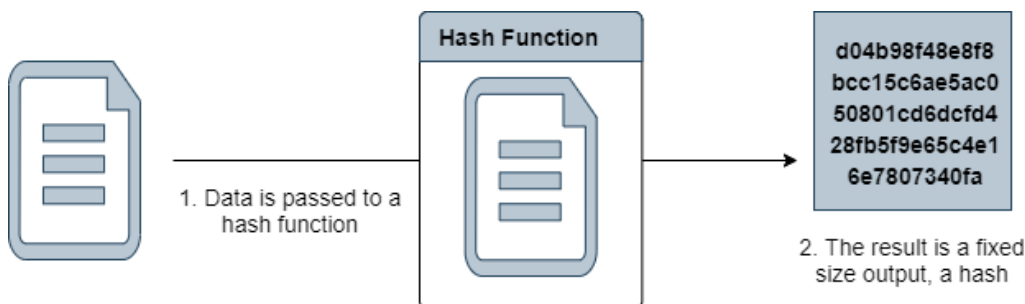


Figure 3: How a hash algorithm works.

Keccak-256 was a submission to the SHA-3 cryptographic hash function competition held in 2007 by The National Institute of Standards and Technology (NIST) [24]. Ethereum uses the original submission of Keccak-256 cryptographic hash function [23]. Bitcoin uses SHA-256 [16, p. 244].

Cryptographic hash functions are used in both Ethereum and Bitcoin for proving integrity of data. In the bitcoin blockchain each block contains a summary of all the transactions in the block using a merkle tree [16, p. 218]. Merkle trees are displayed upside down with the root at the top and leaves at the bottom. Merkle trees was proposed by Ralph Merkle as a data authentication tree, where authentication paths can be used to prove that something is a part of the tree [25]. In the example provided in figure 4, Antonopoulos demonstrate how one could prove that *HL* is included in the merkle tree. We only need to know the blue squares to prove that *HL* is in fact part of the merkle tree. Knowing the hashes in the blue squares, one could calculate the rest of the hashes and if one ends up with the correct root hash, then it has been proved that the hash is included in the tree.

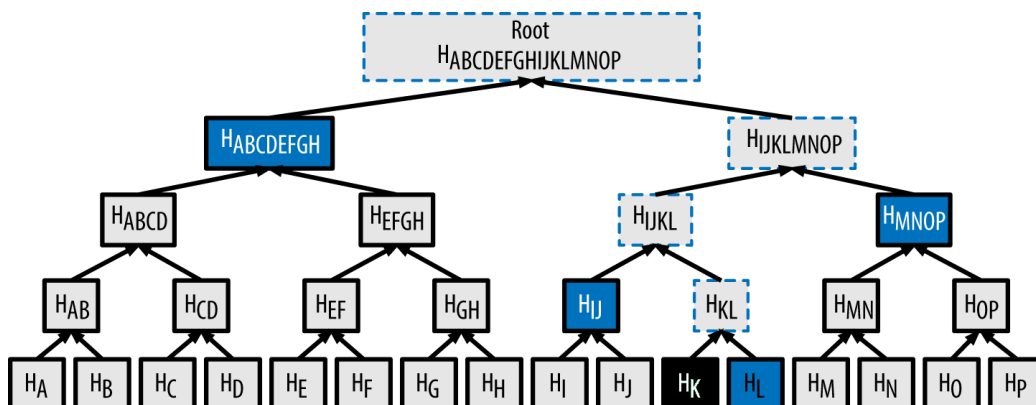


Figure 4: Proof of inclusion using a merkle tree [16, p. 220].

Ethereum, on the other hand, uses a modified Merkle Patricia tree (trie). The trie is used for creating a single value representing a key/value pair [23]. The Ethereum wiki explains it in a better way: “*Merkle Patricia tries provide a cryptographically authenticated data structure that can be used to store all (key, value) bindings. . . They are fully deterministic, meaning that a Patricia trie with the same (key, value) bindings is guaranteed to be exactly the same down to the last byte and therefore have the same root hash, provide the holy grail of  $O(\log(n))$  efficiency for inserts, lookups and deletes, and are much easier to understand and code than more complex comparison-based alternatives like red-black tries*” [25]. This can be used for proving the state in Ethereum, while the merkle tree proves that a transaction is part of a block in bitcoin.

### 3.3.3 Keys and Addresses

Addresses and keys are needed for using the Ethereum blockchain. The Ethereum platform has two different types of accounts; contract account and Externally owned accounts (EOA) [1, p. 59]. Ownership of an EOA is proved using a private key, and private keys are in the center of all user interaction with Ethereum [1, p. 59-60].

Public key cryptography (asymmetric cryptography) keeps information secure using unique keys that are based on a mathematical function with a special property: it is easy to calculate the keys, but hard to calculate their inverse [1, p. 61]. This allows the creation of digital secrets and unforgeable digital signatures secured by the laws of mathematics [1, p. 61]. In Ethereum Elliptic curve cryptography is used, which is a more advanced category of

mathematical functions [1, p. 61]. Using elliptic curve arithmetic, multiplication modulo a prime is simple but division (the inverse) is practically impossible [1, p. 61]. The details of the math behind the cryptography is outside the scope of this thesis and therefore not covered in detail.

Addresses in Ethereum is derived from public keys using the Keccak-256 hash function [1, p. 73], a process shown in figure 5. An address represents an account in Ethereum. The private key controls access to the account. Ethereum addresses are hexadecimal numbers consisting of the last 20 bytes (least significant bytes) of the hash [1, p. 74]. 0x at the beginning of an Ethereum address indicates that it is encoded in hexadecimal. The Bitcoin client has a built-in checksum to prevent mistyping of addresses, Ethereum addresses have none [1, p. 74]. The rationale behind this was that the hexadecimal addresses would be hidden behind abstractions [1, p. 74].

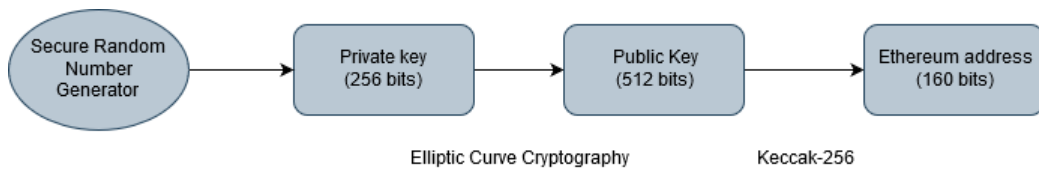


Figure 5: Process of creating an Ethereum address

The private key must be kept secret at all times to prevent unauthorized parties from taking control over the ether and contracts secured by it [1, p. 61-62]. The private key is a unique piece of information used to create digital signatures. Digital signatures can be created to sign any type of message, but in terms of Ethereum the signature is applied to transactions. The mathematics used gives us a way of combining the transactions and the private key in a way that produces a code that could only be created with knowledge of the private key [1, p. 62]. The process of signing and verifying a digital signature is shown in figure 6. In Ethereum the digital signature serves three purposes: proving authorization (ownership), non-repudiation making the denial of authorization impossible and lastly to ensure that data has not been modified after signing [1, p. 115].

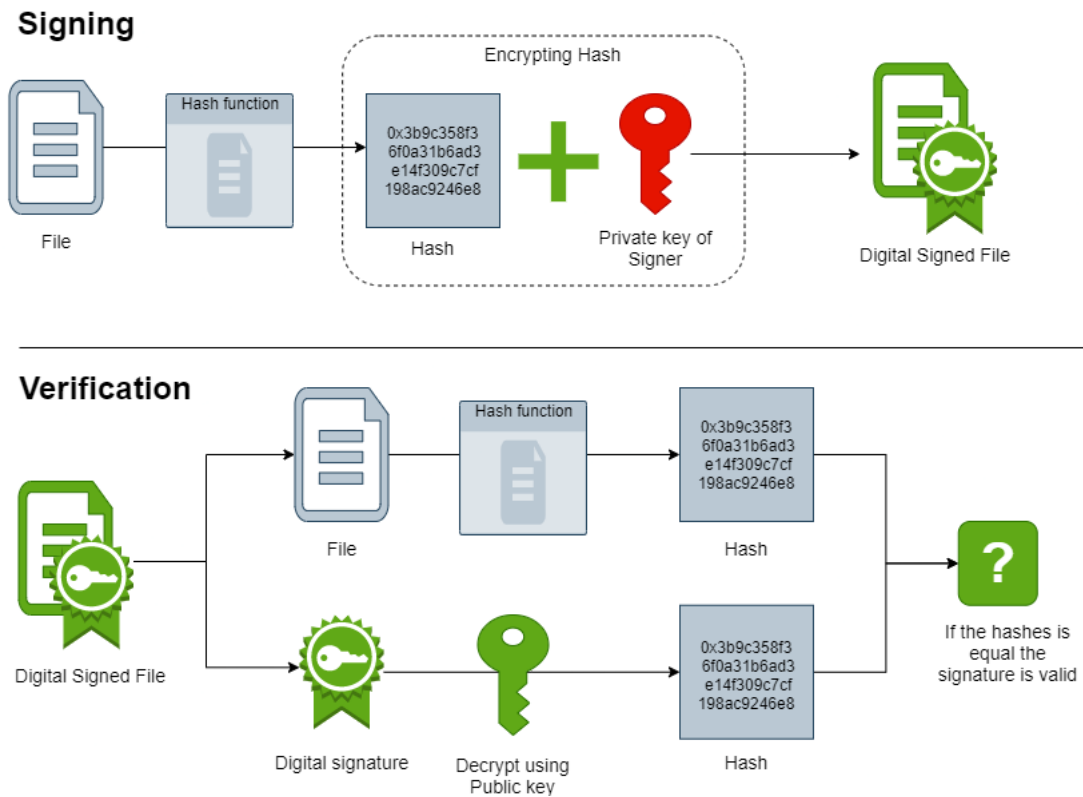


Figure 6: Process of creating and verifying a digital signature

### 3.3.4 Transactions

The Ethereum Yellow paper written by Dr. Gavin Wood defines a transaction as a single cryptographically-signed instruction constructed by an actor externally to the scope of Ethereum [23]. In other words, transactions are instructions used to change the state of Ethereum. Every state change in Ethereum starts with an EOA sending a digital signed transaction. Transactions are transmitted to the Ethereum network and recorded on the blockchain [1, p. 99].

Transactions on Ethereum platform requires computational resources, and those resources must be paid for. Ether is the cryptocurrency of Ethereum, but to protect against the volatile price of ether a separate currency called gas is used [1, p. 106]. The purpose of gas is to provide a metering mechanism to the Turing-complete systems preventing denial of service attacks or resource devouring transactions [1, p. 106]. Gas has its own exchange rate against ether.



The transaction sender can specify two fields regarding gas, the `gasPrice` and the `gasLimit`. Gas price is used to specify how much the transaction sender is willing to pay for one unit of gas, while the gas limit is the maximum number of gas units the sender is willing to buy [1, p. 106]. Some operations have a fix gas need, one of those operations is transferring ether from one account to another. A simple transfer requires 21,000 units of gas [23]. The total cost for transferring ether from one account to another will therefore be 21,000 units of gas times the price of one unit of gas.

If a transaction exceeds the gas limit specified by the sender an “out of gas”-exception is thrown. The result of an “out of gas”-exception is that every change is reverted to the original state [1, p. 158]. Gas used in a failed transaction is still taken as a transaction fee. Developers should try to avoid computational demanding operations such as dealing with dynamically sized arrays and making calls to other smart contracts to keep the costs down. Developers can use `gasEstimate` to estimate the units of gas needed to execute their code [1, p. 159]. Expensive operations could discourage users from sending the transaction to perform the operation.

In Ethereum compared to Bitcoin earlier we talked about the account-based model of Ethereum. A vital part of this model is the nonce. The nonce is part of every transaction and are in place to protect against transaction duplication and to make sure every transaction occurs in the correct order [1, p. 101-102]. The Ethereum network processes transactions sequentially based on the nonce [1, p. 104]. This means that a transaction with nonce two will not be processed and recorded before a transaction with nonce zero and one has been recorded. Recalling the transaction with nonce two is not possible since the transaction is valid, and the nodes has already received that transaction. The nonce can be dynamically calculated by counting the number of confirmed transactions on the blockchain.

Figure 9 shows a simplified illustration of a transaction. We have covered the gas price, gas limit and the nonce, but there are other parts. The digital signature is a signature of the transaction created by the sender, the `to` field is the address of the recipient, `value` is the amount of ether transferred and the data can contain smart contracts to deploy or functions to execute. Transactions can contain any combination of data and value. Transactions with only value is a payment, only data is an invocation. If the transaction contains data and value, then it is both a payment and an invocation. Sending a transaction without a value or data is possible, but that is probably to waste gas [1, p. 108].

Transmitted transactions propagates on the Ethereum network using a ”flood

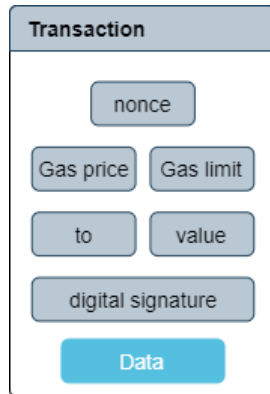


Figure 7: Simplified illustration of a transaction.

routing” protocol between the nodes in the peer-to-peer network [1, p. 123]. Successfully executed transactions ends their journey by being recorded on the Ethereum Blockchain [1, p. 123].

### 3.3.5 Consensus Methods

Consensus is a key component in any blockchain. When talking about Ethereum the most relevant ones are proof-of-work and proof-of-stake, but there exist several other consensus methods. Consensus is the ability to arrive at a common state across a distributed network, under adversarial conditions, without centralizing control [1, p. 321]. Transactions are grouped together in blocks, and consensus is the process of agreeing on the next valid block to append to the blockchain.

**Proof-of-Work** is a process where a value is changed to satisfy some target property. In other words, proof-of-work requires that the CPU performs work to satisfy the target property. In Bitcoin proof-of-work is a process of scanning for a value that when hashed, the hash begins with a number of zero bits [20]. The average work required is exponential in the number of zero bits required, and the work is verified by executing a single hash function [20]. Every block has a nonce that is incremented or changed (requires work by the CPU) until the hash of the block has the required number of zero bits [20].

When the target difficulty and all the rules for a valid block is met then a new block can be proposed by the miner of that block. In others word, the next valid block is randomly found by a miner that proposes it to other

nodes. The other nodes verify the block and show acceptance of the block by appending new blocks after it. The miner that proposed a block that other miners build further upon is awarded with newly minted coins, an incentive to contribute to securing the blockchain [1, p. 320]. How is the blockchain secured by this incentive? Simple, every miner will by default select the chain with the largest number of blocks and append to that one. That chain will have the most computational power invested, meaning that for every block added it gets harder or impossible to redo all the work, resulting in an immutable chain of blocks. An attacker is punished by the fact that he is losing money on the electricity used during an attempted attack [1, p. 320].

The results of this proof-of-work algorithm is emergent consensus [16, p. 233]. That means that there are no fixed moment or election when a block is considered valid, instead consensus emerges over time. Ethereum's Proof-of-Work Algorithm is called Ethash [1, p. 321].

Proof-of-Work mining is using enormous amounts of electricity. The Cambridge Bitcoin Electricity Consumption index estimates that bitcoin mining will use 69.03 TWh of electricity this year [26]. That means that Bitcoin with its Proof-of-Work use more energy than countries like Austria and Colombia [26].

**Proof-of-Stake** consensus keeps track of a set of validators [1, p. 321]. Anyone can become a validator by sending a special transaction that locks up their funds (staking their funds). Validators then take turns on proposing and voting on the next valid block [1, p. 321]. Votes are weighted after each validators stake [1, p. 321]. Honest validators receive a small reward when the block they put their stake on is accepted by the majority. Dishonest validator will lose their stake when the block they staked it on is rejected [1, p. 321].

Ethereum's proposed Proof-of-Stake Algorithm is called Casper. Casper is still under active research and development [1, p. 321].

Proof-of-Stake saves a lot of electricity compared to proof-of-work. One reason for using proof-of-work over proof-of-stake is the nothing at stake problem. The problem occurs when we have a fork and an attempt at rewriting the history and reversing a transaction, cause then the optimal strategy for any validator is to stake on every chain to collect a reward regardless of what chain wins [27]. Assuming that the validators motivated by financial gain follows that strategy, an attacker could potentially win and reverse a transaction since it would not require a very large stake to tip the scale. This

problem is avoided in proof-of-work since dividing the CPU-power used to mine new blocks on several forks will result in loss of electricity and therefore money.

### 3.3.6 Smart Contracts

Smart contract is a term introduced by Nick Szabo already in 1997 [28]. Szabo's idea behind smart contracts is that many kinds of contractual clauses can be embedded in the hardware and software we use in such a way that a breach of contract would be expensive for the breacher.

Antonopoulos and Wood defines a smart contract as an immutable computer program that runs deterministically in the context of the Ethereum Virtual Machine (EVM) on the decentralized world computer [1, p. 127]. They further explain that the term contract has no special meaning, and that a smart contract is simply a computer program. Immutable means that the program cannot change. Deterministic means that the outcome is the same for everyone running the contract. The EVM context gives the smart contract access to its own state, the context of the calling transaction and some information about the most recent blocks [1, p. 128]. The decentralized world computer refers to the fact that all local instances running an EVM-node produce the same final state, making the system as a whole operate as a single "world computer" [1, p. 128].

Smart contracts are built by programming languages such as Solidity, LLL and Serpent. Dr. Gavin Wood created the Solidity language explicitly for writing smart contracts [1, p. 131]. Solidity is the de-facto high level language for Ethereum [1, p. 131]. Smart contracts written in Solidity are compiled down to EVM bytecode [1, p. 131].

Error handling in smart contracts using Solidity is handled by the functions `assert`, `require`, and `revert`. All transactions are atomic meaning they either complete successfully or are reverted entirely [1, p. 148]. `assert` is by convention used for cases where we expect some internal value to be true [1, p. 148]. `require` is used for testing inputs [1, p. 148]. `revert` and `throw` halts the code execution and reverts any changes to the state [1, p. 148].

Transactions that complete regardless of being successful or not produce a transaction receipt [1, p. 149]. Transaction receipts contain log entries providing information regarding the actions that took place during the transaction [1, p. 149]. Events are the high-level objects in Solidity used to construct

log entries [1, p. 149]. These events can be used for debugging purposes or to trigger some action in a listening service.

Smart contracts are deployed with a special transaction to an address called the zero-address. The zero-address is just a destination with the special meaning of "create this contract" [1, p. 112].

All software should follow best practices for security, and smart contracts are no exception. Especially considering that anyone can send a transaction to interact with the smart contract. Antonopoulos and Wood states that defensive programming is well suited to smart contracts, and they emphasizes the following best practices [1, p. 171]:

- *Minimalism/simplicity. Complexity is the enemy of security. The simpler the code, and the less it does, the lower the chances are of a bug or unforeseen effect occurring* [1, p. 171].
- Code reuse. Do not Repeat Yourself. Use a library or contract that already exists if it does most of what you need, since code that has been extensively used and tested is likely more secure than any new code you write.
- Code quality. Smart contract exists on the blockchain forever, apply rigorous engineering and software development methodologies since every bug can lead to monetary loss.
- Readability/auditability. Following conventions makes the code easier to read, thus making it easier to audit.
- *Test coverage. Test everything. Smart contracts run in a public execution environment where anyone can execute them with whatever input they want* [1, p. 172].

### 3.3.7 Oracles

Oracles are defined by Antonopoulos and Wood as systems that can provide external data to smart contracts [1, p. 253]. Furthermore they state that oracles ideally should be trustless systems, meaning that they do not need to be trusted because they operate on decentralized principles.

Antonopoulos and Wood [1, p. 254] states that we should think of oracles as a mechanism for bridging the gap between the off-chain world and smart contracts. Implications of such a mechanism is that smart contracts can enforce contractual relationships based on real-world events [1, p. 254]. This

could introduce external risk to the Ethereum security model [1, p. 254]. One example of external risk is the case where the amount of money in a smart will is high enough to incentives a hacker to hack the oracle. The hacker could then trigger the distribution of the money prior to the owners death. Centralized oracles represents a single point of failure as opposed to decentralized oracles [1, p. 261].

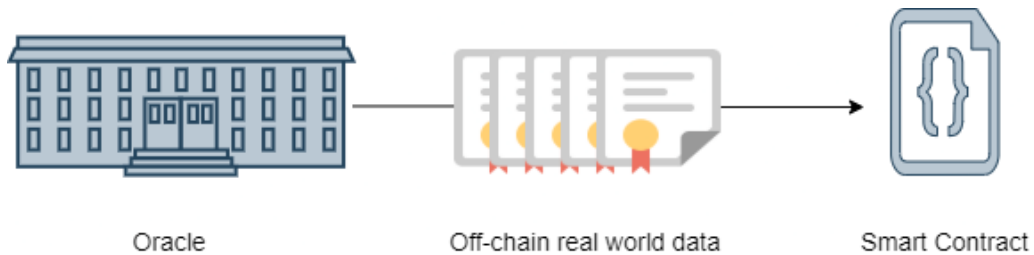


Figure 8: Higher Education Provider working as an oracle by providing diplomas

Antonopoulos definition of an oracle includes data from sources that cannot be provided trustlessly as there is no independently verifiable objective truth [1, p. 254]. Academic diplomas and government IDs where the data is provided by a fully trusted party are examples of sources included in the definition [1, p. 254]. In these cases the truth is subjective and could only be changed by appealing to the authority responsible for the information [1, p. 254]. Data authentication to prevent tampering with the data while transferring it from the oracle to the smart contracts are needed [1, p. 258]. Figure 8 shows an academic institution functioning as an oracle.

### 3.3.8 Decentralized Applications

Decentralized applications (DApps) is a term often used about smart contracts with a web frontend [1, p. 268]. Antonopoulos and Wood defines a DApp as an application that is partially or fully decentralized. Applications usually consists of backend software, frontend software, data storage, message communications and name resolution [1, p. 268]. Each of the parts can be decentralized.

Smart contracts in DApps are used to store the business logic [1, p. 269]. Antonopoulos and Wood uses the oversimplification that smart contracts can be thought of as a replacement for the server-side components in a regular application. Of course, this being an oversimplification they clarify that

smart contracts should only be used for the aspects of the application that need a trusted and decentralized application platform since program execution costs money.

The user interface of an DApp can be developed using standard web technologies (HTML, CSS, JavaScript and so on). No knowledge about the Ethereum virtual machine or languages used on the Ethereum platform is required for this part of the software [1, p. 269]. The interaction with Ethereum are often conducted by the help of a wallet extensions to the browser [1, p. 269-270].

Smart contracts and the Ethereum blockchain is probably not suited for storing large amounts of data when considering the gas costs. One option is to utilize off-chain data storage service, such services can be centralized (own server, Google Drive, Amazon Cloud Storage) or decentralized (IPFS, Swarm). Using decentralized storage like IPFS and Swarm one could store images, videos and even the entire user interface [1, p. 270].

IPFS is an abbreviation for The InterPlanetary File System, and it is a decentralized content-addressable storage system built using P2P technology. The stored objects are distributed among the peers partaking in the IPFS-network. Content addressable means that each piece of content (file) is hashed and the hash is used to identify that file [1, p. 270]. If a user wants to retrieve a specific file from IPFS he uses a node to retrieve the file using the hash. Another content-addressable decentralized storage platform is Swarm, created by the Ethereum Foundation [1, p. 270].

The Domain Name Service (DNS) allows us to use human-readable names resolving to the IP-addresses of the servers hosting the applications [1, p. 281]. Decentralized application can be hosted on a centralized server allowing access through the DNS. Ethereum Name Services (ENS) is a decentralized alternative to DNS [1, p. 281]. ENS solves the same problem but instead of human-readable names resolving to IP-addresses it resolves to Ethereum addresses. Antonopoulos and Wood gives an example using the Ethereum Foundation donation address. The address is `0xfB6916095ca1df60bB79Ce92cE3Ea74c37c5d359` or in a wallet that supports ENS it is simply `ethereum.eth` [1, p. 281].

Developing a DApp over a centralized solution has several advantages like resiliency, transparency and censorship resistance [1, p. 268-269]. Resiliency means that there are no downtime as long as the Ethereum platform is still operating [1, p. 268-269]. Transparency allows anyone to inspect the code and all transactions are recorded forever on the blockchain. DApps using the Ethereum blockchain is censorship resistant. Users capable of sending an

Ethereum transaction can interact with the DApp.

### 3.3.9 Ethereum Improvement Proposals and ERCs

Ethereum Improvements Proposals (EIPs) describes proposed and final standards for the Ethereum platform including core protocol specifications, clients APIs and contract standards [29]. They are divided into several categories: standards track, meta and informational [29].

The standard track describes any change that affects most or all Ethereum implementations. The standard track can be divided into the following categories: core, networking, interface, and Ethereum Request for Comment (ERC) [29]. ERC is used to define standards and conventions including contracts standards [29]. A few mentionable ERCs are ERC-137, ERC-162 and ERC-181 which is the improvements proposals specifying ENS.

## 3.4 Wallets

Wallet is a term used for the software that manages a user's private keys. In terms of Ethereum a wallet can be just a key management system or they can be interfaces to Ethereum-based decentralized applications [1, p. 79]. All wallets have in common that they have some sort of key-management component. Access to the user's ether is controlled by the wallet [1, p. 79]. The wallet is used to manage keys and addresses, tracking balances and to create and sign transactions [1, p. 79].

Nondeterministic and hierarchical deterministic wallets are the two primary types of wallets. In a nondeterministic wallet none of keys are related to each other. For hierarchical deterministic wallets all keys are derived from a single master key called the seed [1, p. 80]. Wallets are not covered in detail in this thesis, if one would like to know more we recommend to read BIP-32<sup>6</sup>, BIP-44<sup>7</sup>, and the Binance academy<sup>8</sup>.

When exploring blockchain-based solutions on Ethereum we will use a wallet called Metamask. Metamask is a crypto wallet and gateway to blockchain apps. Metamask is a browser extension that can be installed on Firefox, Chrome and Brave [30].

---

<sup>6</sup><https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki>

<sup>7</sup><https://github.com/bitcoin/bips/blob/master/bip-0044.mediawiki>

<sup>8</sup><https://www.binance.vision/blockchain/crypto-wallet-types-explained>



Basic usage of Metamask can be seen in the next figures. The first figure shows Metamask asking the user if he wants to connect to the DApp. If a user allows Metamask to interact with the DApp then the account address is made available to that DApp. The first figure also demonstrates how one can select which Ethereum network one would like to use; this allows a developer to run a local blockchain or use a test network to test the DApp.

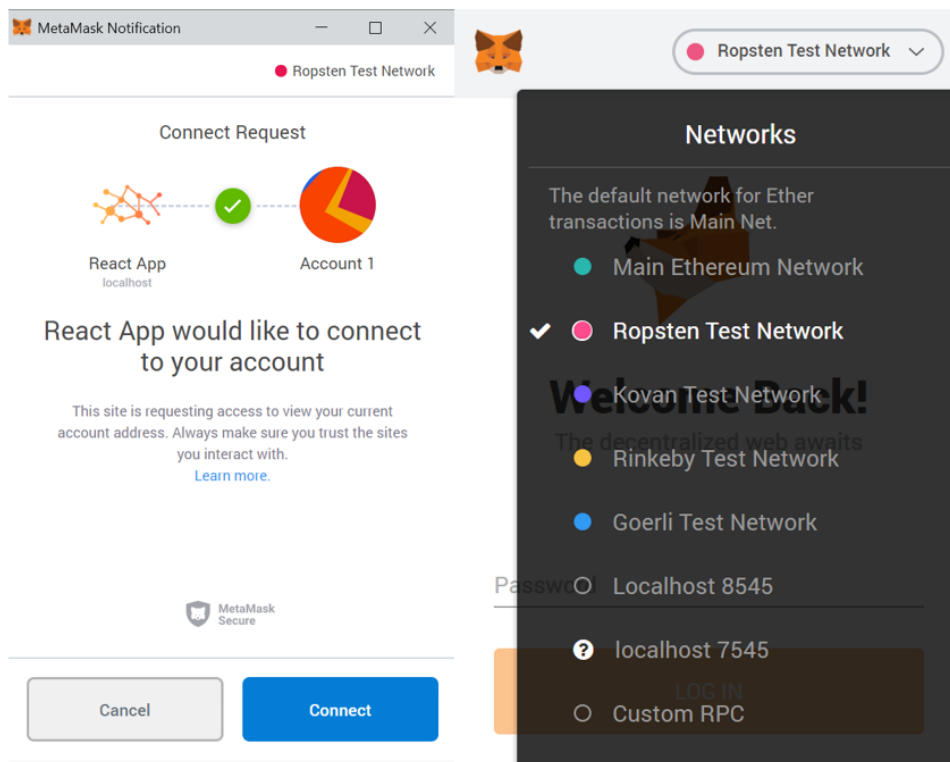


Figure 9: Connecting Metamask to DApp and choosing network.

After Metamask has been allowed to interact with the DApp actions inside the DApp that would require payment, or a digital signature can be performed. Figure 10 shows Metamask while doing a transaction against a smart contract that changes the state of that contract.

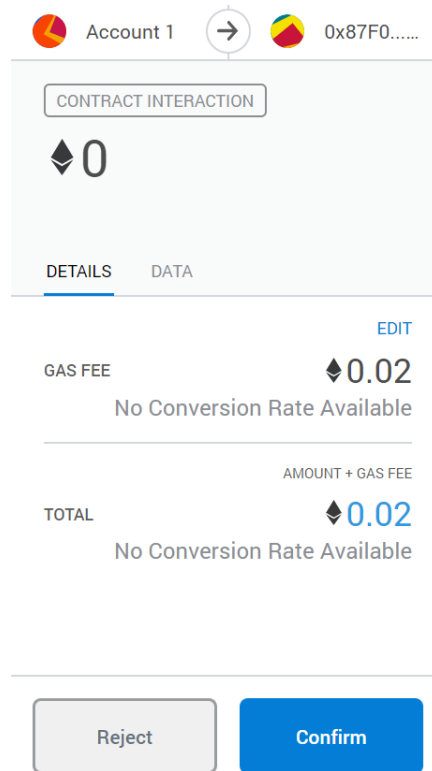


Figure 10: Transactions against a smart contract that changes state.

### 3.5 Permissioned and Unpermissioned Blockchains

Blockchains can be divided into two categories: open and closed. Open blockchains are partially or fully open for ordinary users, while closed ones are not. The table below shows different properties of open and closed blockchains.

Table 1: Blockchain types introduces some terms like read, write, commit, permissionless and permissioned. Read is simply who can read the information (transactions) in the blockchain. Writing in this context is to create new transactions, while commit explains who can append transactions to the blockchain. The commit or append operations must comply with the consensus method used by the blockchain. Permissionless blockchains allow anyone to perform the operations of read, write, and commit. Permissioned blockchain has restrictions on one or more of the read, write, and commit operations. Rule changes are also affected by the blockchain being permissioned or permissionless since the party appending transactions can “choose” what rules to follow when appending new transactions.

			READ	WRITE	COMMIT	EXAMPLE
BLOCKCHAIN TYPES	OPEN	Public permissionless	Open to anyone	Anyone	Anyone	Bitcoin, Ethereum
		Public permissioned	Open to anyone	Authorised participants	All or subset of authorised participants	Supply chain ledger for retail brand viewable by public
	CLOSED	Consortium	Restricted to an authorised set of participants	Authorised participants	All or subset of authorised participants	Multiple banks operating a shared ledger
		Private permissioned "enterprise"	Fully private or restricted to a limited set of authorised nodes	Network operator only	Network operator only	External bank ledger shared between parent company and subsidiaries

Table 1: Blockchain Types [22]

Bitcoin and Ethereum is often referred to as public blockchains which is a synonym for open and permissionless, the term private often refers to a permissioned blockchain like Hyperledger Fabric. In a private blockchain such as Hyperledger Fabric each participant has their own unique identity, enabling the use of policies to constrain access to the blockchain [31].

Private blockchains are more likely to appeal to government than public ones considering that the first one allows the owner of the permissioned blockchain to control the data [32]. Appending data to a private blockchain is carried out by the network operator only, making it easier to achieve consensus and commit transactions. Open ones would require more effort to reach consensus potentially causing scalability issues that enterprises would like to avoid.

## 4 Assessment of Current Diploma Solutions

Higher education providers in Norway currently issues diplomas on paper and through the Diploma registry. Providers in other countries has already started to use or explore blockchain solutions for the issuing and verification of diplomas. Diplomas issued on paper usually have some mechanism protecting against falsification. Falsifying a valid paper version can be done by a skilled student or by hiring someone to perform the falsification. A third option could be to buy a fake diploma online. Another problem with paper-based diplomas is that they can get lost or destroyed, and it is not certain that getting a new copy is going to be possible.

Verifying paper-based diplomas can be a time-consuming task. A report by The Office of the Auditor General of Norway (Auditor General) dated May 12, 2015 states that verification is almost never performed [33]. The report covers verification of diplomas when hiring in the educational sector in Norway. Only five percent of the institutions within the sector always verify the authenticity of the applications diploma, while 27 percent never did. 43 percent did verify the authenticity when suspecting discrepancies [33]. Proving the authenticity of a diploma requires an employer to contact the issuing institution or an agency specializing in verifying the authenticity of diplomas. In 2014 only 41 percent of the institutions did reach out to get diplomas verified [33].

The Auditor General discovered that there were no rules regulating the paper used to print diplomas on. Diplomas can be printed on normal paper or special paper. A survey performed by the Auditor General showed that 63 percent kept special paper locked up, but 63 percent also replied that they cannot guarantee that the special paper has not been stolen [33]. Dishonest employees or thieves could potentially sell that paper on the black market resulting in high quality falsification. We derive the following conclusion regarding paper-based solutions:

**Challenge 1:** Paper-based versions are time-consuming and difficult to verify.

Preventing falsification of paper-based diplomas is necessary, and that resulted in the Diploma registry. Results in the Diploma registry has already been verified by the issuer. The results can be shown on the Diploma registry website or viewed as a digitally signed PDF-file. Students using the Diploma registry can choose what to share and for how long the information is available to the recipient. The digital version provided by the Diploma

registry is a big step in the right direction for securing academic diplomas.

Digital diplomas issued by the Diploma registry cannot be falsified when used correctly. The correct use of the Diploma registry requires the diploma to be sent directly to the employer. The student must sign in on the Diploma registry website, select the results to share and enter the email address of the employer. The employer will receive an email like the one in the redacted figure 11. If the employer correctly verifies the authenticity of the email and the webpage after opening the link, then the diploma shown is valid. Figure 12 shows the certificate proving that the website is the real one.

Results from Thomas Mårstøl Reite



This is an automatically generated email sent from the Diploma registry.

You have received this email because Thomas Mårstøl Reite wants to share his/her results from higher education with you.

By following the link below, you will access the results (use the code [redacted]):  
[https://app.vitnemalsportalen.no/vp/showResultFromLink/en/\[redacted\]](https://app.vitnemalsportalen.no/vp/showResultFromLink/en/[redacted])

This link expires 07/06/2020.

Information and guidance about the Diploma registry can be found on our website.  
<https://www.vitnemalsportalen.no/>

Figure 11: Redacted email received from the Diploma registry.

What happens if the employer does not verify that the website is the real one? Assuming that the results for the Auditor General report is still valid, and employers still does not check the authenticity of the diploma, will they then check that the email is sent from the correct address and that the website is a the real one?

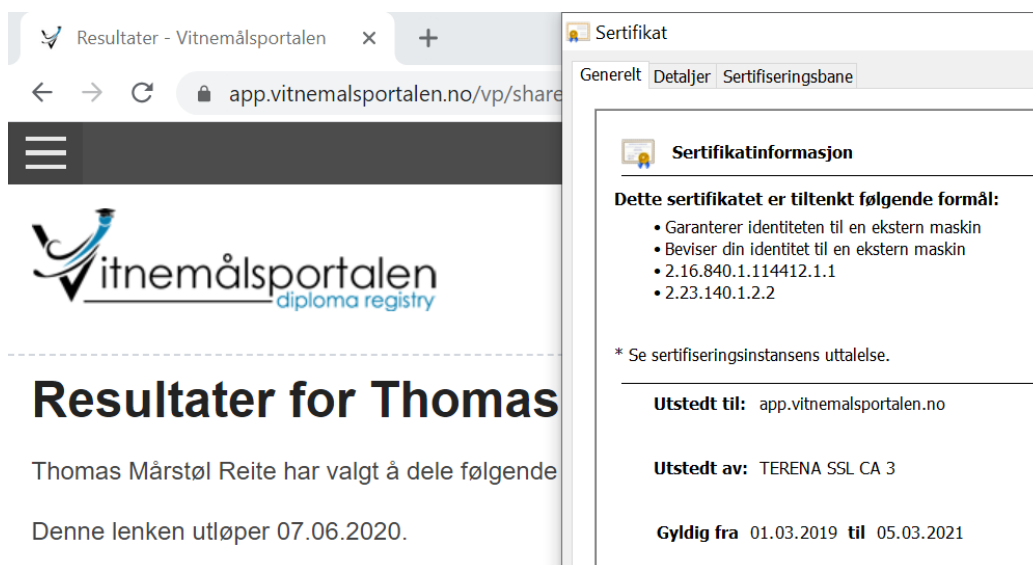


Figure 12: Certificate proving that the websites is in fact the Diploma registry.

Forging a digital diploma from the Diploma registry could be performed like this: (1) get a valid diploma from the registry, (2) create a fake diploma looking exactly the same with the results you want, (3) spoof the email address used by the Diploma registry or send from a similar email address containing a link to a fake Diploma registry website. The employer will now open the fake Diploma registry website and view what he or she believe is an authentic diploma. If an employer is not vigilant, he can be fooled. This presents the following challenge:

**Challenge 2:** Verification requires a recipient to receive an email, emails can be spoofed.

Centralized solutions stored on a single server or just a few servers could be vulnerable to denial of service attacks. Testing the Diploma registry's resilience against denial of service attacks has for obvious reasons not been done. The following scenario is therefore hypothetical. If an attacker can take down the Diploma registry website or servers, he could effectively prevent diplomas from being shared and verified through that service. We assume that the responsible parties have some sort of plan for handling such events, but they are not known to the author of this thesis. If diplomas cannot be shared because the service is not available, and they cannot be verified since the service is down, the impact on the affected students could be high. Therefore, our hypothetical scenario presents the following challenge:

**Challenge 3:** Ensuring the availability of the service, diplomas, and verification mechanism.

Employers receiving the diploma from the Diploma registry can simply check the digital signature and be sure that it is authentic, or can they? Tools like Photoshop can be used to manipulate the diploma and make sure that the signature looks intact. A student could do this himself or hire someone online. Employers that checks the signature will discover that something is not right, they might even report it to the police as attempted fraud. If an employer does not check the signature the fraud might not be detected. A student could also print a falsified diploma and send it to the employer by mail or deliver it in person, but a smart employer would demand a digital version.

We are still law-abiding citizens so we have not been able to test if we could get away with this in real life, but we have performed an experiment that everyone could do. The experiment is simple: (1) get a valid diploma from the Diploma registry, (2) open it in Adobe Acrobat Reader and check the digital signature, (3) open the file in different web browsers and check the digital signature. Part two of the experiment showed how a valid signature should be, but what happened when you attempted using a browser? When we test using Brave (version 1.8.86), Firefox (version 75.0), Chrome (version 81.0.4044.138) and Edge (version 44.18362.449.0) we are unable to verify the signature even with a valid diploma. It appears to be a similar problem with verifying digital signatures in PDF-files using mobile phones. In other words, verifying the authenticity is not necessarily as easy as it should be. We have identified another challenge:

**Challenge 4:** Verifying that a diploma in PDF-format is authentic.

Ownership of the diplomas are another challenge. Who owns a diploma? The issuer or the holder of the diploma? We believe that the holder is the owner of the information. What happens if the Diploma registry has availability issues? What happens if the databases holding the diploma information is corrupted and the backup mechanism does not work? Even worse what if a malicious employee with access deletes the data? Diplomas should always be controlled by and available to the holder if the issuer has not explicitly revoked it for legitimate reasons.

The student can in the Diploma registry view the links that has been shared (both active and expired ones), who the recipient was, when the link was created, when the link expired, and how many times it was accessed [34]. The data remains stored in the Diploma registry until the owner of the informa-

tion deletes the link or after a 12 month period [34]. Results that have been transferred to a HR-system provides the owner of the information with no method of reviewing what results have been transferred, but sometimes the law requires that the student can login and delete the transferred data [34]. The transfer of data is challenge regarding control and ownership of the data, at a very minimum the registry should provide a list of what systems and what content has been transferred to external parties. The next identified challenge:

**Challenge 5:** How can we ensure that the diploma is owned and controlled by the holder, except for cases with legitimate revocation.

Student mobility could increase the need to verify foreign academic diplomas, and without an international system of verification this could be hard to accomplish. Centralized national solutions do not necessarily help with this. A refugee from a country with a digital diploma should be able to get it verified, allowing him to use his education to get a job. The next challenge is:

**Challenge 6:** Verifying diplomas from foreign issuers efficiently and securely.

Hypothesis H1 claimed that there were challenges with our present diploma solutions, and we have identified six challenges. The challenges identified is strengthening our hypothesis. We will in the next chapters attempt to solve those challenges.



## 5 Related Work

Work related to our research is based upon the relevant projects known to the authors as of April 2020. Work discovered after that might be mentioned for transparency but has not been used actively in our work. The search and review of related work has been limited by choice to use cases with academic diplomas only. Literature and work closely related like notary services has not been reviewed in detail, but there are some mentions to some of them in this chapter. All relevant work uses blockchain technology and is partially or fully decentralized.

The University of Nicosia was the first university to issue academic diplomas where the authenticity could be verified using the Bitcoin blockchain [10]. They started in 2015 on a trial basis and from 2017 they issued all diplomas using blockchain technology [10]. The blockchain system used by the University of Nicosia is called block.co and released under a MIT license.

The MIT Media Lab proposed a Bitcoin-based solution in 2016, released under the MIT open-source license. According to MIT Media Lab the solution is a useful starting point for research and experimental projects looking into blockchain diploma systems [15].

Blockcerts is an open standard for building apps that issues and verifies blockchain-based official records [35]. The Blockcerts web page states that: *“The initial design was based on prototypes developed at the MIT Media Lab and by Learning Machine. For ongoing development, this open-source project actively encourages other collaborators to get involved. The MIT Media Lab is not actively involved in the ongoing development.”*

Block.co, the MIT proposal, and Blockcerts is all building upon the Bitcoin blockchain. Below we are looking into Ethereum based solutions. Some of the projects are not open source and therefore more difficult to review. Despite being harder to review, we expect all the solutions and systems mentioned in this chapter to be a source of inspiration in our own endeavor.

BlockEducate, developed by a company called Vottun, is one of these solutions that are not open source and harder to review. Their website states that they have a wallet solution for students and employees [36]. BlockEducate uses smart contracts and the Ethereum blockchain [36].

BCDiploma is a French Ethereum based solution using smart contracts for handling diplomas. Université Paris Descartes and Europe Business School (ESCP) is among the universities that uses BCDiploma [38]. The BCDiploma

white paper states that: “*BCD will have an inexpensive, fast-running open source ecosystem created, in order to deploy on-chain registries while respecting the right of personal data on Ethereum*”. The same paper states: “*in our opinion, the diploma’s value is based on the data’s authenticity rather than on the document itself. Therefore, BCDiploma will store the specific data directly on the Ethereum blockchain*” [37].

EvidenZ is an open source framework developed by the company behind BCDiploma [37]. The framework uses Ethereum. The rationale behind choosing Ethereum is that it is a public blockchain, necessary for users to trust it, and with an optimal safety level due to its large deployment and immutability [37]. The data is stored encrypted on Ethereum making it immutable [37]. The data can be made indecipherable by deleting the associated persistence key [37]. The persistence key is stored in a keystore controlled by the organization that issued the data [37]. Sharing data with the EvidenZ framework complies with the right to be forgotten since the deletion of the persistence key makes the data indecipherable [37]. Encryption ensures that data access can be controlled, and that data cannot be stolen [37]. The EvidenZ framework has the following architecture [37]:

- *Smart contracts (SmartValidation, SmartIdentification, SmartPublications) that is stored on the blockchain.*
- *DApps (Crypto App and Reader App) on the web.*
- *Keystore that is securely stored off-chain.*

The European Blockchain Partnership (EBP) and The European Blockchain Service Infrastructure (EBSI) as mentioned in the background section is working on creating their own solution for diplomas using blockchain technology. In a progress report from EBP called “*Progress report EBSI - Diploma User Group*” dated 07/07/2019 they state that they are still discussing how the solution should work. They are mapping options, use cases, stakeholders and working on the high-level technical specification. Currently no prototype or concrete solution exists [39].

Diploma.report is a bitcoin-based solution for academic diplomas. According to their own website they have been tested by a couple of universities [40]. The about section on their website states: “*Diploma.report is a Blockchain authentication service meant to serve the needs of graduates, schools, universities and recruiters*” [41].

CheckDiploma uses Ethereum and smart contracts for handling academic diplomas [42]. CheckDiploma allows universities to use an admin platform

where they issue diplomas in a list on a smart contract [42]. The smart contract address can then be listed on CheckDiploma and the universities own website [42]. Adding diplomas to the smart contract can only be done in a period before the diploma list is closed to prevent corruption of data [42].

Projects so far has been specific to academic diplomas, but other relevant solutions exist, like blockchain-based notary services. The notary services help proving the authenticity of documents and when they came into existence. We have not reviewed notary services in detail, but there exists several mentionable services such as Blocknotary [43], DeepVault [44] and Bitcoin.com Notary [45]. Certifaction is notary service especially worth mentioning because they currently collaborate with University of Basel and the University St. Gallen [46]. Open Source University [47] is another project worth mentioning. The last one is Smart Certificate a solution funded by the EU program Horizon 2020 [48]. Smart Certificate is a company offering a blockchain solution for all type of documents and not only academic diplomas [48].

*May 1, 2020: The Southern Alberta Institute of Technology (SAIT) has partnered with ODEM to use blockchain technology for administration of diplomas [49]. The ODEM solution was discovered too late by the author of this thesis to make use of it.*

## 5.1 Issuance of Diplomas

This section covers techniques used to issue diplomas on a blockchain-based diploma system.

### 5.1.1 Store the hash of diploma on a blockchain

Blockcerts invites a recipient to receive a blockchain diploma [35]. If the recipient accepts, he or she sends their blockchain address to the issuer. The diploma is then hashed and saved on the blockchain [35]. The diploma is then sent to the student, which again can share his diploma with an employer that can verify the authenticity using Blockcerts [35]. The cert-issuer project of Blockcerts issues the diploma by creating a transaction from the issuer to the recipient on the Bitcoin blockchain that contains the hash of the diploma [50].

Block.co allows issuers to add metadata to the diploma before a fingerprint (hash) of the entire file is included in a bitcoin transaction, after that the

diploma is given to the recipient [51]. The Block.co system issues diplomas in PDF-format [52]. The batch of PDF-files together with metadata in a CSV-file is combined and hashed using SHA-256 [52]. The hash is added to a merkle tree and the root of the tree is published on the blockchain [52].

Diploma.report computes a SHA-256 digest of the diploma being issued and stores the digest on the blockchain. Diploma.report offers to store the diploma file safely on Amazon cloud services [40]. The SHA-256 digest is recorded on the bitcoin blockchain using the OP\_RETURN function [41]. Diploma.report supports the file formats of PNG, JPG and PDF [40].

Blockcerts, Block.co and Diploma.report all uses a similar approach for issuing diplomas by hashing the diploma and storing that hash on the bitcoin blockchain.

CheckDiploma requires the university to prepare the diplomas in PDF-format [42]. The university has access to their own smart contract where they create a new list [42]. When the university needs to issue new diplomas, they are hashed and attached to the new list. The original digital diploma file will be encrypted and uploaded to Google Cloud platform. The encryption key is a security token which is shown only once and never saved, the university is supposed to print out that key and hand it personally to the student [42]. When all diplomas are uploaded, the university closes the list making it impossible to add new diplomas [42]. The addresses of the smart contracts and the number of diplomas are published on CheckDiploma and the university's own website [42].

### **5.1.2 Storing the diploma on the blockchain**

BCDiploma only allows issuers where an already trusted party has vouched for that issuer's identity [37]. Issuers are given an ID certificate that must be included in the transaction that issued the diploma, the purpose is to prove that the issuer in question did in fact issue that diploma [37]. The diploma itself is put on the blockchain and secured using cryptography [37].

BlockEducate have a certificate/badge creation tool that can be used to create a diploma [36]. Students are given a digital wallet for their diplomas [36]. Issued diplomas are permanently stored on the Ethereum blockchain and encrypted [36]. We have not been able to find out exactly how the system works.

## 5.2 Verification of Diplomas

Validation techniques used by the different projects to prove the authenticity of diplomas.

### 5.2.1 Verifying a diploma by hash

Blockcerts validates the authenticity by comparing the hash of the diploma with the hash stored on the blockchain [35]. The holder of the diploma sends the JSON-file representing the diploma to the employer in need of verifying the diploma [35]. The verifier can easily perform the verification using the Blockcerts website.

Block.co allows verification of the diploma directly from their website, or from the website of the issuer [51]. The process is simple: upload the PDF-file and the result of the verification is returned. If it can be proved that the hash of the diploma is included in the merkle tree, the status is returned as valid [51].

Diploma.report allows the original diploma file to be downloaded from Amazon Cloud Storage, so it can be used in the verification process [40]. Verifying the authenticity of a diploma requires the verifier to use a blockchain explorer aware of OP\_RETURN transactions [41]. The verifier must search the bitcoin transactions listed in Diploma.report website [40]. One of the transaction should display the diploma digest as an output and the school address as an input [40]. Using the information from the list and the transaction, we can determine if the diploma is included in the class list proving that it is valid [40].

CheckDiploma describes three different ways of verifying diplomas issued on their platform [42]. The first option is to use their website and upload the PDF. The website will calculate the hash of the diploma and check it against the hash published in the smart contract lists [42]. The second option is to use an Ethereum block explorer. Using a block explorer one must calculate the hash of the pdf-file, then go to the [diplomacheck.org/schools](https://diplomacheck.org/schools) and find the address of the smart contract holding the list [42]. The validator must then search the smart contract using a block explorer and check if the hash is included [42]. In both the first and second option you need to trust that the websites being used are not hacked. The most secure option is the last one. Option number three is to install an Ethereum node and perform the verification as explained in option number two [42].

### 5.2.2 Verify a diploma through cryptography and certificates

BCDiploma sends a URL that can be opened in their reader app [37]. The reader app will decrypt the diploma [37]. The Reader App will allow the person with the URL to see the diploma and the certificate of the issuer, proving that the issuer actually did issue the diploma [37].

The technical details of BlockEducate verification mechanism is not known. Diplomas are instantly verified by scanning a QR-code shared by the diploma holder according to their website [36].

## 5.3 Revocations of diplomas

One critical feature in a diplomas system is to be able to revoke diplomas. This section is inspired by the structure in the paper ”*Revoking Records in an Immutable Ledger*” by Konstantinos Karasavvas [53], but also contains methods not mentioned in his paper.

### 5.3.1 Centralized revocation

Centralized revocation is a straightforward approach where a diploma contains a URL endpoint used to validate the diploma. The use of the URL makes revocation trivial since an internal database is consulted. Karasavvas believes that having a centralized validator makes the use of blockchain in the first place pointless [53].

### 5.3.2 Re-issue all diplomas

Re-issue all the diplomas is a method where every diploma must be re-issued every time. After re-issuing the batch of current valid diplomas, the ones left out of that batch is effectively revoked. Despite this being easy to implement there are drawbacks. Karasavvas explains the drawback like this [53]:

- *it requires perfect management of diplomas since re-issuing the diplomas can lead to new mistakes.*
- *and every time we would have to send out every diploma to all the awardees.*

### **5.3.3 Extra transaction outputs per diploma**

The extra transaction outputs per diploma approach was used in the first version of Blockcerts [54]. The process is simple, the issuing transaction contains two extra outputs of 2750 satoshis, the issuer and the holder own one output each, if one of them spend the UTXO the diploma is considered revoked. The drawbacks of this approach is fluctuation in bitcoin prices and the infrastructure needed to manage addresses and private keys for the institutions. Karasavvas explains a scenario where issuing 1,000 diplomas resulting in 2,000 extra inputs \* 2750 Satoshi being equal to 0.055 bitcoins [53]. 0.055 bitcoins are at the time of writing equal to approximately 440 USD [55].

### **5.3.4 Revocation is handled by an issuer-hosted CRL**

Blockcerts revokes diploma using an issuer-hosted content-revocation list (CRL). The JSON-file (diploma) contains a URI that resolves to a file containing the CRL [54]. CRL revocation work similar as centralized revocation and can be considered as a single point of failure [53].

### **5.3.5 Use an additional decentralization layer for revocation**

Karasavvas explains the approach as using a second blockchain to store the revocation information [53]. The second blockchain must point at the appropriate transaction in the Bitcoin network [53]. Entries on the second blockchain holds the truth to whether the diploma is revoked or not [53]. Karasavvas states that this is a promising approach. A drawback with this approach is the added complexity both design-wise and implementation wise according to Karasavvas [53].

### **5.3.6 Blockchain specializing in diploma issuing**

Creating a specialized blockchain that can handle all the aspects of a diploma system could be another approach [53]. Using this approach, revocation can be handled by mechanism in that blockchain. Such a blockchain could be private or public, from the perspective of Karasavvas the latter is better [53]. The revocation mechanism could require that the blockchain must be redactable.

”*Redactable Blockchain - or - Rewriting History in Bitcoin and Friends*” is a paper proposing such a blockchain [56]. The paper states: “*by redaction we mean one of the following actions (and any combination of those): re-writing one or more blocks, compressing any number of blocks into a smaller number of blocks, and inserting one or more blocks. Redactions can be made only by authorized entities and under specific constraints; moreover, redactions are publicly auditable by existing miners, since they must approve the new blockchain and have access to its old copies*” [56]. According to Karasavvas these types of blockchain solutions that allows past transactions to change, will in practice introduces centralization which in turn invalidates the purpose of using a blockchain [53].

### 5.3.7 Credentialing meta-protocol

Credentialing meta-protocol is used by block.co. The concept is to encode meta-protocol information in the 80 bytes available in the OP\_RETURN in the Bitcoin system [53]. The meta-protocol described by Karasavvas contains operators specific for diplomas operations such as issuing diplomas, revoking a diploma batch or specific diplomas. Determining the validity can require that other transactions are consulted prior to the final determination of the diploma’s validity [53]. One drawback of this approach is the limit of 80 bytes means only two diplomas could be revoked at a time [53].

### 5.3.8 Revoke using cryptography

BCDiploma utilize cryptography to revoke diplomas. Their whitepaper does not state this exactly, but the way they handle the symmetric encryption keys makes it possible to revoke a diploma by deleting the key.

## 5.4 Summary

Several projects attempt to create a blockchain-based diploma system. In this chapter, we have looked at some of these projects. In table 2 an overview of the solutions we have reviewed is summarized. Bitcoin and Ethereum is the most popular blockchain for diploma systems.

In the column covering issuing of diplomas we can see that there are primarily two ways of issuing a diploma, either by storing the hash on-chain or by storing the entire diploma encrypted on-chain. On the verification part there



is primarily two different choices: hash digest (used by most of them) and cryptography, where the diploma is decrypted to show its authenticity and content.

Revocation of diplomas is the category where the solutions separates from each other the most. Diploma.report and CheckDiploma does not have any known revocation mechanism. BlockEducate and BCDiploma uses cryptography, they simply forget the encryption keys. Blockcerts uses a revocation list, while block.co uses the credentialing meta-protocol.

Solution	Blockchain	Issuance	Verification	Revocation
<b>Block.co</b>	Bitcoin	Hash on chain	Hash	Credentialing meta-protocol
<b>Blockcerts</b>	Bitcoin	Hash on chain	Hash	Revocation list
<b>BlockEducate</b>	Ethereum	Diploma on chain	Unknown	Cryptographic
<b>BCDiploma</b>	Ethereum	Diploma on chain	Cryptographic	Cryptographic
<b>EBP and EBSI</b>	Unknown	Unknown	Unknown	Unknown
<b>Diploma.report</b>	Bitcoin	Hash on chain *	Hash	None
<b>CheckDiploma</b>	Ethereum	Hash on chain**	Hash	None

\* Stores the diploma on Amazon Web Services

\*\* Stores the diploma on Google Cloud Storage

Table 2: Overview of related work

At this point we have a good understanding of what others have attempted in the field of blockchain based diploma systems.

## 5.5 Assessment of Related Work

The projects above except for the EBP-project are based upon the Ethereum or Bitcoin blockchain. One explanation for this could be the fact that Bitcoin and Ethereum are the largest blockchains at the time of writing [55]. The rationale behind choosing one of them could be that the largest blockchain are more popular and therefore more likely to endure. Another reason could be that security wise they have endured more scrutiny, and the work put into the consensus mechanisms makes the data immutable.

Every solution reviewed has some sort of mechanism for issuing diplomas to the blockchain and to verify diplomas by the help of the blockchain. BlockE-

ducate takes it a step further by introducing a student page where diplomas can be viewed and shared. The Diploma.report offers to store the diploma on the Amazon Cloud, while CheckDiploma offers to store it on the Google cloud platform.

All the solutions solve the following tasks:

- A mechanism for issuing a diploma
- A mechanism for verifying a diploma

Some of the solutions solve these tasks:

- A mechanism for sharing the diplomas
- Storage of the diplomas
- Revocation of diplomas

Most of the solutions uses the SHA-256 hash algorithm to create a fingerprint or digest of the diploma prior to storing the digest on the blockchain. The minority of the solutions store the entire diploma encrypted directly on the blockchain. The Bitcoin wiki states that storage of arbitrary data is discouraged and even viewed by some members of the community as irresponsible [57]. The Ethereum blockchain allows storing of arbitrary data, but it comes with a price.

Assuming that every diploma is one megabyte in size we can calculate the costs of issuing diplomas. In Ethereum storing a single byte costs 68 gas [23]. Eth Gas Station estimates the median gas price to 0.006 USD at the time of writing [58]. If you want to store a diploma with size of one megabyte the cost will be:

$$\begin{aligned}1024bytes * 68 &= 2176gas \\ 2176gas * 0.006USD &= 13.056USD\end{aligned}$$

The total cost would then be the number of diplomas times approximately 13 USD. Storing the diplomas on the blockchain could therefore not be considered cost-effective. Another drawback with storing the entire diploma on-chain is that the diploma is depending on that blockchain. Changing blockchains is easier when only the hash is stored on chain, but when the diploma itself is stored on chain it could be more difficult and costly to change blockchain. Hypothetical speaking a blockchain can be abandoned or subject

to a fork that breaks the functionality of the DApp. Using the largest and most popular blockchains this is of course less likely.

Storing the diploma off-chain could save money and be more efficient. Check-Diploma and Diploma.report are storing the diplomas in cloud storage, a disadvantage with this again is the introduction of centralization. Perhaps a peer-to-peer storage like IPFS or Swarm could be a better solution?

CheckDiploma raised the issue of trusting websites. If verification is performed through a website, then the verifier must trust that website. Therefore, the web user interface for verification of diplomas must be secure. Reputable security frameworks should be used to avoid the common security pitfalls. The BlockEducate solution verifies a diploma by scanning a QR-code. One clear disadvantage with such an approach is that the QR-code could be changed and lead to a fake verification page.

Revocation techniques reviewed earlier has several drawbacks such as binding up capital in locked positions, introducing centralized solutions, and re-issuing all the diplomas. All these approaches are cost-ineffective, tedious or a single point of failure. Better revocation mechanisms are needed.

Based upon the previous work a preliminary conclusion can be that blockchain technology can potentially improve the current diploma systems, but this needs to be further explored.

## 6 BlockDiploma - Design and Implementation

BlockDiploma is the name of our proof-of-concept solution. This chapter explains how BlockDiploma was developed. This chapter will cover the discovery and formulation of requirements, choice of technology, architecture, and implementation of BlockDiploma. The purpose of the chapter is to demonstrate how the Diploma registry can be decentralized. BlockDiploma is a DApp using Ethereum and smart contracts to solve our use case with academic diplomas.

### 6.1 Requirements

Requirements are formulated based upon the challenges identified in chapter four, assessment of the relevant work, information found on the Norwegian Diploma registry website, legal requirements (laws and regulation in Norway and the EU regarding diplomas, privacy and accessibility), and EBP reports and minutes. The requirements formulated below is not only inspired by the sources mentioned, some of the requirements are the authors own ideas for a complete blockchain-based diploma system.

#### 6.1.1 Stakeholders

A stakeholder is an individual or group that has a stake (interest) in the software used for the administration of academic diplomas. Using common knowledge, we can identify students, universities, and employers as stakeholders in this case. The European Blockchain Partnership states in a progress report that the following roles are involved [39]:

1. *Learner/worker/job applicant/university applicant.*
2. *Standard Educational Institutional roles*
3. *MOOC companies*
4. *Employers*
5. *Recruitment companies*
6. *National/Government Agencies*
7. *Sector specific guilds*

8. *European Agencies*

9. *Non-European Agencies*

10. *Refugees*

In other words, there are plenty of roles involved that could be considered stakeholders for our project. The explorative and technical nature of our research has led to not using systematic methods for formulating and prioritizing requirements together with stakeholders. We recognize that stakeholders are important, and prior to finalizing a blockchain-based diploma system they should be consulted. Choices had to be made and the time needed for consulting stakeholders in a systematic way would take up to much time, therefore we decided to use information about existing projects, how the Diploma registry works, and the innovative opportunities presented by decentralized technology to formulate the requirements. Our focus is on showing how the different aspects of a complete diploma system can be implemented, and not necessarily on how the final system used by stakeholders should be.

To determine if requirements are valid without consulting stakeholders, we can ask the following control questions: (1) Is the requirement part of solving any of the challenges identified in chapter four? (2) Is the requirement necessary for BlockDiploma to solve the same tasks as the Diploma registry? (3) Is the requirement helping to fulfill legal obligations? If the answer is yes to one or more of those questions, then it is a valid requirement for a blockchain-based diploma system.

### **6.1.2 Legal requirements**

The Ministry of Education and Research in Norway passed a regulation for a national portal of diplomas [59]. The regulation is specific to the Diploma registry but is still interesting for our use case. Section two in the regulation states the purpose of the portal. The national portal shall ensure truthful information about degrees, easy sharing of the information, and prevent the use of fraudulent diplomas and transcripts [59]. Sharing academic results is further regulated in section five where the regulation state that the awardee dictates access to the information. The awardee decides who gets access, to which parts and for how long [59].

New blockchain solutions can be developed regardless of the regulation, but there might be a need for regulatory changes if such a solution is to replace the

Diploma registry. The intentions of the regulation with truthful information about degrees we intend to comply with.

### 6.1.3 Non-functional requirements (NFRs)

Non-functional requirements are not related to functional aspect of software, and they are implicit or expected characteristic of software [60].

In 2016 The General Data Protection Regulative abbreviated GDPR<sup>9</sup> came into effect. When handling important documents such as diplomas the solution must comply with the GDPR-regulative. If diplomas are stored on the blockchain, then they will stay on-chain forever, and that could be a challenge regarding GDPR.

NFR-1: BlockDiploma must be GDPR-compliant.

Confidentiality of information refers to keeping information private from parties that should not have access to that information, in this case the information will be the diplomas. A diploma can contain information about a specific individual and his academic achievements; therefore, diplomas must be handled with care. Publishing all that data on an open blockchain could be a breach of privacy laws and even common sense. The GDPR-regulative must be followed to avoid liabilities and to ensure that the privacy rights of the diploma holder is adhered to. To this end we need to use some sort of cryptographic technique to ensure confidentiality and privacy.

NFR-2: Confidentiality of the diplomas must be guaranteed.

Integrity of the diploma is crucial for a functional diploma system. Diplomas must be protected from being modified or tampered with. Attempts at falsifying or tampering with a diploma should be exposed by the system. Diploma integrity should be easy to determine by anyone that needs to verify the authenticity of a diploma. Systems lacking a sound integrity mechanism cannot be a trusted verification source for diplomas.

NFR-3: Diploma integrity should be enforced to prevent falsification.

Diploma verification systems must always remain available. Students and employers should always be able to verify diplomas. Universities need to be able to issue and revoke diplomas and therefore depending on a high level of system availability. Former students need to be able to verify their

---

<sup>9</sup>EU General Data Protection Regulative: <https://gdpr-info.eu/>

diplomas for the duration of their lives. Should a university close down the verification mechanism should be unaffected. If websites are hacked or data servers corrupted, then the verification mechanism should be unaffected. The verification mechanism should be available permanently.

NFR-4: BlockDiploma needs to always be available.

Costs of using the system must be kept to a minimum. Keeping the costs down makes wide-scale adoption of the software easier to accomplish.

NFR-5: Costs of operating the system should be kept to a minimum.

Disaster recovery enabling the diploma system to function shortly after an unforeseen event is vital. Diplomas verification systems should be easy to recover, or even better the need for disaster recovery should be redundant.

NFR-6: Disaster recovery should be easy or unnecessary.

Every university or higher education provider should be able to install the verification page on their own website. Administration of diplomas should be possible to integrate with existing diploma management software. The purpose of this is to enable verifiers to verify diplomas on the universities own websites and make it easier for the staff managing the diplomas.

NFR-7: BlockDiploma should be easy to integrate to universities own websites and their diploma management system.

Diplomas are critical for the holder of the diploma, therefore there should be some sort of backup mechanism making it possible for the holder to retrieve his or hers diploma again.

NFR-8: Diplomas should be stored and backed up in a secure way.

The EU uses ECTS when credits taken at one higher education provider is to be compared with another [61]. ECTS states that 60 ECTS credits is the equivalent of a full year of study [61]. The credits can be further broken down into smaller modules [61]. The purpose of the ECTS is to enable higher education degrees within EU to be comparable. Diplomas that are comparable could increase student mobility and make it easier to apply for position in other EU countries.

NFR-9: BlockDiploma should be possible to use in all EU countries.

Unfaithful employees or hackers could attempt to misuse the diploma administration system. Attacks and misuse can be discovered and rectified by having a proper logging system.

NFR-10: All diploma actions should be logged.

#### 6.1.4 Functional Requirements (FRs)

Functional requirements are related to functional aspect of software and they define functions and functionality within and from the software system [60]. This section covers briefly all the functional requirements for our diploma system.

Issuance of diplomas is key functionality for any diploma administration system. Issuing diplomas should be easy and flexible. Higher education providers might already have a system that generates diplomas as pdf-files or similar. Diplomas (PDF-files) generated by existing system should be possible to register in BlockDiploma using the existing diploma file as input.

FR-1: Higher education providers should be able to register diplomas using existing diploma files.

The Common Student System (FS) fetches diploma data from the higher education providers own databases [62], and this data is used in the Diploma registry. Results that are approved and registered in the higher education providers databases are available to be issued as a diploma or as a transcript. Our solution should allow this type of issuance to happen automatically too.

FR-2: Higher education providers should be able to issue diplomas using automated systems.

BlockDiploma should restrict the diplomas to a widely accepted file format, making it easier for users to handle. Diploma should comply with international standards. One effort to create an international standard is the diploma supplement. The content of the diploma supplement was decided by the European commission, the council of Europe and the United Nations Educational, Scientific and Cultural Organization (UNESCO) [63]. The objective of the diploma supplement is to support the recognition of academic qualifications [63]. The EU website describe the diploma supplement like this: “*The Diploma Supplement is a document accompanying a higher education diploma providing a standardized description of the nature, level, content and status of the studies completed by its holder*” [63].



FR-3: Diplomas should be issued in PDF-format.

FR-4: Diploma supplement should be generated by the help of our solution and added to the diploma file.

Revoking a diploma might be necessary after a mistake by staff or if cheating has been discovered. Error and cheating should result in a diploma being revoked immediately to avoid that incorrect diplomas are verified as authentic. Revocation should be implemented in such a way that the university representative cannot do it accidentally. There needs to be a confirmation mechanism for this type of operations.

FR-5: Diplomas must be possible to revoke and there should be a confirmation mechanism before the revocation is performed.

Verification of diplomas is another key component in a diploma management system. Admission offices, recruiters, employers, and other relevant parties should be able to verify the authenticity of a diploma through a user-friendly public gateway.

FR-6: Diplomas should be easy to verify through a user interface.

Students should be able to access a student portal allowing them to retrieve their original diploma and to share it with an employer.

FR-7: Student should be able to retrieve their diploma.

FR-8: Students should be able to share their diplomas.

Access control is a necessary component in a diploma management system. One part of this system is logging. Higher education providers should be able to audit logs to keep track of actions regarding diplomas. An administrator should be able to give and revoke permissions to employees in the system. Administrative privileges should be possible to transfer to another account when an administrator leaves or have been reckless with his private keys.

FR-9: Higher education provider administrator should be able to give permissions to an employee account.

FR-10: Higher education provider administrator should be able to remove permissions from an employee account.

FR-11 Higher education provider administrator should be able to transfer the administrative privileges.

FR-12: Higher education provider administrator should be able to inspect logs of all actions related to diplomas.

Higher education providers that no longer wish to use BlockDiploma, should be able to terminate their smart contract. Diplomas and information about the issuer should be kept ensuring that issued diplomas always can be verified.

FR-13: Higher education providers should be able stop using our solution, but data needed for validating already issued diplomas should be kept.

Preventing diploma mills from creating their own university and publishing falsified diplomas merits the need for a higher education provider registry. NOKUT has this function in Norway and a decentralized registry that can verify if a university is reputable is needed. The decentralized registry could be managed by NOKUT or an entity with a self-interest in keeping diploma mill out of the registry. Another management system of the registry could be a multi-signature scheme or proof-of-stake variety where university stakes their reputation on the applicant being reputable.

FR-14: An authority should be able to add a university to the registry.

FR-15: An authority should be able to remove a university from the registry.

### 6.1.5 User Interface Requirements

Software that is easy to operate, quick in response, handles error effectively and provides a consistent user interface is more likely to be accepted by users [60]. Average users might struggle with using blockchain technology. Developing a decentralized application with a user-friendly interface might make the transition easier for users.

UI-1: BlockDiploma must have a consistent and user-friendly interface.

*“When websites and web tools are properly designed and coded, people with disabilities can use them. However, currently many sites and tools are developed with accessibility barriers that make them difficult or impossible for some people to use”* [64]. All people should be capable of using our software for the administration of diplomas and for verification. A regulation created by the Ministry of Local Government and modernization in Norway states that all general public websites must comply with the Web Content Accessibility Guidelines 2.0 (WCAG 2.0) [65].

UI-2: BlockDiploma must comply with legal requirements for accessibilities and comply with the Web Content Accessibility Guidelines 2.0 (WCAG 2.0).

Users in terms of a diploma systems can primarily be grouped into students, universities, employers, and quality assurance agencies. Those user groups have different needs and would probably require different user interface implementations. In Figure 13 the different users and uses cases are illustrated, as one can see they have different tasks with no overlap, therefore every users should have an adapted user interface to their needs.

UI-3: The user groups should have their own frontend/user interface allowing them to get their use-cases done.

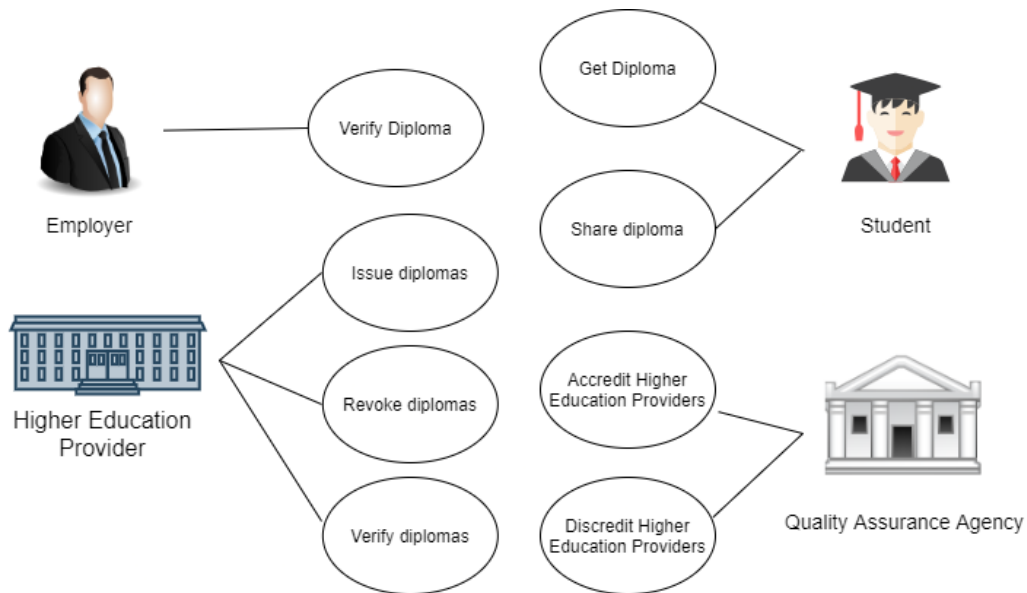


Figure 13: User groups and their use cases.

### 6.1.6 Requirements Summary

Requirements mentioned so far is summarized and categorized in this section. The requirements are divided into must have, should have, could have, and wish list. For an explanation of each category see table 3 [60]:

Below we have summarized all the different requirements and assigned them a category.

Categories	Description
<b>Must have</b>	Must be implemented, software is not operational without it.
<b>Should have</b>	Enhancing the functionality of software, debatable with stakeholders.
<b>Could have</b>	Software can still properly function with these requirements.
<b>Wish list</b>	These requirements do not map to any objectives of software.

Table 3: Requirement Categories.

ID	Description	Category
<b>NFR-1</b>	BlockDiploma must be GDPR-compliant.	Must have
<b>NFR-2</b>	Confidentiality of the diplomas must be guaranteed.	Must have
<b>NFR-3</b>	Diploma integrity should be enforced to prevent falsification.	Must have
<b>NFR-4</b>	BlockDiploma needs to always be available.	Must have
<b>NFR-5</b>	Costs of operating the system should be kept to a minimum.	Wish list
<b>NFR-6</b>	Disaster recovery should be easy or unnecessary.	Should have
<b>NFR-7</b>	BlockDiploma should be easy to integrate to universities own website and their diploma management system.	Could have
<b>NFR-8</b>	Diplomas should be stored and backed up in a secure way.	Should have
<b>NFR-9</b>	BlockDiploma should be possible to use in all EU countries.	Wish list
<b>NFR-10</b>	All diploma actions should be logged.	Should have

Table 4: Non-functional Requirements.

ID	Description	Category
<b>UI-1</b>	BlockDiploma must have a consistent and user-friendly interface.	Must have
<b>UI-2</b>	Solution must comply with legal requirements for accessibilities and comply with the "Web Content Accessibility Guidelines 2.0 (WCAG 2.0).	Must have
<b>UI-3</b>	The user groups should have their own frontend/user interface allowing them to get their use-cases done.	Must have

Table 5: UI-requirements.

ID	Description	Category
FR-1	Higher education providers should be able to issue diplomas from already existing files.	Must have
FR-2	Higher education providers should be able to issue diplomas using automated systems.	Should have
FR-3	Diplomas should be issued in PDF-format.	Must have
FR-4	Diploma supplement should be generated by the help of our solution and added to the diploma file.	Should have
FR-5	Diplomas must be possible to revoke and there should be a confirmation mechanism before the revocation is performed.	Must have
FR-6	Diplomas should be easy to verify through a user interface.	Must have
FR-7	Student should be able to retrieve their diploma.	Should have
FR-8	Students should be able to share their diplomas.	Should have
FR-9	Higher education provider administrator should be able to give permissions to an employee account.	Must have
FR-10	Higher education provider administrator should be able to remove permissions from an employee account.	Must have
FR-11	Higher education provider administrator should be able to transfer the administrative privileges.	Must have
FR-12	Higher education provider administrator should be able to inspect logs of all actions related to diplomas.	Must have
FR-13	Higher education providers should be able stop using our solution, but data needed for validating already issued diplomas should be kept.	Should have
FR-14	An authority should be able to add a university to the registry.	Must have
FR-15	An authority should be able to remove a university from the registry.	Must have

Table 6: Functional Requirements.

## 6.2 Choice of Technology

In this section the technology stack used is explained briefly and the reason for selecting that technology elucidated.

Storing diplomas on the blockchain is a cost-effective way of storing and securing vital information [66]. This could be a promising technology for

all types of permanent, or relatively permanent public documents according to a paper by Svein Ølnes [66]. Based upon his work we chosen to use a blockchain for the verification of academic diplomas.

### 6.2.1 Why Ethereum and not some other blockchain?

Ethereum was built to be a general purpose blockchain with support for smart contracts, while Bitcoin was built for being electronic cash. Since Bitcoin is built for payment the stack-based scripting language is limited, while Ethereum is Turing-complete meaning capable of simulating any Turing-machine. One could argue that developing blockchain programs with Ethereum is easier than with bitcoin. However, bitcoin could be used to solve other use-cases than payment as demonstrated in several projects mentioned in the related work chapter.

Among the permissionless blockchains with smart contracts capabilities Ethereum is the largest [55]. Ethereum is an open permissionless blockchain that is not limited by a policy set by an owner. The Turing-completeness, size of the blockchain and the fact that it is open made us choose Ethereum.

### 6.2.2 Truffle Suite

After selecting Ethereum as the blockchain, we started considering frameworks that could be used. There are a few frameworks one could choose like the Truffle Suite, Waffle or Embark. The Truffle Suite is selected because it is an easy to use and powerful tool for developing DApps. The Truffle Suite consists of Truffle, Ganache and Drizzle.

**Truffle** is a development environment, testing framework and asset pipeline for blockchains using the Ethereum Virtual Machine (EVM) [67]. Truffle provides a built-in smart contract compilation and deployment management system that allows us to deploy to any blockchain both private and public. Another feature is the automated contract testing [67].

**Drizzle** Drizzle handles the connection between the frontend and the smart contracts using a redux store<sup>10</sup>. Drizzle extends web3 contracts but still gives the option of using web3 functions [68] ”*web3.js is a collection of libraries*

---

<sup>10</sup>Read more about redux stores here: <https://redux.js.org/>

which allow you to interact with a local or remote Ethereum node, using a HTTP or IPC connection” [69].

The truffle suite offers boxes or templates that can be unpacked and used by the help of truffle console, and for this solution the drizzle truffle box [70] is used.

**Ganache** Ganache is a personal local blockchain that can be used for the entire development cycle of decentralized applications [71]. Ganache was formerly called TestRPC [71]. Ganache offers a graphical user interface (GUI) allowing the developer to easily inspect the content of the blockchain while testing new functionality, the GUI can be seen in figure 14.

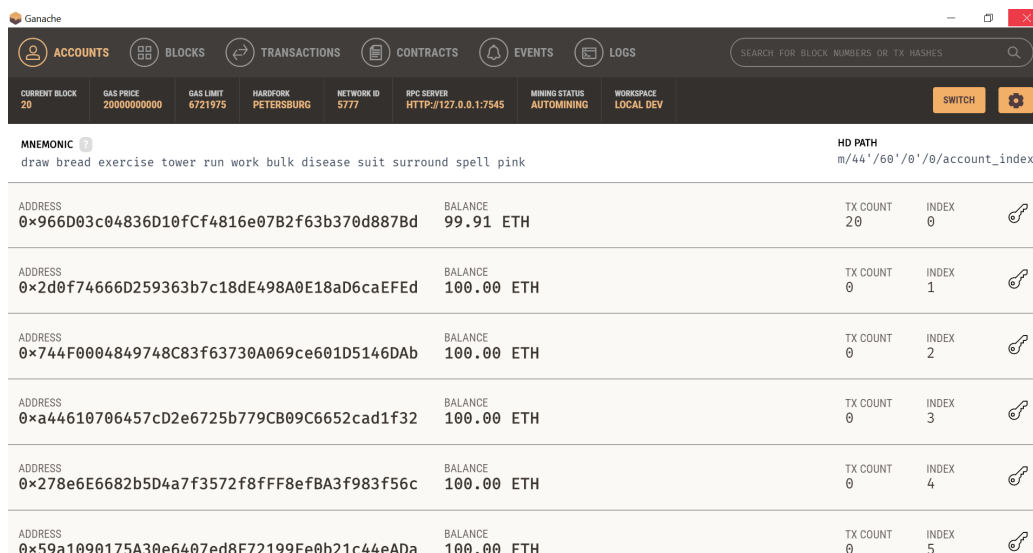


Figure 14: The Ganache GUI used for local development.

### 6.2.3 ReactJS

The JavaScript framework React from Facebook is used to code the frontend in BlockDiploma. ReactJS is chosen because it has already been tested with the Truffle suite and there exists some tutorials on YouTube and other platforms explaining how different aspects of a decentralized application can be solved. This is especially important considering that decentralized applications are a relatively new concept.

### 6.2.4 Why IPFS and not Swarm?

IPFS and Swarm both offers decentralized storage. Swarm has an advantage over IPFS when considering Ethereum as the blockchain since the Ethereum Foundation is the creator of Swarm. The main reason for selecting IPFS over Swarm is the fact that we use Infura to connect to the Ethereum blockchain, when developing BlockDiploma Infura had support for IPFS and not Swarm. Infura provides API access to both IPFS and Ethereum [72].

## 6.3 Architecture

BlockDiploma consists of three main parts: Smart Contracts on Ethereum, IPFS for storage and a frontend using standard web technologies. Infura is used for communications between the frontend and the smart contracts, and between IPFS and the frontend. The communication flow between the three main parts is illustrated in figure 15. Some interactions between the smart contracts and the user interface requires a wallet like Metamask. Files stored on IPFS are encrypted by the frontend prior to uploading it for confidentiality and integrity reasons.



Figure 15: BlockDiploma interacting with IPFS (storage) and Ethereum (smart contracts) using Infura.

The BlockDiploma DApp consists of three smart contracts with specify responsibilities:

- **The University Registry Smart Contract** is responsible for handling the list of reputable higher education providers allowed to interact with the diploma registry smart contract. The decentralized registry is functioning like a quality assurance agency performing some of the task usually done by quality assurance agency like NOKUT.



- **The Diploma Registry Smart Contract** is the contract holding the meta-information about every diploma. The contract is used for verifying diplomas, and for students to retrieve their diplomas from BlockDiploma.
- **The University Smart Contract** is an example contract of a university contract implementations. This contract interacts with the diploma registry smart contract and is responsible for specifics for that university.

BlockDiploma has four main user groups: quality assurance agencies, students, employers, and higher education providers. In figure 16 an overview of the BlockDiploma DApp is presented. Every user group has their own part of the BlockDiploma frontend that they can access through a web browser. For some of the operations in BlockDiploma a wallet is required.

Figure 16 shows how the different parts of BlockDiploma interact. In this case there are two user groups serving as oracles: the quality assurance agency and the higher education providers. Higher education providers are providing the diploma registry smart contract with correct diploma information through their own smart contract, and in that sense working as an oracle. When providers perform an action that require a change in the diploma registry smart contract, the university registry smart contract is consulted to determine if the sender is a reputable university added in that registry. Quality assurance agencies providers own the truth about what higher education providers are reputable and who is not.

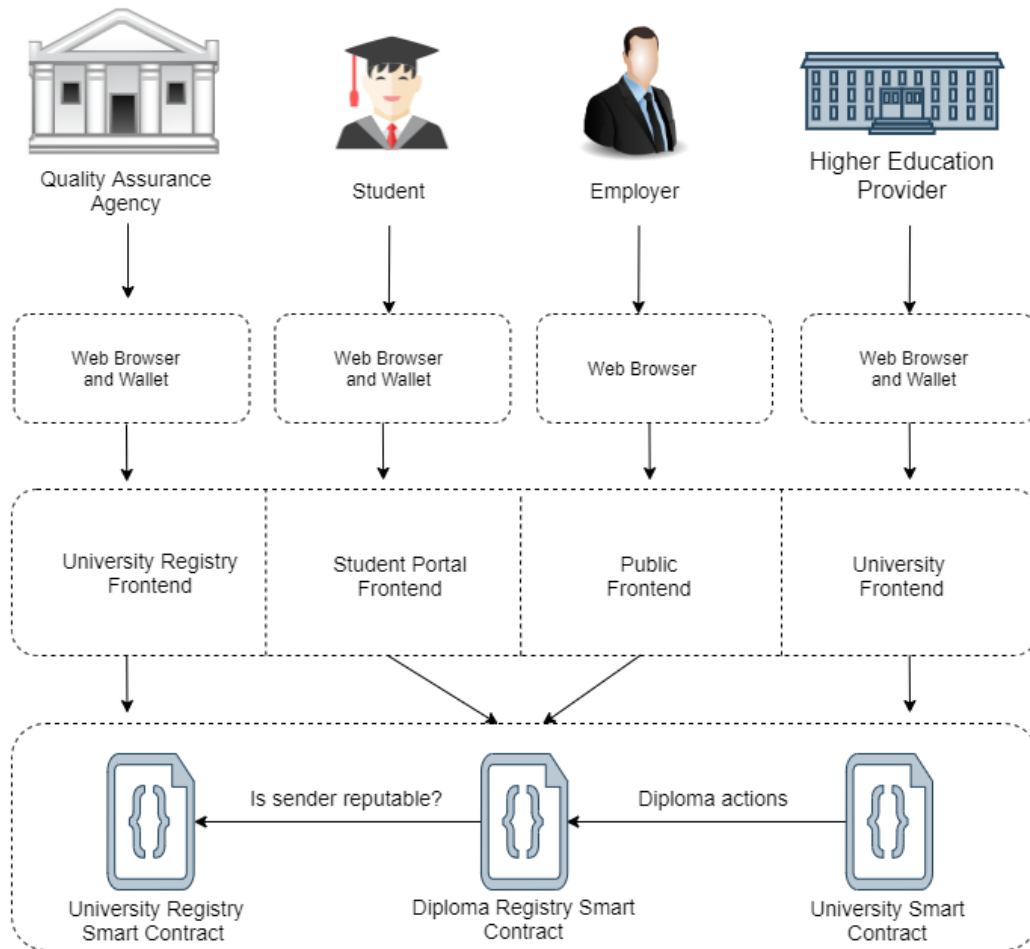


Figure 16: BlockDiploma Overview.

## 6.4 Implementation

BlockDiploma is implemented as a proof-of-concept and therefore the implemented system is not necessarily ready for use in the real world. The purpose of BlockDiploma is to demonstrate how the Diploma registry can become fully decentralized using decentralized technologies.

### 6.4.1 Authentication with Ethereum Address

When using the student portal, university, or university registry frontend a wallet is needed. A wallet provides an Ethereum address used to give access to the frontend, a process illustrated in figure 17. We have taken a

shortcut that allows anyone capable of providing a correct address to view the frontend of the application. Our frontend authentication is in other words flawed, but the smart contract is still secure because any change in the state would require a valid digital signature. The frontend could become better secured if we required a digital signature signing a value, and then use that value as proof that the user is who he says he is, prior to showing him the restricted parts of the DApp.

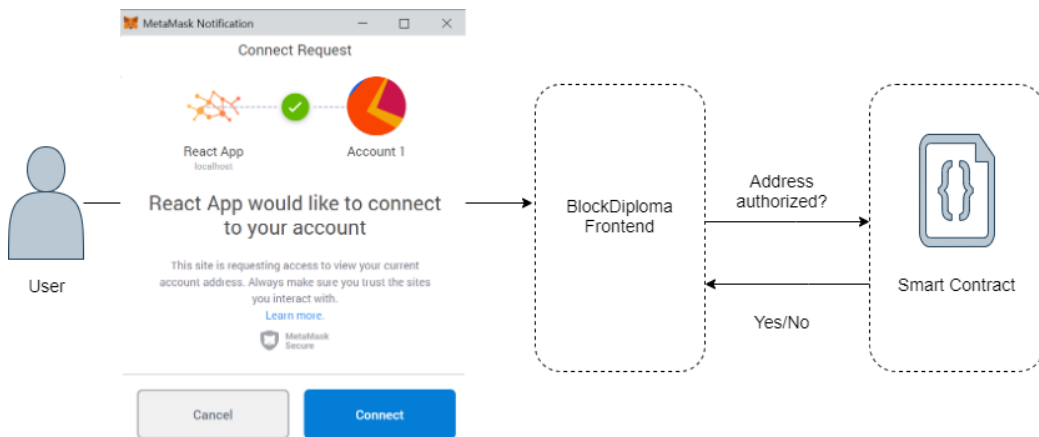


Figure 17: The currently flawed wallet Authentication and Authorization flow.

### 6.4.2 Issuing Diplomas

The first two functional requirements concern the different techniques of issuing diplomas, which are from an existing file or using an automated system that fetches the data and generates the diploma. Regardless of method the process used to publish a diploma to the diploma registry smart contract is the same. In figure 18 the process is simplified: the higher education provider issues a diploma using one of two techniques and the diploma is hashed and sent to the providers own smart contract. The second transaction containing the diploma hash goes to the diploma registry smart contract. The contract stores the hash of the diploma together with some meta-data about the diploma and recipient, but only if the university registry smart contract says that the sender is authorized.

Currently we store as little information as possible in the smart contracts. Below the code struct representing a diploma and a recipient can be viewed. The Boolean valid is true if diploma is valid and the issuerAddr is the Ethereum

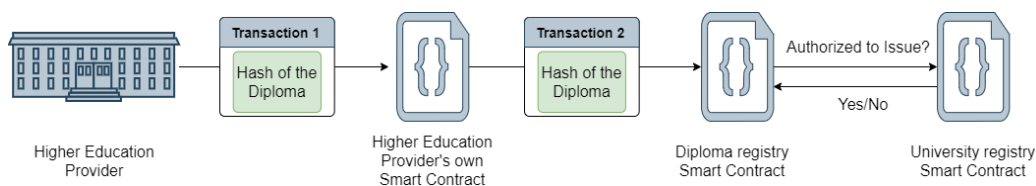


Figure 18: Process of issuing a diploma.

address of the smart contract belonging to the issuer. The IPFS string contains the content-addressable hash used to fetch the diploma from IPFS. The mapping called `diplomas` maps the hash of a diploma to a diploma struct, while the mapping called `recipients` maps an Ethereum address belonging to a student to the recipient struct.

```

1
2  /// @notice this mapping contains all the valid diplomas.
3  mapping(bytes32 => Diploma) diplomas;
4
5  /// @notice this mapping contains all the recipients.
6  mapping(address => Recipient) recipients;
7
8  /// @dev Structure used to represent a diploma.
9  struct Diploma {
10     bool valid;
11     address issuerAddr;
12 }
13
14 /// @dev Structure used to represent a diploma holder or
15     recipient
16 struct Recipient {
17     bytes32 ofDiploma;
18     string ipfs;
19 }

```

Listing 1: Diploma and recipient mapping

The Diploma Registry is implemented without a controlling entity or self-destruct function to ensure that issued diplomas remain issued. The implemented Diploma Registry contract can be viewed in appendix A.

The complete process of issuing a diploma with BlockDiploma is:

1. Provide diploma data either by an existing file or by having BlockDiploma generate the diploma file.
2. The diploma file is hashed.
3. The diploma file is encrypted and uploaded to IPFS.

4. IPFS returns hash that can be used to retrieve the diploma.
5. The hash of diploma and the IPFS hash is stored on the diploma registry smart contract.
6. The diploma is now issued and can be verified and retrieved.

Figure 19 shows a screenshot of the BlockDiploma frontend for issuing a diploma from an existing file.

**Issue a Diploma**

Select a PDF-version of the Diploma to publish it to the DiplomaRegistry

Student ID Student Ethereum Address Bla gjennom ... Ingen fil valgt

Issue Diploma

Figure 19: Screenshot of form used to issue diplomas using an existing file.

Automated systems using APIs to issue diplomas are the way to go for a complete and successful diploma system. BlockDiploma does not solve the entire process of automated issuing of diplomas, but we have implemented the first steps in making that happen. BlockDiploma has a manual form for typing in diploma data, even if this could function like a backup system, it is not likely to be used considering that it is a time-consuming effort. Figure 20 shows a screenshot of the user interface with the form. The rationale behind implementing the manual form solution was to make BlockDiploma capable of generating correctly formatted PDF-files based upon data objects. By adapting the data objects, to fit with the diploma data fetched from databases using APIs, we can make BlockDiploma capable of becoming a more automated system for issuing diplomas.

One day BlockDiploma might be part of a fully automated system that automatically issues diplomas on behalf of a university when all the conditions have been fulfilled. As stated earlier the Diploma registry fetches the diploma data directly from the universities own databases. Should BlockDiploma do this then there must be some mechanism for paying for the transactions required to issue the diplomas. Implementing this is out of the scope of this thesis because of the time needed and the need for cooperation with relevant parties, but we recognize that this is a key feature for creating a complete diploma system.

BlockDiploma only issues diplomas in PDF-format as required by FR-3. That means that if a higher education provider would like to register a diploma on

**Manual Registration of Diploma**

**Student details**

Family Names  Given Names  Date of Birth  Student ID  Address  [Lock Student](#)

**Add Courses**

	Course code and name	Semester	Credits	Grade	Distribution	
▼						<a href="#">Add</a>

Current Courses

**Common Courses**

**Programme Courses**

**Specialization Courses**

**Elective Courses**

[Issue Diploma](#)

Figure 20: Screenshot of form used to issue diplomas manually.

BlockDiploma from an existing file then that file should be in PDF-format. If the manual system is used, the first step to create an automated system, then the diploma is generated as a PDF-file by BlockDiploma. File generation is handled by the help of a JavaScript library called React-PDF. Figure 21 shows an example of a BlockDiploma generated diploma in PDF-format.

Diploma should be followed by a diploma supplement and generated diplomas will contain one. Using the option of issuing from an existing diploma file, one should include a diploma supplement prior to registering the diploma with BlockDiploma, otherwise the diploma might not be recognized by others. A screenshot of a BlockDiploma generated diploma supplement can be viewed in the appendix.



Name: q q  
Date of Birth: q

## Higher Education

### Western Norway University of Applied Sciences

Degree: Bachelor of Engineering (180 ECTS) achieved 25/06/2018

Study programme: Computing

Note: Bachelor i ingeniørfag

Note: Studiet er fullført ved studiested Bergen.

Course	Semester	Credits	Grade	Distribution
<b>Common Courses</b>				
DAT103 Databaser og os	Autumn 2015	10	A	n/a
<b>Programme Courses</b>				
<b>Specialization Courses</b>				
<b>Elective Courses</b>				

The distribution of grades is shown by the percentage for courses using the graded scale A-F. Fail (F) is not included in the distribution. All results from the last five years are included in the calculation. The distribution is also shown for courses that have been active for less than five years. There has to be at least 10 approved results during the period.

DOCUMENT GENERATED: MON, 23 MAR 2020 09:38:57 GMT

The document is electronically produced by BlockDiploma – the Decentralized and trustless diploma portal for educational results and contains results from the issuing institutions administrative system. The document can be verified by using the website of blockdiploma, <https://www.blockdiploma.com>

Figure 21: Screenshot of diploma generated by BlockDiploma.

### 6.4.3 Revoke a Diploma

If a mistake is made or cheating is discovered a diploma might have to be revoked. BlockDiploma allows a diploma to be revoked by the issuer. To revoke a diploma, the issuer needs either the existing diploma file or the student ID of that student. BlockDiploma offers a search and revoke functionality, see figure 22, where the university can search for diploma by typing in the student id code. After finding the diploma they can simply press the delete icon and confirm in the popup screen that they do in fact want to revoke this diploma.

After the higher education provider has located the diploma and confirmed that they want to revoke the diploma, the following process takes place:

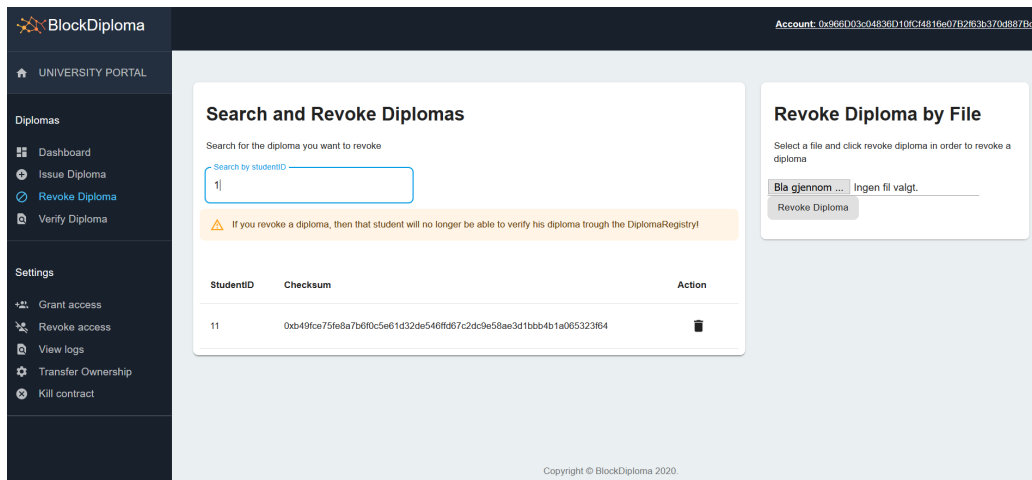


Figure 22: Screenshot showing the screen used to revoke diplomas.

(1) a transaction containing the hash, called checksum on figure 23, is sent to the higher education provider's own contract, (2) the contract sends a second transaction to the Diploma registry smart contract that the diploma with this hash should be revoked. (3) The Diploma registry smart contracts check with the university registry smart contract whether the sender of the transaction is a reputable university allowed to perform this action. If the answer is yes, then the diploma valid status is changed from true to false.

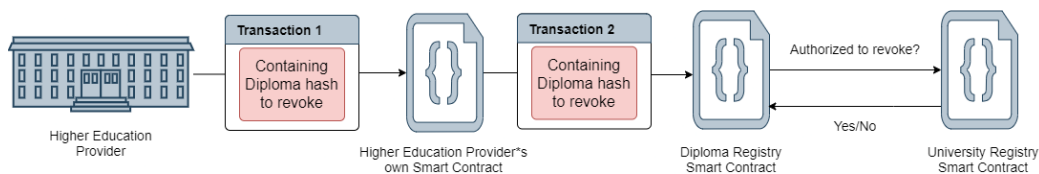


Figure 23: Process of revoking a diploma.

#### 6.4.4 Verify a Diploma

Verification of diploma should be simple, and it is using BlockDiploma. After a student has shared his diploma with an employer, that employer can visit the public site of the BlockDiploma DApp to get it verified. The employer simply selects the diploma and submits the file to the BlockDiploma DApp. The diploma is hashed, and the validity status is checked against the diploma registry smart contract. If the valid status is true then the message of “*valid diploma*” is returned, else the message “*could not be verified, contract issuer*” is returned. See figure 24 for a visual overview of the verification process.



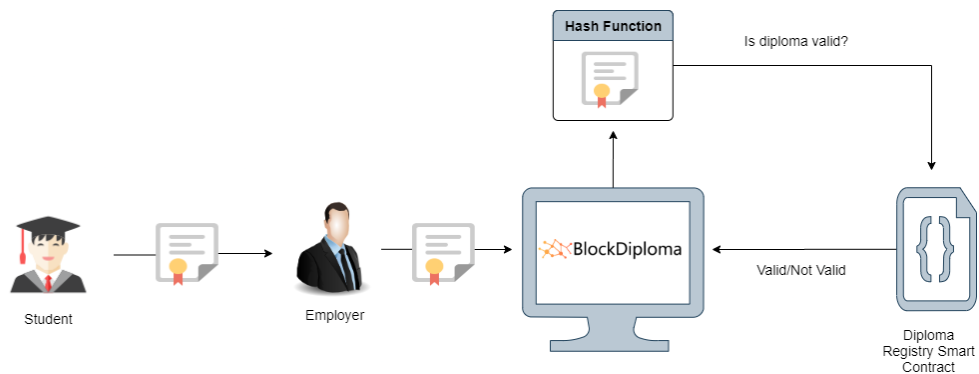


Figure 24: Process of verifying a diploma.

### 6.4.5 Student Portal

BlockDiploma has a student portal meant to serve as a way for student to retrieve and share their diplomas. Students get access to the student portal by selecting the Ethereum address given to or received from the university they attended in their wallet. After the student has provided his address to BlockDiploma, information about the diploma such as the IPFS content-addressable hash and metadata about the issuer is retrieved. The diploma is fetched from IPFS and decrypted so that the student can download the diploma. The process is illustrated in figure 25 and in figure 26 a screenshot of the student view is shown.

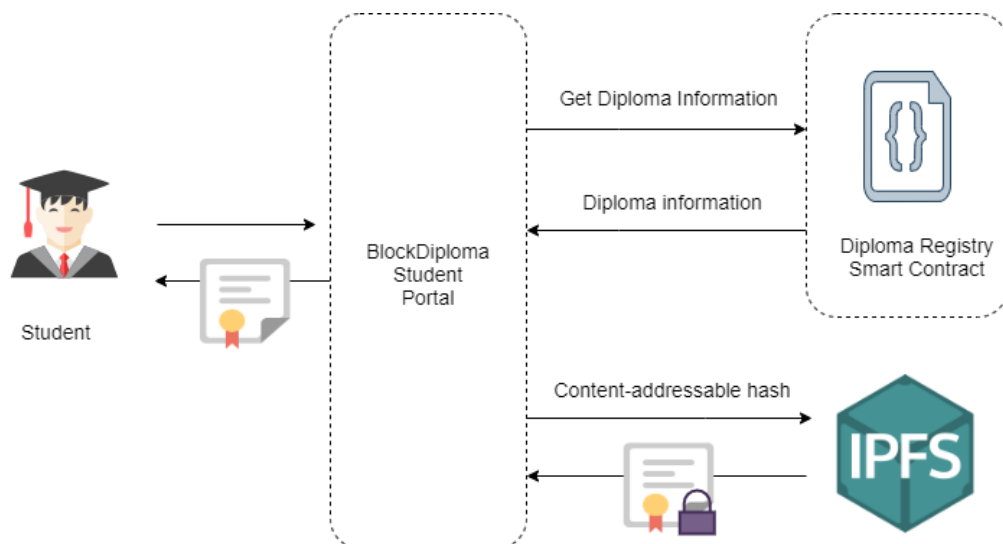


Figure 25: Student Portal inner workings.

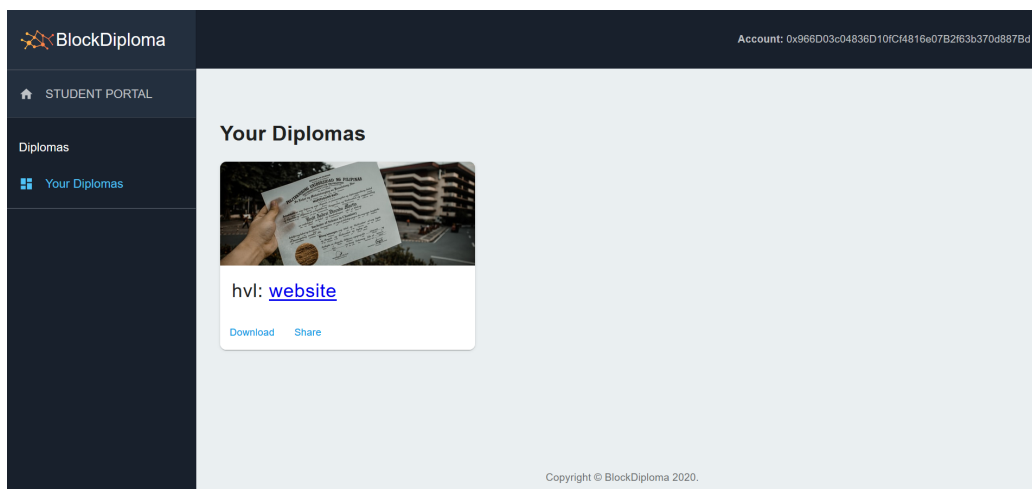


Figure 26: Screenshot of Student Portal.

Sharing diplomas is one of the key features of the current Diploma registry, BlockDiploma provides a way of sharing diplomas too. Diplomas can either be shared unencrypted or encrypted using the private sharing mechanism of BlockDiploma. BlockDiploma has no central server capable of sending email, therefore the student must share the results himself. The private sharing uses asymmetric encryption using the Crypto-JS library and the RSA algorithm. The private sharing option requires that the student obtain the public key of the employer and then gives that key to BlockDiploma. After that BlockDiploma will encrypt the diploma using that key, and upload the encrypted diploma to IPFS, the student will be presented with a link he can share with the employer. The employer decrypts the diploma and verifies it by checking against the verification page on BlockDiploma.

Using public key encryption, one could argue that this is not the way to go considering a company like car dealership might not be competent or willing to use such a solution. Therefore, sending the diploma the old-fashioned way without using private sharing is possible. The student can download his diploma and email it to the company representative or print it out and mail it. The mailing solution sends us right back at challenge number one with verification of diplomas. Perhaps, a better solution would be some sort of cryptographic application that handles the decryption for the recipient. A concept already used by BCDiploma.

### 6.4.6 University Portal

Higher education providers are given their own portal for performing operation in BlockDiploma such as issuing, revoking, and verifying diplomas. The portal is meant to serve as a user interface for the university smart contracts that comply with the standard interface for university smart contracts. The rationale behind this extra smart contract adding complexity is to allow universities to adapt how they want their contracts to interact with their administration software.

The university smart contract has an owning address with extra privileges, think of it like an administrator account. That administrator account can add other addresses to a mapping of trusted employees, that mapping lead to a struct where the trusted status can be checked and what employee identifier code is connected to that address. The employer identifier code is used for privacy reason since this information is stored in the smart contracts, but the university would be able to identify employees using that identifier.

```
1
2     /// You should implement this as a mapping.
3     struct TrustedEmployee{
4         bool trusted;
5         string employee_identifier;
6     }
7
8     /// @notice mapping of trusted employes
9     mapping(address => TrustedEmployee) trusted_employees;
```

Listing 2: Trusted employee mapping

Addresses added to the trusted employee list can act on behalf of the university, allowing them to issue, verify and revoke diplomas. The trusted employee list serves two purposes: (1) The university no longer has to use their administrative address by default, keeping the use of the administrative account to a minimum, (2) they can give different addresses to employees making it possible to track who did what. Permission is given by a transaction containing the Ethereum address and employee identifier of the employee being given permission to access the contract.

The ability to act on behalf of the university regarding diplomas must be protected, and therefore a university can at any time remove the permission by sending another transaction from the administrator address. The transaction will change the boolean value, trusted, from true to false, preventing that address from acting on behalf of the university in the future. This should always be done when employee has resigned.

The administrative account must also be protected and therefore there is a transfer of ownership mechanism in the university smart contract. The transfer functionality is an implementation of the ERC-173 Ownership standard. The ERC-173 interface can be viewed in appendix A. Transferring the ownership to another account should happen every time one suspects that the key has been exposed or an employee with access leaves their job.

If an employee with access to diplomas are misbehaving, then it can be discovered and proved by auditing the logs. Every time a diploma is issued or revoked an event is triggered. The event contains the hash of the diploma, the employee identifier, and the student ID. When permissions are given or removed from university staff an event is emitted. The logs can be viewed and inspected in the BlockDiploma DApp. These logs can always be inspected to discover discrepancies, BlockDiploma has a frontend solution for searching and inspecting the logged events.

Higher education providers can terminate their contract using the implemented self-destruct functions that terminates the smart contract. Terminating the smart contract means that the higher education provider no longer can issue or revoke diplomas from the diploma registry contract. Diplomas already issued remains issued so that they can be verified, the information stored about the university in the university registry is kept for verification purposes.

#### **6.4.7 Fighting Diploma Mills**

Keeping fraudulent diplomas out of the diploma registry smart contract is vital for it to be a trusted diploma verification source, therefore the university registry smart contract was created. The purpose of the contract is to be a trusted source of reputable higher education providers for the diploma registry smart contract. The owner of that contract must determine whether the higher education provider is reputable or not prior to adding their smart contracts address to the registry.

Figure 27 illustrates how the cooperation between the university registry smart contract and the diploma registry smart contract prevents diploma mills from issuing fake diplomas to our trusted diploma verification source.

The address of the contract belonging to a higher education provider is stored in the university registry contract alongside its status, name, and website. The meta-information can be adapted later to suit the needs of the final system.

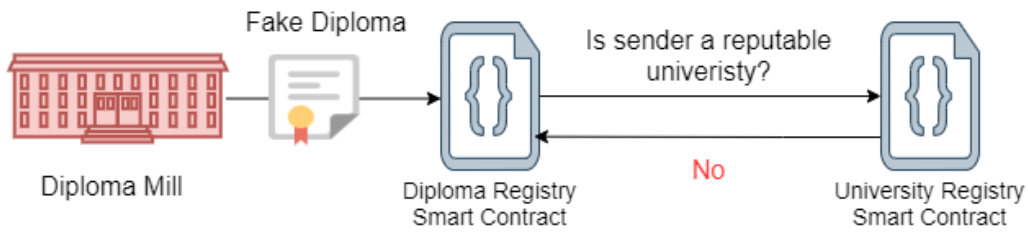


Figure 27: How diploma mills are denied access to the diploma registry smart contract.

Higher education providers can be added and removed by the owner of the university registry smart contract through transactions. When adding a provider, the owner must provide the required metainformation and sign the transaction. If the owner of the registry wants to remove a university he can search for their name, website or address using the BlockDiploma DApp and click the delete icon, which will trigger a transaction changing the valid status from true to false. The user interface for removing a university from the registry can be seen in figure 28. Providers with a valid status of false will be denied when attempting to perform transactions against the diploma registry smart contract.

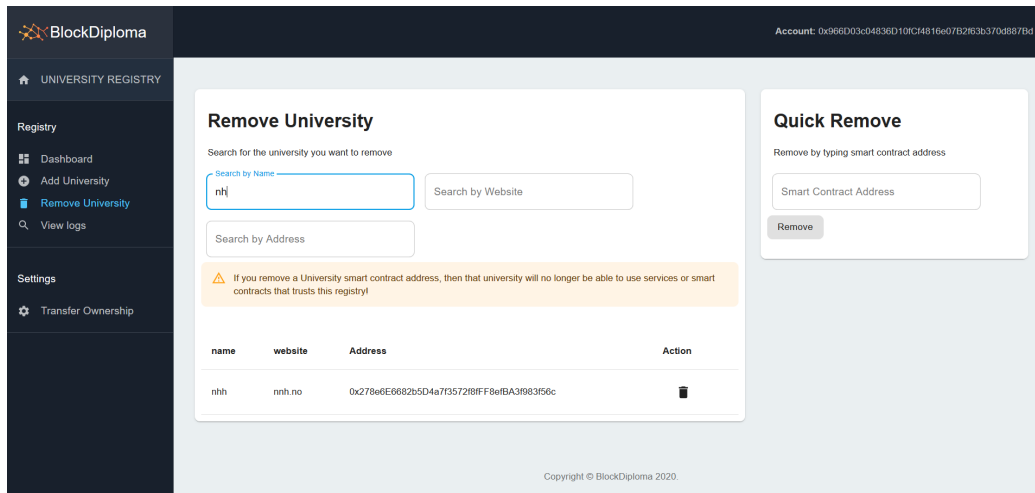


Figure 28: Screenshot of the form used to remove higher education providers from the university registry smart contract.

### **6.4.8 Customizing the solution**

The university registry contract will remain under the control of a central authority. The diploma registry smart contract is not supposed to be controlled by a single entity making sure it will always be available. The university contract has an interface implementation allowing anyone to create their own version fitting their specific needs. University contracts should comply with the University Standard interface (UniversityStandard.sol).

### **6.4.9 Implementation of user interface requirements**

The consistent and user-friendly requirement is resolved using ReactJS together with Material UI to create a web interface familiar to users. Wallets could be an issue for the user-friendliness of the user interface, making it difficult for some users to use the solution regardless of how the web interfaces is coded.

The second user interface requirement stated that the solution must comply with legal requirements for accessibilities and the Web Content Accessibility Guidelines 2.0 (WCAG 2.0). Since BlockDiploma is an explorative proof-of-concept we have disregarded this requirement. A finished diploma system would need to comply with the accessibility requirements. One challenge regarding accessibility could arise when requiring the use of wallet software. Understanding gas and addresses could also make it difficult to comply fully with the WCAG 2.0.

The third and last user interface requirement stated that the different user groups should have their own frontend/user interface that allows them to get their use cases done. This has been solved by creating a public site for verification, a university portal for higher education providers, a student portal for students and a portal for quality assurance agencies. Screenshots of the BlockDiploma user interfaces can be viewed in Appendix B.

### **6.4.10 Implementation of non-functional requirements (NFRs)**

GDPR-compliance was the first of the non-functional requirements. Since blockchain is an immutable chain of blocks containing information it is hard to rectify or delete data. Data stored on the blockchain is limited to the hash of the diploma, anonymous addresses, and pseudo-anonymous identifier. Pseudo-anonymous identifiers includes student and employee identification

codes. If the university keeps the identification codes secret, then anonymity is ensured. Storing only the hash of the diploma ensures that the diploma can be verified but there is no way of going from the hash to diploma thus keeping it private.

Verifiers attempting to verify your diploma will be able to find out your student ID and connect the Ethereum address used to your name. The address used for connecting to the student portal should therefore not be used in other types of transactions for privacy reasons. The diploma is stored encrypted on IPFS to prevent prying eyes from viewing diplomas. The encryption used has not been tested and audited enough to be considered safe, this should be viewed as a proof-of-concept. Total GDPR-compliance cannot be guaranteed and should be reviewed before BlockDiploma can be used outside of this thesis.

Confidentiality, integrity, and availability are security properties related to the next three requirements. NFR-2 states that the confidentiality of the diplomas must be ensured. The confidentiality is ensured by encrypting the diploma before uploading it to IPFS. Storing only the hash on the blockchain ensures confidentiality by the fact that there is no way of going from hash to diploma.

Integrity of the diploma is covered by NFR-3 and this is enforced using a cryptographic hash algorithm on the diploma and storing the hash on-chain. Verifying a diploma is as simple as computing the hash of the diploma and ask the smart contract whether that diploma is valid or not. Using a cryptographic hash algorithm, we make it impossible to tamper with the file without breaking the integrity of the file resulting in a different hash.

Availability can be achieved by hosting the smart contracts on the Ethereum blockchain and the frontend on IPFS. Decentralized approaches protect against denial of service attack when enough nodes have the data. In the theoretical foundation we explained ENS and Swarm which would be the alternative to IPFS and a regular domain. The rationale behind going for IPFS is that Infura, the service used to deploy our smart contracts, offers an easy gateway to use with IPFS.

Keeping the costs down was NFR-4 and this is achieved by storing a minimum of information. Transactions fees are what we pay for keeping the Ethereum blockchain secure, therefore transactions against the smart contracts will have a fee. The programming itself is done carefully to avoid operations that requires a lot of gas (money).

Disaster recovery with BlockDiploma is simple, it is already there by the

choice of technology. BlockDiploma uses blockchain technology and peer-to-peer storage to ensure integrity and availability. Transactions against the smart contracts are atomic meaning all or nothing, if an error occurs everything is reverted to the original state. Peer-to-peer storage ensures that the service can be recovered as long as one node has the data. Availability of the service are higher and better for every node in possession of the data.

NFR-6 states that universities should be able to integrate BlockDiploma to their websites and diploma management systems. This requirement is not resolved. Deploying a smart contract is the first step in using BlockDiploma. Deploying a smart contract is a simple Ethereum transaction, but frontend software is not ready to be easily installed.

Storing and backing up diplomas in a secure way is achieved using IPFS. The diplomas are encrypted with a symmetric encryption key. The handling of the encryption keys is not sufficiently reviewed to be used in real life. CheckDiploma and Diploma.report uses centralized storage platforms for their diploma backups, but since BlockDiploma attempts to fully decentralize the Diploma registry we are going for decentralized storage.

BlockDiploma should be possible to use in all EU countries as stated in NFR-9. Every country in Europe following the diploma supplement and ECTS convention should be capable of using BlockDiploma. The implemented solution allows anyone issuing diploma in PDF-format to use the solution. The challenge here is whether the law in different countries allows the use of the solution or not, but that is out of the scope for this thesis.

The last non-functional requirement is the logging of all actions regarding diplomas. Every time a diploma is issued or revoked an event is emitted. The event can then be found recorded on the blockchain allowing anyone to audit diploma actions.

#### **6.4.11 Summary of implemented requirements**

The tables 7, 8, and 9 below show the status of all the requirements found in this thesis. Implemented means that the solution is ready for real life testing. Partial means further analysis is required. Incomplete means that it is not started on or does not yet have a viable solution.



ID	Description	Category	Status
<b>NFR-1</b>	BlockDiploma must be GDPR-compliant.	Must have	Partially
<b>NFR-2</b>	Confidentiality of the diplomas must be guaranteed.	Must have	Partially
<b>NFR-3</b>	Diploma integrity should be enforced to prevent falsification.	Must have	Complete
<b>NFR-4</b>	BlockDiploma needs to always be available.	Must have	Incomplete*
<b>NFR-5</b>	Costs of operating the system should be kept to a minimum.	Wish list	Partially
<b>NFR-6</b>	Disaster recovery should be easy or unnecessary.	Should have	Complete
<b>NFR-7</b>	BlockDiploma should be easy to integrate to universities own website and their diploma management system.	Could have	Incomplete
<b>NFR-8</b>	Diplomas should be stored and backed up in a secure way.	Should have	Partially
<b>NFR-9</b>	BlockDiploma should be possible to use in all EU countries.	Wish list	Complete
<b>NFR-10</b>	All diploma actions should be logged.	Should have	Partially

\* BlockDiploma is currently not hosted anywhere, this task is intended to be solved using IPFS.

Table 7: Status of Non-functional Requirements.

ID	Description	Category	Status
<b>UI-1</b>	BlockDiploma must have a consistent and user-friendly interface.	Must have	Partially
<b>UI-2</b>	Solution must comply with legal requirements for accessibilities and comply with the "Web Content Accessibility Guidelines 2.0 (WCAG 2.0).	Must have	Incomplete
<b>UI-3</b>	The user groups should have their own frontend/user interface allowing them to get their use-cases done.	Must have	Complete

Table 8: Status of UI-requirements.

ID	Description	Category	Status
FR-1	Higher education providers should be able to issue diplomas from already existing files.	Must have	Complete
FR-2	Higher education providers should be able to issue diplomas using automated systems.	Should have	Incomplete/ Partially
FR-3	Diplomas should be issued in PDF-format.	Must have	Complete
FR-4	Diploma supplement should be generated by the help of our solution and added to the diploma file.	Should have	Partially
FR-5	Diplomas must be possible to revoke and there should be a confirmation mechanism before the revocation is performed.	Must have	Complete
FR-6	Diplomas should be easy to verify through a user interface.	Must have	Complete
FR-7	Student should be able to retrieve their diploma.	Should have	Complete
FR-8	Students should be able to share their diplomas.	Should have	Complete
FR-9	Higher education provider administrator should be able to give permissions to an employee account.	Must have	Complete
FR-10	Higher education provider administrator should be able to remove permissions from an employee account.	Must have	Complete
FR-11	Higher education provider administrator should be able transfer the administrative privileges.	Must have	Complete
FR-12	Higher education provider administrator should be able to inspect logs of all actions related to diplomas.	Must have	Complete
FR-13	Higher education providers should be able stop using our solution, but data needed for validating already issued diplomas should be kept.	Should have	Complete
FR-14	An authority should be able to add a university to the registry.	Must have	Complete
FR-15	An authority should be able to remove a university from the registry.	Must have	Complete

Table 9: Status of Functional Requirements.

## 7 Security Assessment

BlockDiploma must be secure for higher education providers to be able to use the solution for administration of diplomas. In this chapter BlockDiploma will be reviewed against commonly known security pitfalls. Considering the components used in BlockDiploma we have smart contracts serving as a backend, a frontend using standard web technologies and IPFS for storage.

### 7.1 Smart Contract Security Assessment

The Decentralized Application Security Project (DASP) is an initiative by a group called NCC Group. DASP is an open and collaborative project with the goal of discovering smart contracts vulnerabilities [73]. The list of top 10 security vulnerabilities in smart contracts as of 2018 is [73]:

1. *Reentrancy*
2. *Access Control*
3. *Arithmetic*
4. *Unchecked Low Level Calls*
5. *Denial of Services*
6. *Bad Randomness*
7. *Front Running*
8. *Time Manipulation*
9. *Short Addresses*
10. *Unknown Unknowns*

#### 7.1.1 Reentrancy

*”Reentrancy occurs when external contract calls are allowed to make new calls to the calling contract before the initial execution is complete. For a function, this means that the contract state may change in the middle of its execution as a result of a call to an untrusted contract or the use of a low level function with an external address.” [73]*

BlockDiploma mitigates this by only calling trusted contracts. The Diploma Registry contract only calls the University Registry contract to check if a sender is authorized. The University registry contract is the absolute authority what contracts are authorized. Transactions from other addresses will be disregarded. The contracts in the BlockDiploma system do not contain any funds that an attacker could try to steal.

### 7.1.2 Access Control

*“While insecure visibility settings give attackers straightforward ways to access a contract’s private values or logic, access control bypasses are sometimes more subtle. These vulnerabilities can occur when contracts use the deprecated tx.origin to validate callers, handle large authorization logic with lengthy require and make reckless use of delegatecall in proxy libraries or proxy contracts.” [73]*

The deprecated tx.origin is not used in BlockDiploma, nor the use of delegatecall. Diploma Registry contract uses require statement with a call to the university contract, but this appears to work as intended. Visibility settings are deliberately set as they are in the contracts, with clear access modifiers enforcing the security. The use of constructors to set the important variables like owner and contract addresses ensures that an attacker cannot simply call that function again to take control over the contract.

### 7.1.3 Arithmetic

*“Integer overflows and underflows are not a new class of vulnerability, but they are especially dangerous in smart contracts, where unsigned integers are prevalent and most developers are used to simple int types (which are often just signed integers). If overflows occur, many benign-seeming codepaths become vectors for theft or denial of service.” [73]*

BlockDiploma does not use integers and are therefore not vulnerable.

### 7.1.4 Unchecked Low Level Calls

*One of the deeper features of Solidity are the low level functions call(), callcode(), delegatecall() and send(). Their behavior in accounting for errors is quite different from other Solidity functions, as they will not propagate (or*

*bubble up) and will not lead to a total reversion of the current execution. Instead, they will return a boolean value set to false, and the code will continue to run. This can surprise developers and, if the return value of such low-level calls are not checked, can lead to fail-opens and other unwanted outcomes. Remember, send can fail! [73]*

BlockDiploma never uses `call()`, `callcode()` or `delegateCall()` and is therefore not vulnerable to this type of attack.

### **7.1.5 Denial of Services**

*”Denial of service is deadly in the world of Ethereum: while other types of applications can eventually recover, smart contracts can be taken offline forever by just one of these attacks. Many ways lead to denials of service, including maliciously behaving when being the recipient of a transaction, artificially increasing the gas necessary to compute a function, abusing access controls to access private components of smart contracts, taking advantage of mixups and negligence, etc. This class of attack includes many different variants and will probably see a lot of development in the years to come.” [73]*

BlockDiploma has no known denial of service attack weaknesses. This is critical and should be further investigated before such the system can be used. BlockDiploma is attempting to solve the challenge regarding availability and this is one of the main threats against that goal.

### **7.1.6 Bad Randomness**

*”Randomness is hard to get right in Ethereum. While Solidity offers functions and variables that can access apparently hard-to-predict values, they are generally either more public than they seem or subject to miners’ influence. Because these sources of randomness are to an extent predictable, malicious users can generally replicate it and attack the function relying on its unpredictability.” [73]*

BlockDiploma does not use randomness and this is therefore not a threat.

### **7.1.7 Front Running**

*Since miners always get rewarded via gas fees for running code on behalf of externally owned addresses (EOA), users can specify higher fees to have their*

*transactions mined more quickly. Since the Ethereum blockchain is public, everyone can see the contents of others' pending transactions. This means if a given user is revealing the solution to a puzzle or other valuable secret, a malicious user can steal the solution and copy their transaction with higher fees to preempt the original solution. [73]*

BlockDiploma requires all transactions to come from known parties, the rest of the transactions are disregarded. Front-running attack would not succeed against BlockDiploma.

### **7.1.8 Time Manipulation**

*"From locking a token sale to unlocking funds at a specific time for a game, contracts sometimes need to rely on the current time. This is usually done via `block.timestamp` or its alias `now` in Solidity. But where does that value come from? From the miners! Because a transaction's miner has leeway in reporting the time at which the mining occurred, good smart contracts will avoid relying strongly on the time advertised. Note that `block.timestamp` is also sometimes (mis)used in the generation of random numbers as is discussed in *Bad Randomness*." [73]*

Not an issue for BlockDiploma.

### **7.1.9 Short Addresses**

*"Short address attacks are a side-effect of the EVM itself accepting incorrectly padded arguments. Attackers can exploit this by using specially-crafted addresses to make poorly coded clients encode arguments incorrectly before including them in transactions. Is this an EVM issue or a client issue? Should it be fixed in smart contracts instead? While everyone has a different opinion, the fact is that a great deal of ether could be directly impacted by this issue. While this vulnerability has yet to be exploited in the wild, it is a good demonstration of problems arising from the interaction between clients and the Ethereum blockchain. Other off-chain issues exist: an important one is the Ethereum ecosystem's deep trust in specific Javascript front ends, browser plugins and public nodes. An infamous off-chain exploit was used in the hack of the Coindash ICO that modified the company's Ethereum address on their webpage to trick participants into sending ethers to the attacker's address." [73]*

No mitigation for this issue is implemented. BlockDiploma could be vulnerable.

### 7.1.10 Unknown Unknowns

*”Ethereum is still in its infancy. The main language used to develop smart contracts, Solidity, has yet to reach a stable version and the ecosystem’s tools are still experimental. Some of the most damaging smart contract vulnerabilities surprised everyone, and there is no reason to believe there will not be another one that will be equally unexpected or equally destructive. As long as investors decide to place large amounts of money on complex but lightly-audited code, we will continue to see new discoveries leading to dire consequences. Methods of formally verifying smart contracts are not yet mature, but they seem to hold great promise as ways past today’s shaky status quo. As new classes of vulnerabilities continue to be found, developers will need to stay on their feet, and new tools will need to be developed to find them before the bad guys do.” [73]*

It goes without saying, BlockDiploma is not secured against unknown attacks and weaknesses. To be fair neither are any other known systems.

### 7.1.11 Summary

In table 10 an overview of the threats and the mitigated is shown.

	Threat	Mitigated
1	Reentrancy	Yes
2	Access Control	Yes
3	Arithmetic	Yes
4	Unchecked Low-Level Calls	Yes
5	Denial of Services	Yes*
6	Bad Randomness	Yes
7	Front Running	Yes
8	Time Manipulation	Yes
9	Short Addresses	No
10	Unknown Unknowns	No

\* No currently known vulnerabilities

Table 10: Status of DASP Top 10 Threats Handling.

## 7.2 Frontend Security Assessment

The Open Web Application Security Project (OWASP) is a community effort led by The OWASP Foundation with the goal of improving web application security [74]. OWASP Top 10 is an awareness documents for developers of web applications. The document has a broad community consensus covering the most critical security risks for web applications [74]. OWASP Top 10 is used to assess the security of our frontend that has similarities with a standard web application, but not all threats are equally relevant for BlockDiploma since there is no traditional server side. As of 2017 the main risk to web applications were [74]:

1. *Injection.*
2. *Broken Authentication.*
3. *Sensitive Data Exposure.*
4. *XML External Entities (XXE).*
5. *Broken Access Control.*
6. *Security Misconfiguration.*
7. *Cross-Site Scripting XSS.*
8. *Insecure Deserialization.*
9. *Using Components with Known Vulnerabilities.*
10. *Insufficient Logging and Monitoring.*

### 7.2.1 Injection

*"Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization" [74].*

BlockDiploma uses no database queries and all stored data is read from the smart contracts through the DApp. Input in the BlockDiploma DApp is digitally signed and tampering with the data after signing will invalidate the signature. The BlockDiploma is therefore not vulnerable to input tampering that would execute unintended commands with the smart contracts, but



BlockDiploma is vulnerable to Ethereum address injection that could result in an attacker obtaining private information.

### 7.2.2 Broken Authentication

*"Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently" [74].*

Changing data in the smart contracts requires a valid digital signature from the wallet of the authorized user. This prevents the misuse of someone credentials if they have protected their wallet correctly. Bypassing the authentication of BlockDiploma requires an attacker to be capable of injecting the address of an authenticated user (this information can easily be found on the blockchain), but to make any changes or damage he would still need a valid digital signature. An attack like this could expose a diploma to the attacker.

### 7.2.3 Sensitive Data Exposure

*"Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser" [74].*

BlockDiploma itself does not store any sensitive data. If a university keeps student and employee identifier secret and secure it should not be possible to identify the owner of the diploma. If an attacker can bypass the authentication in react router then BlockDiploma will decrypt the diploma, this could be done by injecting the students Ethereum address into BlockDiploma. A decrypted diploma can contain sensitive data.

### 7.2.4 XML External Entities (XXE)

*"Many older or poorly configured XML processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks" [74].*

BlockDiploma does not handle XML-files and is therefore not exposed to this threat.

### 7.2.5 Broken Access Control

*"Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc" [74].*

Access is enforced by the blockchain and the smart contracts. Without a valid digital signature unauthorized changes are not possible. The frontend implemented in React checks for permissions and will show an unauthorized page to unauthorized users. The unauthorized page is more user-friendly than secure. An attacker could bypass the access control in the react app by sending in the public address in the same way a wallet would do.

### 7.2.6 Security Misconfiguration

*"Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched/upgraded in a timely fashion" [74].*

All error messages in the BlockDiploma is generic preventing an attacker from getting information that would help in his or hers endeavor. Currently there are no web server running the system, therefore that part is not evaluated.

### 7.2.7 Cross-Site Scripting XSS

*"XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites" [74].*

The frontend of BlockDiploma is developed using React. React is secure against XSS by escaping any values embedded in the JSX (syntax extension to JavaScript) [75].

### 7.2.8 Insecure Deserialization

*"Insecure deserialization often leads to remote code execution. Even if deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks" [74].*

Privilege escalation attacks is not possible without being able to create a valid digital signature from the owning account. The smart contracts are configured to deny others than the owner to perform tasks requiring extra privileges. BlockDiploma avoid using deserialization, but this might not be the case anymore if the automated system requirement is implemented.

### 7.2.9 Using Components with Known Vulnerabilities

*"Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts" [74].*

All components and libraries used in BlockDiploma has not been properly audited. We know that there are some issues with a some of the libraries used. Those issues must be addressed before BlockDiploma can be used in real life.

### 7.2.10 Insufficient Logging and Monitoring

*"Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring" [74].*

All smart contracts actions of importance are logged by the help of events. The web application has no logging except this, but in a production environment there should be logging set up at the server running the solution if a server is used.

### 7.2.11 Summary

In table 11 an overview of the threats and the mitigated is shown.

	Threat	Mitigated
<b>1</b>	Injection	Partially*
<b>2</b>	Broken Authentication	Partially*
<b>3</b>	Sensitive Data Exposure	Partially*
<b>4</b>	XML External Entities (XXE)	Yes
<b>5</b>	Broken Access Control.	Partially*
<b>6</b>	Security Misconfiguration	Partially**
<b>7</b>	Cross-Site Scripting XSS	Yes
<b>8</b>	Insecure Deserialization	Yes
<b>9</b>	Using Components with Known Vulnerabilities	No
<b>10</b>	Insufficient Logging and Monitoring	Yes

\* *Wallet authentication vulnerability*

\*\* *Handling of encryption keys is an issue*

Table 11: Status of OWASP Top 10 Threats Handling.

### **7.3 Storage and communication Security Assessment**

Diplomas stored on IPFS are encrypted using symmetric encryption. All diplomas are encrypted prior to upload and decrypted after they have been retrieved. This in theory ensures the confidentiality and integrity of the diplomas when stored on IPFS. However, the size of the key used is too small and handling of the key is not secure. The process of handling encryption keys might need some sort of centralization to be properly secured. Since BlockDiploma is a proof-of-concept and not a market ready product this is not handled, but a final solution would need to handle this.

### **7.4 Other known vulnerabilities and weaknesses**

If private keys or wallet backup phrases are not stored in a secure manner, then there is a risk of a hacker or employee stealing the private keys. Private keys and wallet backup phrases could be exposed by incompetent users because of poor user-friendliness when dealing with private keys and wallets. The private keys can later be misused by a disloyal employee or a malicious attacker thus making the diploma system insecure. Access to a higher education provider's private keys for issuing fake diplomas could become a black-market business. Removing the leaked account from the system or transferring the ownership is the only defensive technique available to the higher education providers in BlockDiploma.

Should the transfer mechanism be used to steal control of the smart contract, the only option for the higher education provider is to reach out to the authority controlling the university registry contract and have their contract removed. If a fake diploma has been issued prior to the contract being removed then there is no way to revoke that diploma in BlockDiploma. A potential fail safe could be to give the owner of the university registry contract access to revoke diplomas, but this would mean that the authority controlling the university registry can revoke a diploma from a university without their permission opening the door for another type of misuse.

The transfer option has been added to allow control of the contract to be transferred from one account to another for cases such as employees quitting their jobs, so that the university can make sure ex-employees do not have the current private keys. One could argue that this is poor design since it allows an attacker with access to that private key to steal control of the contract. On the other side, one could argue that is exactly why this option is needed.

Perhaps, the best way is to create some system or mechanism that never allows employees to have direct access to the private keys.

Authorized universities in the university registry could override a diploma if the diploma they attempt to issue has the same hash. If the university goes through with the override, ignoring a warning and confirmation dialog, the meta-information about the diploma is changed meaning that the student and his university no longer are the owner on record. A malicious university or staff member wishing to prevent a student from another university to have it his diploma verified could succeed if they have an original PDF-version of that diploma. The attack is simple: (1) the university reissues the diploma, and is now the owner of the diploma, (2) the university revokes the diploma using the BlockDiploma DApp. This kind of attack is unlikely for several reasons, the first being that hash collisions are unlikely and the second that a malicious reputable university in the university registry is unlikely. This kind of attack could be discovered by auditing logs and a reputable university discovered in performing this action would face sanctions. If a successful attempt at stealing control of a higher education providers contract succeeds then this attack is more likely to occur. This attack should in the future, be prevented by BlockDiploma regardless of how unlikely the attack is.

## 7.5 Summary

We have seen that there are security issues threatening the BlockDiploma system. The approach taken to address the security of our solution has been to review the OWASP Top 10 and the DASP Top 10 and the results are not devastating, but they are not uplifting either. Further work and investigation on the security of BlockDiploma is necessary before it can be used.

## 8 Analysis and Assessment

We have learned about individuals that falsify their diplomas to get a job, and we have learned about diploma mills and accreditation mills. BlockDiploma has been developed to demonstrate how different aspects can be resolved by decentralizing the Diploma registry. This chapter presents our findings and lessons learned in this process, followed by an assessment of the methodology used and the validity of the results.

The second research question asked if the challenges with present diploma solutions can be resolved by decentralizing the Diploma registry using blockchain and peer-to-peer technology. First step on the way was to survey relevant work, the information gathered were presented in the relevant work chapter. After reviewing several solutions and proposals we decided to create our own proof-of-concept to experiment and explore solutions using blockchain and decentralized technology.

### 8.1 BlockDiploma – a Decentralized Diploma System

Using the relevant work as inspiration we started to formulate requirements for a decentralized diploma system. Based upon the requirements we developed our proof-of-concept used to figure out whether the Diploma registry can be decentralized and resolve the identified challenges.

#### 8.1.1 Stakeholders and requirements

Stakeholders were identified using common knowledge. Students, higher education providers and validators (employers) was identified as stakeholders by using common knowledge. EBP has a progress report that mentions several roles involved, and they could be considered stakeholders too [39]. Stakeholders are important since they are the users of the solution and the people such a solution affects. Performing surveys and workshops with stakeholders are a time-consuming task and was not prioritized in favor of explorative efforts into building a decentralized diploma system. Following our research, stakeholders could be consulted with the knowledge of what is possible and not, prior to developing a complete solution for real world use.

Despite not involving stakeholders we believe that the requirements are solid. Discovering and formulating requirements has been done using common knowledge, legal documents and by studying existing solutions. In addition, we

have used the discovery of the challenges with current solutions to formulate some of the requirements. The control questions mentioned earlier in the requirements subchapter was also important in keeping the requirements relevant. Remember that BlockDiploma is an explorative proof-of-concept and not a market ready product. Originally, the plan was to test BlockDiploma on users group including students, university staff and validators, due to the Covid-19 outbreak this plan changed.

### 8.1.2 BlockDiploma and the challenges with the present diploma solutions

One of the earliest chapters identified challenges with the present diploma systems, in this section we revisit them and our proposed solutions to those challenges:

**Challenge 1: Paper-based versions are time-consuming and difficult to verify.** We believe that a digital solution is the correct path to take when it comes to diplomas. There exist techniques that can be used with computers to verify digital versions with more ease than the paper version. The first challenge can be resolved using digital diploma systems regardless of being centralized or decentralized.

**Challenge 2: Verification requires a recipient to receive an email, emails can be spoofed.** Digital solutions sending emails containing links train users to trust emails. In general, we believe that sending emails like this is not best practices because of the possibility of spoofing attacks. Validators should visit the verification source directly. Doing it that way will protect against spoofed emails and fake websites appearing to be legitimate. Therefore, we propose that a diploma is sent the traditional way from student to the employer, and the employer opens the public site of the BlockDiploma DApp to validate the authenticity of the diploma. Employers opening the DApp directly instead of through a link could help prevent an employer from using a fake DApp.

Bypassing the student and sending the diploma directly to the recipient to avoid tampering as demonstrated in figure 29 is an understandable tactic. In fact, so understandable that we started exploring a similar solution for the BlockDiploma student portal. After many attempts making a decentralized sharing-solution without sending email and setting up centralized servers we



concluded that it is possible but not necessarily user friendly and that it would require an extra validation step.

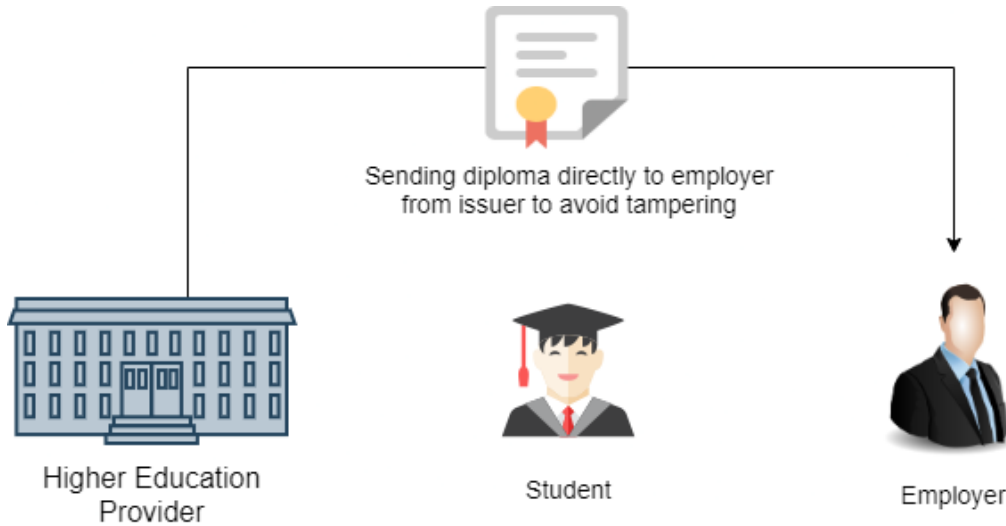


Figure 29: Diplomas sent directly to employer to prevent fraud.

Diploma should be shared by sending the diploma to the recipient either as an unencrypted file, or through the private sharing mechanism. Students share their diploma by providing the public key of the recipient. BlockDiploma uses the public key to encrypt the diploma prior to uploading it to IPFS. After the upload is completed, a shareable link is presented to the student, which again can share it with an employer. The employer decrypts the diploma using his private key. Does this sharing mechanism prevent diploma tampering by students? Unfortunately, the answer is no. A student can download the diploma, modify it, encrypt it, and upload it to IPFS before sending a new link appearing to be real. If the employer performs the verification through the BlockDiploma as intended then this is not a problem. Our proposed solution is therefore to still force the employer to visit the BlockDiploma DApp for verification. The private sharing is simply sharing, without a verification mechanism. To avoid confusion BlockDiploma does not offer to decrypt the diplomas for a validator. BlockDiploma private sharing simply allows students to share their diploma using asymmetric encryption for privacy reasons.

Challenge 2 can be resolved using a decentralized approach without using email, but not using email can affect the user-friendliness.

**Challenge 3: Ensuring the availability of the service, diplomas, and verification mechanism.** Hosting the BlockDiploma DApp and diplomas on decentralized storage such as IPFS or Swarm should increase availability. Centralized and decentralized storage could be combined since at the time of writing decentralized storage might be slow. Centralized hosting could ensure speed and performance when available, and if the servers goes down the decentralized hosted version would still be available. Decentralized storage availability increases for every node holding a copy of the service and the diplomas. Denial of service attacks could take down the centralized hosting, but the decentralized hosting with enough nodes holding a copy should be more resistant. The decentralized version can be accessed through an IPFS-gateway using the content-addressable hash, the drawback being that this is not user-friendly. If the BlockDiploma service cannot be found on centralized servers or decentralized storage, then it is time for plan B. Plan B is using a service like Remix IDE to send a call to the diploma registry smart contract. Diplomas hashed and sent in a query to the smart contract can be verified regardless of the availability of the BlockDiploma DApp. One possible downside with IPFS and Swarm is speed and the potential future costs for storing files.

**Challenge 4: Verifying that a diploma in PDF-format is authentic.** Never trust the diploma, it should always be verified. Using a cryptographic hash algorithm to create a hash digest of the diploma we can create a verification mechanism where the correct digest is stored on the blockchain. Since the blockchain is an immutable ledger, the hash stored on-chain cannot be tampered with. Verifying the authenticity of the diplomas is as simple as computing the hash of the diploma and send it to the smart contract that will return true or false. BlockDiploma translates true or false into more user-friendly messages such as “*diploma is valid*” and “*diploma could not be verified, contact issuer*”.

The hash algorithm used must be a cryptographic secure one, providing collision resistance. Decentralized solutions utilizing blockchain technology solves this challenge in a better way than current centralized solutions. Why? Because blockchain technology consist of an immutable chain of blocks holding the hash that proves the integrity of the diploma. That solution is more resilient and more open, allowing easier and secure verification of the authenticity of the diplomas.

**Challenge 5: How can we ensure that the diploma is owned and controlled by the holder, except for cases with legitimate revocation.** Students should not depend on a third party to access or verify his diploma. BlockDiploma stores the diploma on decentralized storage. The verification is performed against smart contracts and can be performed without using the BlockDiploma frontend. The diploma registry contract cannot be terminated, if the Ethereum blockchain endures then so does the contract and the verification mechanism. At no point does the student need to tell BlockDiploma who he wants to share his diploma with, no information such as email address of the employer is necessary. BlockDiploma is a pure JavaScript frontend of the smart contracts and all the information stored can be viewed in the smart contracts. Verification attempts are not logged either to ensure the privacy of the recipient. We believe that BlockDiploma gives the user ownership and full control of his diploma, which is only partially true for the present solutions. However, when using the IPFS backup mechanism the student is depending on BlockDiploma to decrypt the diploma.

**Challenge 6: Verifying diplomas from foreign issuers efficiently and securely.** Foreign diplomas cannot be verified using the Diploma registry. Foreign diplomas must be verified by contacting the issuer in another country, and then check with NOKUT whether the diploma is valid in Norway. BlockDiploma does not solve this directly, but if the EU required all higher education providers to issue their diplomas using standardized blockchain solution then the process is as simple as verifying domestic diplomas. Decentralized solution like BlockDiploma could help verifying foreign academic diplomas, but that would require that the student's country issues diplomas on the blockchain system and that the country he moves to accepts that type of verification. Centralized solutions could also be created to resolve this challenge.

### 8.1.3 BlockDiploma at a larger scale

The EBP progress reports points out that sharing and authenticating diplomas remains a problem when every country use their own blockchain solution [39]. Therefore, simply using a blockchain solution is not a complete answer. When different national blockchain solutions exist next to each other without being compatible it would limit the positive effects [39].

We have shown how BlockDiploma can resolve the challenges identified with our present centralized solutions. Earlier in the implementation chapter we

have demonstrated how the Diploma registry could be decentralized regarding issuing, verification, and revocation of diplomas.

Determining if a higher education provider is reputable or not can be hard. Within the EU this requires that one checks with EQAR who the quality assurance agency of that country is and then again checks with that agency whether the higher education provider is reputable or not. The process is illustrated in figure 30. After determining if the university is reputable one could start assessing the candidate.



Figure 30: Process of determining if a higher education provider is reputable.

If we look at BlockDiploma at a larger scale than just whether it can be used to replace the current Diploma registry or not, we could imagine how the entire system could be updated to using decentralized technology. The architecture section earlier explained that we use three different smart contracts: the university registry contract, the diploma registry contract, and the university contract. The purpose of the university registry is to be an authority on what higher education providers are reputable and should be able to interact with the diploma registry contract. The diploma registry was meant to replace the current Diploma registry and the university contract to represent the university.

If using BlockDiploma at a larger scale, EQAR could be the owner of the university registry contract and be an oracle on what quality assurance agencies are legitimate. Remembering from the theoretical foundation we explained an oracle like this: “*Antonopoulos definition of an oracle includes data from sources that cannot be provided trustlessly as there is no independently verifiable objective truth [1, p. 254]. Academic diplomas and government IDs where the data is provided by a fully trusted party are examples of sources included in Antonopoulos and Woods definition [1, p. 254]. In these cases the truth is subjective and could only be changed by appeal to the authority responsible for the information [1, p. 254].*”

EQAR is a fully trusted party and there is no objective truth on who is a reputable quality assurance agency, only EQAR determines that within the

EU. Therefore, EQAR should serve as an oracle here. This would require that the trusted employee functionality be implemented in the university registry contract. For the rest of our case we will use Norway as an example again.

NOKUT is the quality assurance agency for Norway. EQAR could add NOKUT as a trusted quality assurance agency giving them permission to add universities to the university registry contract. NOKUT is here serving as an oracle since they are a fully trusted party and they own the subjective truth on who is a reputable university in Norway. Higher education providers in Norway like the Western Norway University of Applied Sciences (HVL) could then apply to NOKUT to be accredited. NOKUT would determine if they are reputable and add them to the university registry contract. Have you guessed it? Yes, HVL is a fully trusted party and the owner of the subjective truth about diplomas originating from them. HVL is serving as an oracle. Something wrong with your diploma? Then the only option of appeal is to HVL. In figure 31 the process of using BlockDiploma at a large scale is illustrated. The results are a trusted diploma verification source (diploma registry smart contract) and a trusted source of to what higher education provider are reputable and not (university registry smart contract).

## 8.2 Assessment of Ethereum for BlockDiploma

Ethereum has been used for developing BlockDiploma, and in this section we will assess how well this worked for us. Choosing Ethereum over smaller blockchains was a smart play in terms of community driven documentation, tutorials, and answers to common questions on different forums making it easier to troubleshoot and develop our DApp.

From the permissioned blockchain section we learned that there was primarily to types of blockchains: open and closed. Ethereum is an open one where anyone can read the information in the blockchain and this has been an important factor for our solution. One could argue that for a use-case with academic diplomas private blockchains should be used to remain in total control. A key challenge identified in this thesis was that diploma holders should be in control of their diplomas and not a closed organization, therefore an open and transparent blockchain like Ethereum has been a good choice.

Getting the DApp to talk to our smart contracts has been a challenge since some of the newer solidity versions did not work with Drizzle. The community resources available helped finding the fixes and solutions to those issues. A

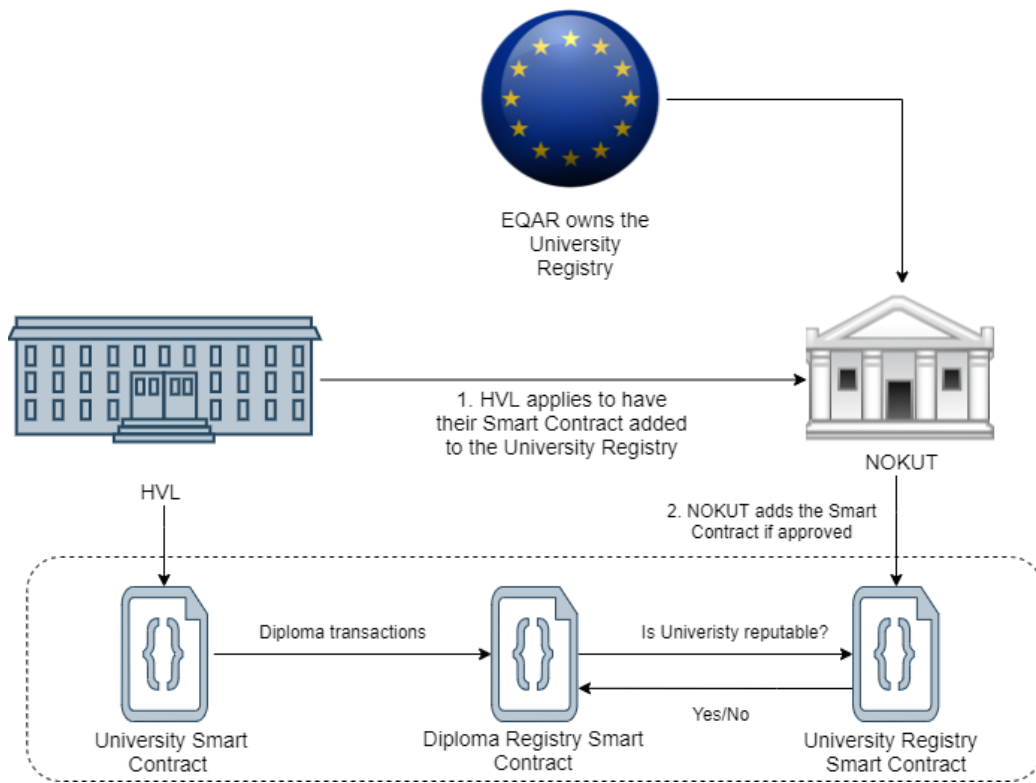


Figure 31: BlockDiploma at a larger scale.

large community is important to learn and resolve errors. Choosing newer or other blockchains with a smaller developer community could make it harder to find answers to issues.

The public site of the BlockDiploma DApp used for verification was supposed to be available to all users with no need for a wallet. Despite verification being a simple read operation, this was quite difficult to accomplish. The difficulties arise because we are not running our own node and need to use an API to access the Ethereum blockchain. When using Infura we are required to provide an Ethereum address and private key. A standardized way of providing an account used by the public for read-only operations would have been great. In fact, we believe that this is quite important for new users allowing them to use a DApp without having to start with a wallet for simple read operations. One option of course is to use a block explorer, but this is not necessarily a user-friendly solution.

There exists a great number of different blockchains and many of them support

smart contracts, among them one could find EOS<sup>11</sup> and TRON<sup>12</sup>. Comparing all the different blockchains against each other is a task too large for the scope of this thesis. The comparison of what blockchain is most suited for different types of DApps could be a research effort on its own.

The conclusion is that we are satisfied with Ethereum as the blockchain used in our solution, but that there are some issues with read-only operations when not operating your own node.

### 8.3 Results and Learning Outcomes

After exploring decentralized solutions through the development of Block-Diploma, we believe that the hypothesis stating that there are challenges with our present solutions are strengthened. We also believe that those challenges can partially be resolved by decentralizing the Diploma registry using blockchain and peer-to-peer technology in the future.

**Legal and regulatory challenges** We have learned about several legal requirements for a diploma system. Web accessibility requirements which are reasonable, other requirements might need regulatory changes. The regulation on the Diploma registry might not be suitable for using blockchain technology. Our first learning outcome is that there might be a need for regulatory changes to be able to use the decentralized systems. The EBP progress report states that the greatest challenge is not technology, but the governance and standards needed to manage and interoperate a complex transnational ecosystem built on the existing solutions [39]. GDPR-compliance must also be ensured.

**Diploma file format** Some existing projects use JSON-files and OpenBadges-specifications, but we believe that the PDF-format should be used. The rationale being that the PDF-format is already widely adopted. We believe that a diploma system should use PDF as the default format, but this is more an opinion than a researched claim. Regardless of the format used, average users should be able to view it without needing a specific software, and a PDF-file is a format that can be read by many different computer programs.

---

<sup>11</sup>EOS and smart contracts <https://eos.io/build-on-eosio/>

<sup>12</sup>Tron and smart contracts [https://developers.tron.network/docs/introduction-](https://developers.tron.network/docs/introduction-1)

**Issuing Diplomas** BlockDiploma at the current stage is created for only handling diplomas and not transcripts. The Diploma registry allows students to choose which parts of their results they want to share. A complete and decentralized Diploma solution should probably have the same capability. One way to solve this is by issuing one diploma file per institution, but what about joint study programs? What about transcripts prior to achieving a complete degree? Often when applying for jobs while still studying, a transcript is shared with the potential employer. BlockDiploma should have this functionality, but that would require that every course upon completion is issued on-chain and combined into a verifiable transcript or diploma upon request. Perhaps BlockDiploma together with the Diploma registry is the solution? The Diploma registry fetches the result from the higher education providers, combines and create the transcript or diploma file before appending that file to BlockDiploma. This would improve the verification process, and issued diplomas are controlled by the student since the Diploma registry is only needed for creating the transcript or diploma.

**Verifying Diplomas** Webpages used for verification should be publicly available and free to use. Existing providers allow users to verify diploma without using wallet software or paying fees. This is necessary for creating a secure and reliable educational background check source. Allowing a party to interact with BlockDiploma without a wallet is hard task to implement in a satisfactory way.

Diplomas should be verified using a hash ensuring that the integrity of the diploma is intact. We believe this to be better than storing the entire diploma on-chain and using encryption for verification.

**Revoking Diplomas** Not all existing solutions has this option, but mistakes happen so we believe that a diploma system should have an option to revoke a diploma. The revocation mechanism must verify that the user wants to revoke the diploma, this simply cannot be allowed to happen by mistake. The method used should not lock up funds or require any centralized list, it should be as simple as a boolean switch.

**Hosting the DApp** Decentralized applications should stay decentralized and this include hosting, however decentralized storage is currently slower than the centralized ones. Therefore, we believe that the best option to ensure performance is to use a centralized server, but for availability and



Plan B the best would be to host such a solution on decentralized storage like IPFS. The decentralized system could be connected to a traditional domain or by using Ethereum Name Services.

**Diploma storage** Diplomas should always be available to the holder of the diploma. Storing diplomas on IPFS or other decentralized storage will ensure the availability if enough nodes has a copy of the file. Some of the projects reviewed used commercial storage platforms like Google Cloud platform and Amazon Cloud. The most important thing is that no matter what happens to the issuer an original digital version can be retrieved; a backup mechanism must exist. Diplomas stored on centralized or decentralized storage must be encrypted to prevent prying eyes from looking at an arbitrary person's diploma. That would probably be a breach of privacy laws and the expectation of confidentiality and privacy.

**Fighting diploma mills** Only reputable universities should have access to the issuing verified diplomas. That requires that some sort of authority keeps a list of reputable universities and prevent diploma mills from entering that list.

**Integration with existing solutions** The number of diplomas issued every year makes it hard for staff to perform every diploma action manually. Manual actions are probably not desired either, making it crucial to have options for integrating a decentralized diploma solution into already existing administration software. No university will use a decentralized solution if that requires extra employees and a large workload handling the diplomas. Decentralized solutions capable of performing most of the diploma issuance automatically is more likely to succeed. The verification mechanism should be integrated on the websites of the universities too.

**Potential pitfalls** There are several potential pitfalls using the new decentralized technology. The first one is user-friendliness; will users be able to use wallets correctly? Some might give up and never try, while others might expose their wallet somehow leaving the door open for hackers. The second pitfall is unforeseen bugs in our smart contracts. Creating secure and reliable smart contracts requires that rigorous development techniques to be applied. The third potential pitfall is mistakes when handling the encryption keys of diplomas, which could result in data exposure.

Despite the potential pitfalls we believe that one day a complete decentralized solution will replace the Diploma registry fully or partially.

**Authentication by Ethereum Address** Authentication in a DApp should enable login by simply using a wallet. In BlockDiploma we authenticate users based upon the address they have in their account, that is not secure enough. A wallet login system should require a digital signature ensuring that the person trying to interact is in fact the owner of the address. If not, an attacker could create a custom wallet and perform an injection attack where the Ethereum address of the student is injected to fool BlockDiploma into decrypting a diploma for the attacker.

## 8.4 Research

### 8.4.1 Own Contribution

We have explored how the Diploma registry can be fully decentralized through our proof-of-concept called BlockDiploma. The idea of using blockchain technology to issue and verify diplomas must be accredited to MIT and the University of Nicosia. Using decentralized storage technology is inspired by the book *Mastering Ethereum* by Antonopoulos and Wood, and various solutions attempting to use decentralized storage.

What is our contribution then? Our contribution has been to review the existing solutions for handling diplomas in Norway. We started by reviewing the traditional way of issuing diplomas using paper-based versions, followed up by reviewing challenges with the current Diploma registry. We identified several challenges with the current solutions for diplomas.

After identifying the challenges with present solutions, we started to gather and aggregate information regarding other relevant work on blockchain-based diploma systems. We reviewed the relevant projects to figure out how they solve the different aspects of being a diploma management system.

Based upon the capabilities of the Diploma registry, the legal requirements found and the other relevant solutions we formulated a set of requirements for a complete blockchain-based diploma system. Based upon our requirements we selected Ethereum as our blockchain based upon the fact that it had already been used for this purpose and is one of the largest public blockchains. We started through the development of BlockDiploma to identify ways of

implementing the requirements in a way that would result in a decentralized diploma system. BlockDiploma can do many of the things needed to become a diploma system, but some work remains before it could be considered a complete diploma system.

Our total contribution is the sum of identifying challenges, formulating requirements, and attempting to implement a system that resolves the identified challenges. Our findings are presented in this chapter and in the conclusion.

### 8.4.2 Validity

Developing our own proof-of-concept could result in researcher bias. Falling in love with our own creation poses a risk to the validity of our research. Remaining unbiased has been focused on during the analysis and assessment. Requirements has been formulated based upon the functionality of the Diploma registry, legal requirements, and relevant work by others. Based upon that work with different sources of information we believe that we ended up with valid requirements. The process of gathering and formulating requirements could have been more credible if a survey collecting information from stakeholders and relevant parties had been performed.

If we had started this research from the beginning again several changes to improve the validity would have been made. BlockDiploma would then be tested on users that could provide feedback on how they experienced the system and if they believed it could handle the job of the Diploma registry. Improvements could then be made, and a second round of user test or acceptance testing could have been performed.

Despite that we would have done some parts differently, we believe that we have a sufficient basis for our findings and conclusion. We believe that the results of our research is valid for several reasons: (1) The amount of blockchain solutions for academic diplomas, (2) the MIT “*What we learned from designing an academic certificates system on the blockchain*” [15] which has similar findings and (3) our own proof-of-concept BlockDiploma demonstrating how the different parts can be solved.

Screenshots and code snippets should be sufficient for another researcher to investigate our findings and conclusion. This would be the ultimate test of the validity of our research. If other researchers arrive at the same conclusions this would strengthen our findings.

### 8.4.3 The method used

Computer research with engineering paradigm has been the selected method for our research. The description of the engineering paradigm states that one should keep improving until no further improvements are possible. For the scope of our research we believe that we have done that, but if one is considering making a complete and final alternative to the Diploma registry then there is still more work to do.

The phases have been suitable for our research. During the information phase we gathered and aggregated information about the Diploma registry, how widespread falsification is, what authorities and solution exists. Based upon the information gathered we formulated our hypothesis and research questions. The hypothesis lead to a proposed proof-of-concept solution called BlockDiploma. The concrete proposal was formulated in the form of requirements. BlockDiploma has been used to test and experiment with potential decentralized solutions.

During the analysis phase we investigated challenges with the present solutions and used BlockDiploma to explore decentralized solutions to those challenges. The research effort concluded after the evaluation phase that resulted in this thesis formulating our findings. We believe that this method has been suitable for our research.

## 9 Conclusion

We have presented challenges with the current diploma systems and how decentralized technology could improve it. Pitfalls of the traditional paper versions of diplomas has been explained regarding verification and backup. Norwegian authorities have attempted to improve the security of the diplomas by creating the Diploma registry service. The registry allows students to share their academic results with an employer. Diplomas retrieved from the Diploma registry are signed digitally introducing some trust to the authenticity.

The working hypothesis stated that there were challenges with the present solutions for diplomas and that these could be resolved by decentralizing the diploma registry using blockchain and peer-to-peer technology. We believe that both hypotheses have been strengthened through the analysis of the paper versions and of the Diploma registry and through the development of BlockDiploma.

### *Can the proposed solution do the same as the Diploma registry?*

The short answer is partially. Using the Diploma registry, the student can see all his results on a website and select what parts to share, the data is fetched directly from the higher education providers own databases. BlockDiploma can only handle a complete diploma and students cannot select what parts to share. The API functionality of the Diploma registry is not part of the BlockDiploma solution at the time. BlockDiploma would need to implement support for APIs and allow students to select courses to be written into their diploma to solve all the same tasks as the Diploma registry. BlockDiploma can in other words not do everything the Diploma registry does, but it can do parts of the same tasks such as proving that authenticity of diplomas.

### *Can the proposed solution match or supersede the Diploma registry in terms of availability?*

The best option is to host the service on centralized servers for performance, but as a failsafe the service should be hosted on decentralized storage to ensure that the service is always available. If both the centralized hosted service and the decentralized one is down, then a pure blockchain verification can be performed manually. Manual verification requires that the user hashes the diploma and sends a call to the diploma registry smart contract.

### *Can the proposed solution be a better way of verifying academic diplomas?*

Short answer, yes. The solution allows anyone to verify diplomas. Using a cryptographic hash function which is collision resistant any change to the diploma file will yield a large change of output thus exposing tampering. Checksums like this is more secure because there is no digital signature inside a PDF-file that can be falsified using tools like Photoshop. Browsers not allowing digital signatures in PDF-file to be verified is no longer a problem, every browser capable of running the BlockDiploma DApp can verify the authenticity. If there are any issues with using the BlockDiploma DApp a user can simply use plan B, and a manual call against the diploma registry smart contract will return true or false to the validity of the diploma.

Another advantage is that if a diploma is revoked the verification system will respond to this immediately, while the PDF-version of the diploma from the Diploma Registry is digital signed and will still appear as authentic after the issuer revoked the diploma or as long as the digital signature is valid.

***RQ-1: What are the challenges with our present solutions for academic diplomas?***

The findings earlier in this report points out the following challenges with our present solutions:

- Paper-based versions are time-consuming and difficult to verify.
- Verification of diplomas using the Diploma registry requires the recipient to receive an email, and emails can be spoofed. Current solutions do not have a backup verification process or a verification process that discovers revocation of diplomas immediately. Verifying diplomas issued as PDF-files are not easy beyond the digital signature.
- Availability of services. The Diploma registry will not likely go down since it is a government backed solution. The availability of websites and servers general speaking is not as secure as multiple nodes or peers hosting the service together decentralized. The availability of the diplomas and the verification mechanism could be a challenge with present solutions.
- Ownership of diplomas. The student or recipient does not have the data available regardless of the existence of the Diploma registry. The control of the diploma is governed by the rules of the Diploma registry thus the student is not in complete control of his diploma.
- Diploma cross-border usage is not achieved directly, but foreign employers could trust the digital signature enabling more mobility. Verifying diplomas from foreign issuers efficiently and securely is a challenge.

***RQ-2: Can the challenges with our present diploma solutions be resolved by decentralizing the diploma registry using blockchain and peer-to-peer technology?***

Through exploration using BlockDiploma, our proof-of-concept solution for a decentralized diploma system, we have shown that it is possible to resolve some of the challenges by decentralizing the Diploma registry. Challenges regarding verification can be resolved using blockchain technology. The other challenges with availability, ownership and control of the data can be resolved, but presents new challenges regarding authentication, handling of encryption keys and building an automated system. Because of the new challenges that arises with our decentralized approach we cannot conclude that the challenges can be resolved in a satisfactory way using our approach. We believe that solving all the challenges in a satisfactory way could be achieved by integrating our blockchain-based solution with the present Diploma registry.

## 10 Further Work

BlockDiploma is at a proof-of-concept stage, and for this solution to be used in the Diploma Registry or as a supplement then further development is necessary. The security assessment is not sufficient, and the smart contracts should be applied more rigorous engineering methodologies to, and automated tests should be implemented. Improvements to the user interface and experience is also necessary prior to real world use.

How can individual courses and transcripts be handled using decentralized technology? The Diploma registry allows students to selected what parts of their results to share. Our proof-of-concept cannot perform this task. Building a complete and better diploma system should handle all aspects of handling academic results. Figuring out how to issue single courses and combing them in an elegant and verifiable way is the next step for BlockDiploma.

During the process of developing BlockDiploma we used Ethereum and smart contracts to develop our “backend” and we discovered that there might be a need for upgrading the smart contracts in the future. This is a difficult endeavor that could merit a research effort into how one best could upgrade smart contracts. Smart contracts are quite new, and the solidity language is still new. At the time of writing this thesis solidity is at version 0.6.6 meaning it has not yet reached a stable version and is currently considered under development. Can we trust smart contracts written in Solidity? Unknown unknowns regarding the security risk of smart contracts and Solidity could potentially be a research effort of its own.

The world of decentralized storage is interesting and could and should be furthered researched. Learning more about decentralized storage could contribute in creating better and more secure decentralized storage and diploma solutions.

Finalizing BlockDiploma could be another research effort or project. That research should involve requirements analysis and prioritizing with the stakeholders. Review of this research combined with user testing to find out how the final decentralized solution should be. One key requirement for a complete solution is to solve functional requirement number two regarding issuing diplomas using automated systems. If the blockchain solution could be integrated in existing administration systems used by higher education providers this could enable wide adoption of the solution. Integrating blockchain and decentralized solution into existing systems to build more automated system would be a interesting research effort.



## References

- [1] Andreas M. Antonopoulos and Dr. Gavin Wood, *Mastering Ethereum*, Sebastopol, CA: O'Reilly Media, 1nd edition, 2019.
- [2] M. T. Halvorsen, A. Hoemsnes, "Brekke bekrefter feilaktig cv", *Dagens Næringsliv*, November 19, 2015. [Online], Available: <https://www.dn.no/telenor/sigve-brekke/brekke-bekrefter-feilaktig-cv/1-1-5511143> [Accessed Dec. 05, 2019]
- [3] "Zholia Alemi: Foreign doctor checks after fake psychiatrist case", *BBC*, November 19, 2018. [Online], Available: <https://www.bbc.com/news/health-46258687> [Accessed Oct. 22, 2019]
- [4] E. Tønnessen, "221 saker i registeret for utestengte studenter", *Khrono*, April 23, 2020. [Online], Available: <https://khrono.no/221-saker-i-registeret-for-utestengte-studenter/481506> [Accessed Apr. 23, 2020]
- [5] Risk Advisory Group, "CV Lies 2017", April 23, 2020. [Online], Available: <https://www.riskadvisory.com/campaigns/cv-lies-2017/> [Accessed Dec. 05, 2019]
- [6] Police Security Service, The National Security Authority, The Police and The Business and Industry Security Council, "Personnel security – before, during and after termination of employment", 2018. [Online], Available: [https://www.nsr-org.no/getfile.php/1310408-1521207709/Dokumenter/NSR%20publikasjoner/Veiledninger%20og%20orienteringer/sikkerhet\\_ved\\_ansettelsesforhold\\_2018\\_EN.pdf](https://www.nsr-org.no/getfile.php/1310408-1521207709/Dokumenter/NSR%20publikasjoner/Veiledninger%20og%20orienteringer/sikkerhet_ved_ansettelsesforhold_2018_EN.pdf) [Accessed Oct. 22, 2019]
- [7] Vitnemålsportalen, "The Diploma registry is a secure solution", Mar 6, 2017. [Online], Available: <https://www.vitnemalsportalen.no/english/security/> [Accessed Aug. 6, 2019]
- [8] EQAR, "Diploma and accreditation mills". [Online], Available: <https://www.eqar.eu/kb/accreditation-mills/> [Accessed Oct. 22, 2019]
- [9] EQAR, "NOKUT - Norwegian Agency for Quality Assurance in Education". [Online], Available: <https://www.eqar.eu/register/agencies/agency/?id=36> [Accessed May. 05, 2019]
- [10] University of Nicosia, "Continues a four-year track record of "firsts" in academia and blockchain/cryptocurrency", November 2, 2017. [Online], Available: <https://www.unic.ac.cy/university-of-nicosia-is-the->

- first-university-in-the-world-to-publish-diplomas-of-all-graduating-students-on-the-blockchain/ [Accessed Dec. 05, 2019]
- [11] I. Martin, "Malta becomes first country to explore blockchain education certificates", *Times of Malta*, September 22, 2017. [Online], Available: <https://timesofmalta.com/articles/view/malta-becomes-first-country-to-explore-blockchain-education.658599> [Accessed Dec. 05, 2019]
- [12] CEF Digital, "Introducing the European Blockchain Services Infrastructure (EBSI)". [Online], Available: <https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/ebsi> [Accessed Dec. 05, 2019]
- [13] S. Haber, W.S. Stornetta, "How to time-stamp a digital document", *In Journal of Cryptology*, vol. 3, no. 2, p.99-111, January 1991. Available: SpringerLink, <https://link.springer.com/article/10.1007/BF00196791> [Accessed Aug. 30, 2019]
- [14] R. L. Glass, "A structure-based critique of contemporary computing research", *Journal of Systems and Software*, vol. 28, no. 1, p.3-7 January 1995. Available: ScienceDirect, <https://www.sciencedirect.com/galanga/hvl.no/science/article/pii/016412129400077Z>. [Accessed Mar. 29, 2020]
- [15] MIT Media Lab, "What we learned from designing an academic certificates system on the blockchain", June 2, 2016. [Online], Available: <https://medium.com/mit-media-lab/what-we-learned-from-designing-an-academic-certificates-system-on-the-blockchain-34ba5874f196> [Accessed Feb. 15, 2019]
- [16] A. M. Antonopoulos, *Mastering Bitcoin*, Sebastopol, CA: O'Reilly Media, 2nd edition, 2017.
- [17] L. Lamport, R. Shostak, M. Pease, "The Byzantine Generals Problem", *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, July 1982.
- [18] C. Dwork, M. Naor, "Pricing via Processing or Combatting Junk Mail", 1993. [Online]. Available: [https://link.springer.com/chapter/10.1007/3-540-48071-4\\_10](https://link.springer.com/chapter/10.1007/3-540-48071-4_10). [Accessed Oct. 28, 2019]
- [19] A. Back, "Hashcash- A Denial of Service Counter-Measure", August 1, 2002. [Online]. Available: <http://www.hashcash.org/papers/hashcash.pdf>. [Accessed Oct. 28, 2019]

- [20] Satoshi Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System", 2008. [Online]. Available: <https://bitcoin.org/bitcoin.pdf>. [Accessed Sep. 17, 2019]
- [21] Merriam-Webster, "blockchain". [Online]. Available: <https://www.merriam-webster.com/dictionary/blockchain>. [Accessed May. 20, 2020]
- [22] Dr. G. Hileman, M. Rauchs, "Do you need a Blockchain?", *Crypto Valley Conference on Blockchain Technology (CVCBT)* 2018.
- [23] Dr. G. Wood, "Ethereum: A secure decentralized generalised transaction ledger", October 20, 2019. [Online]. Available: <https://ethereum.github.io/yellowpaper/paper.pdf>. [Accessed Oct. 21, 2019]
- [24] G. Bertoni, J. Daemen, M. Peeters, G. V. Assche, "The Keccak SHA-3 submission", January 14, 2011.[Online]. Available: <https://keccak.team/files/Keccak-submission-3.pdf>. [Accessed: Oct. 31, 2019]
- [25] James Ray, "Patricia Tree", Mar 4, 2019. [Online]. Available: <https://github.com/ethereum/wiki/wiki/Patricia-Tree#merkle-patricia-tree-specification>. [Accessed Apr. 30, 2020]
- [26] Univeristy of Cambridge, "Cambridge Bitcoin Electricity Consumption Index". [Online]. Available: <https://www.cbeci.org/cbeci/comparisons>. [Accessed May. 18, 2020]
- [27] Vitalik Buterin, "Problems", August 22, 2018. [Online]. Available: <https://github.com/ethereum/wiki/wiki/Problems#8-proof-of-stake>. [Accessed Dec. 06, 2019]
- [28] Nick Szabo, "Formalizing and Securing Relationships on Public Networks", *First Monday* 1997. [Online]. Available: <https://nakamotoinstitute.org/formalizing-securing-relationships/>. [Accessed May. 25, 2020]
- [29] Ethereum Community", "Ethereum Improvement Proposals". [Online]. Available: <https://eips.ethereum.org/>. [Accessed Dec. 13, 2019]
- [30] Metamask, "Metamask". [Online]. Available: <https://metamask.io/>. [Accessed: May. 05, 2020]
- [31] M.Gupta, "Blockchain For Dummies", *John Wiley & Sons, Inc.* [Online]. Available: <https://www.ibm.com/downloads/cas/OK5M0E49>. [Accessed Dec. 10, 2019]

- [32] Mark Walport, "Distributed ledger technology: Beyond block chain", *UK: Government Office for Science*, December 2019. [Online]. Available: [https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment\\_data/file/492972/gs-16-1-distributed-ledger-technology.pdf](https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/492972/gs-16-1-distributed-ledger-technology.pdf). [Accessed Dec. 10, 2019]
- [33] Riksrevisjonen, "Etterlevelsesrapport for 2014 om bruk og utstedelse av vitnemål", May 12, 2015. [Online]. Available: [https://khrono.no/files/2017/11/15/bruk\\_og\\_utstedelse\\_av\\_vitnemal\\_i\\_kunnskapssektoren.pdf](https://khrono.no/files/2017/11/15/bruk_og_utstedelse_av_vitnemal_i_kunnskapssektoren.pdf). [Accessed May. 08, 2020]
- [34] UNIT, "02 Resultateiers kontroll over delte resultater". [Online]. Available: <https://www.fellesstudentsystem.no/dokumentasjon/brukerdok/vitnemalsportalen/resultateier/resultateiers-kontroll-delte-resultater.html>. [Accessed May. 21, 2020]
- [35] Blockcerts, "Introduction". [Online]. Available: <https://www.blockcerts.org/guide/>. [Accessed Feb. 19, 2019]
- [36] Vottun, "LIFELONG LEARNING: BLOCKCHAIN FOR EDUCATION". [Online]. Available: <https://blockeducate.com/services/blockchain-for-education/>. [Accessed Sep. 17, 2019]
- [37] BCD, "BCDiploma", April 20, 2018. [Online]. Available: [https://www.bcdiploma.com/ico/img/BCD-WhitePaper\\_last.pdf](https://www.bcdiploma.com/ico/img/BCD-WhitePaper_last.pdf). [Accessed Jan. 29, 2020]
- [38] BCD, "They certify with BCDiploma". [Online]. Available: <https://www.bcdiploma.com/issuers-list.html>. [Accessed Sep. 17, 2019]
- [39] European Blockchain Partnership, "Progress report EBSI - Diploma User Group", *EU* July 7, 2019. [Accessed Dec. 10, 2019]
- [40] diploma.report, "About Diploma". [Online]. Available: <https://diploma.report/>. [Accessed Jan. 29, 2020]
- [41] diploma.report, "About Diploma.report". [Online]. Available: <https://diploma.report/pages/about>. [Accessed Jan. 29, 2020]
- [42] CheckDiploma, "Non-technical guide". [Online]. Available: <https://checkdiploma.org/works#non-technical>. [Accessed Jan. 29, 2020]
- [43] BlockNotary, "BlockNotary". [Online]. Available: <https://www.blocknotary.com/>. [Accessed Jan. 29, 2020].

- [44] DeepVault, "Safeguard my files". [Online]. Available: <https://deepvaultonline.com/>. [Accessed Jan. 29, 2020].
- [45] Saint Bitts LLC, "Bitcoin.com Notary". [Online]. Available: <https://notary.bitcoin.com/>. [Accessed Jan. 29, 2020].
- [46] Certification, "Certification". [Online]. Available: <https://certifaction.io/>. [Accessed Apr. 30, 2020].
- [47] Open Source University, "We are what's next for education & professional development". [Online]. Available: <https://os.university/platform/>. [Accessed Apr. 30, 2020].
- [48] Smart Certificate, "Smart Certificate". [Online]. Available: <https://www.cvtrust.com/>. [Accessed May. 1, 2020].
- [49] ODEM, "It's your future own your education with blockchain technology". [Online]. Available: <https://odem.io/SAIT/>. [Accessed Apr. 30, 2020].
- [50] K. Duffy, J. Nazaré, Y. Ribbens, A. Ronning and L. Parker, "cert-issuer". [Online]. Available: <https://github.com/blockchain-certificates/cert-issuer>. [Accessed Oct. 27, 2019].
- [51] block.co, "How it Works," [Online]. Available: <https://block.co/our-product/#how-it-works-nav>. [Accessed Jan. 31, 2020].
- [52] "Issue PDF certificates", 2019. [Online]. Available: [https://github.com/UniversityOfNicosia/blockchain-certificates/blob/master/docs/issue\\_certificates.md](https://github.com/UniversityOfNicosia/blockchain-certificates/blob/master/docs/issue_certificates.md). [Accessed Oct. 24, 2019].
- [53] K. Karasavvas, "Revoking Records in an Immutable Ledger". [Online]. Available: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8525400>. [Accessed Oct. 22, 2019].
- [54] K. Duffy, "V2: Revocation improvements", February 9, 2017 [Online]. Available: <https://github.com/IMSGlobal/cert-schema/issues/24>. [Accessed Oct, 22 2019].
- [55] Coinmarketcap, "Top 100 Cryptocurrencies by Market Capitalization". [Online]. Available: <https://coinmarketcap.com/>. [Accessed Oct. 23, 2019].
- [56] G. Ateniese, B. Magri, D. Venturi and E. R. Andrade, "Redactable Blockchain—or—Rewriting History in Bitcoin and Friends", 2017. [Online]. Available: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7961975>. [Accessed Oct. 23, 2019].

- [57] Bitcoin Wiki Community, "OP\_RETURN". [Online]. Available: [https://en.bitcoin.it/wiki/OP\\_RETURN](https://en.bitcoin.it/wiki/OP_RETURN). [Accessed Mar. 27, 2020].
- [58] Concourse Open Community, "ETH Gas Station". [Online]. Available: <https://www.ethgasstation.info/>. [Accessed Mar. 27, 2020].
- [59] Kunnskapsdepartementet, "Forskrift om Nasjonal vitnemåls- og karakterportal", December 29, 2016. [Online]. Available: <https://lovdata.no/dokument/SF/forskrift/2016-12-17-1819>. [Accessed April. 27, 2020].
- [60] Tutorialspoint, "Software Requirements". [Online]. Available: [https://www.tutorialspoint.com/software\\_engineering/software\\_requirements.html](https://www.tutorialspoint.com/software_engineering/software_requirements.html). [Accessed April. 27, 2020].
- [61] European Commission, "European Credit Transfer and Accumulation System (ECTS)". [Online]. Available: [https://ec.europa.eu/education/resources-and-tools/european-credit-transfer-and-accumulation-system-ects\\_en](https://ec.europa.eu/education/resources-and-tools/european-credit-transfer-and-accumulation-system-ects_en). [Accessed Jan. 25, 2020].
- [62] UNIT, "02 Hvilke data hentes fra institusjonenes databaser?". [Online]. Available: <https://www.fellesstudentsystem.no/dokumentasjon/brukerdok/vitnemalsportalen/data-i-vitnemalsportalen/data-fra-studiestedene.html>. [Accessed May. 21, 2020].
- [63] European Commission, "Diploma Supplement". [Online]. Available: [https://ec.europa.eu/education/diploma-supplement\\_en](https://ec.europa.eu/education/diploma-supplement_en). [Accessed Apr. 24, 2020].
- [64] W3C, "Introduction to Web Accessibility". [Online]. Available: <https://www.w3.org/WAI/fundamentals/accessibility-intro/>. [Accessed Apr. 24, 2020].
- [65] Kommunal- og moderniseringsdepartementet, "Forskrift om universell utforming av informasjons- og kommunikasjonsteknologiske (IKT)-løsninger". June 25, 2013. [Online]. Available: <https://lovdata.no/dokument/SF/forskrift/2013-06-21-732>. [Accessed Apr. 24, 2020].
- [66] Svein Ølnes, "Beyond Bitcoin: Public Sector Innovation Using the Bitcoin Blockchain Technology". *International Conference on Electronic Government and the Information Systems Perspective*, Springer, 2015. [Accessed Dec. 5, 2019].
- [67] Truffle Suite Community, "Truffle Overview". [Online]. Available: <https://www.trufflesuite.com/docs/truffle/overview>. [Accessed Dec. 11, 2019].

- [68] Truffle Suite Community, "Drizzle Overview". [Online]. Available: <https://www.trufflesuite.com/drizzle>. [Accessed Dec. 11, 2019].
- [69] Ethereum Community, "web3.js - Ethereum JavaScript API". [Online]. Available: <https://web3js.readthedocs.io/en/v1.2.4/index.html>. [Accessed Dec. 12, 2019].
- [70] Truffle Suite Community, "Drizzle". [Online]. Available: <https://www.trufflesuite.com/boxes/drizzle>. [Accessed Dec. 11, 2019].
- [71] Truffle Suite Community, "Ganache Overview". [Online]. Available: <https://www.trufflesuite.com/docs/ganache/overview>. [Accessed Dec. 11, 2019].
- [72] Infura, "Infura". [Online]. Available: <https://infura.io/>. [Accessed May. 6, 2020].
- [73] NCC Group, "Decentralized Application Security Project (or DASP) Top 10 of 2018". 2018. [Online]. Available: <https://dasp.co/index.html>. [Accessed Mar. 28, 2020].
- [74] The OWASP Foundation, "OWASP Top 10", 2017. [Online]. Available: <https://owasp.org/www-project-top-ten/>. [Accessed Mar. 28, 2020].
- [75] ReactJS, "JSX Prevents Injection Attacks", [Online]. Available: <https://reactjs.org/docs/introducing-jsx.html#jsx-prevents-injection-attacks>. [Accessed Mar. 28, 2020].

# A Appendix A: Smart Contract Implementation

Appendix A shows the implementation of all smart contracts used in Block-Diploma.

## A.1 The University Registry Contract

```
1 pragma solidity 0.5.16;
2
3 import "./ERC173.sol";
4
5 /// @author Thomas Reite
6 /// @title Contract containing trusted universities.
7 /// @notice This contract is the authority determining if the
8     university is real or a diploma mill
9 contract UniversityRegistry is ERC173 {
10
11     /// @dev structure used to represent a university
12     struct University{
13         bool valid;
14         string name;
15         string website;
16         address contract_address;
17     }
18
19     /// @notice mapping holding the universities
20     mapping (address => University) universities;
21
22     /// @notice owner or authority is the address of the
23     approving entity.
24     address private authority;
25
26     /// @dev This emits when a university is added to the
27     trusted list.
28     event UniversityAccepted(address university_address,
29         string university_name, string university_website);
30
31     /// @dev This emits when a university loses its status as
32     trusted.
33     event UniversityStatusRevoked(address university_address)
34         ;
35
36     /// @notice the contract needs to know the address of the
37     approving entity.
38     constructor() public{
```



```

32     authority = msg.sender;
33 }
34
35 /// @notice getter for fetching the address of current
    owner/authority.
36 function owner() view external returns (address){
37     return authority;
38 }
39
40 /// @notice this is used when ownership needs to be
    transferred.
41 function transferOwnership(address _newOwner) external
    onlyOwner{
42     emit OwnershipTransferred(authority, _newOwner);
43     authority = _newOwner;
44 }
45
46 /// @notice adds a university to the mapping.
47 function addUniversity(address university_address, string
    memory university_name, string memory
    university_website) public onlyOwner{
48     universities[university_address] = University({valid:
        true, name: university_name, website:
        university_website, contract_address:
        university_address});
49     emit UniversityAccepted(university_address,
        university_name, university_website);
50 }
51
52 /// @notice removes a university from the mapping.
53 function removeUniversity(address university_address)
    public onlyOwner{
54     universities[university_address].valid = false;
55     emit UniversityStatusRevoked(university_address);
56 }
57
58 /// @notice used for checking if an address (university)
    is in the list.
59 function isUniversityTrusted(address university_address)
    public view returns (bool){
60     return universities[university_address].valid;
61 }
62
63 /// @notice getter for fetching all the data about a
    university.
64 function getUniversity(address university_address) public
    view returns (bool, string memory, string memory,
    address){
65     return (universities[university_address].valid,

```

```

66         universities[university_address].name,
67         universities[university_address].website,
68         universities[university_address].contract_address);
69     }
70
71     /// @notice access modifier ensuring only the authority
72     or owner can use a function.
73     modifier onlyOwner() {
74         require(authority == msg.sender);
75         -;
76     }
77 }

```

Listing 3: UniversityRegistry.sol

## A.2 The Diploma Registry Interface

```
1 pragma solidity 0.5.16;
2
3 /// @author Thomas Reite
4 /// @title Standard used for academic diploma registries.
5 interface DiplomaStandard{
6
7     /// @dev Structure used to represent a diploma.
8     struct Diploma {
9         bool valid;
10        address issuerAddr;
11    }
12
13    /// @dev Structure used to represent a diploma holder or
14    recipient
15    struct Recipient {
16        bytes32 ofDiploma;
17        string ipfs;
18    }
19
20    /// @dev This emits when a diploma is issued.
21    event DiplomaIssued(bytes32 diploma, address indexed
22        university);
23
24    /// @dev This emits when a diploma is revoked.
25    event DiplomaRevoked(bytes32 diploma, address university)
26        ;
27
28    /// @dev Should check the msg.sender with the
29    UniversityRegistry before allowing a diploma to enter
30    the verified diplomas list.
31    function issueDiploma(bytes32 diploma, address recipient,
32        string calldata ipfs) external;
33
34    /// @dev Should check the msg.sender with the
35    UniversityRegistry before setting valid to false.
36    function revokeDiploma(bytes32 diploma) external;
37
38    /// @dev gets a diploma used for verifying that the
39    diploma is authentic
40    function verifyDiploma(bytes32 diploma) external returns
41        (bool, address);
42
43    /// @dev gets the recipient data based upon msg.sender
44    function getRecipient() external returns (address,
45        bytes32, string memory);
```

Listing 4: DiplomaStandard.sol

## A.3 The Diploma Registry Contract

```
1 pragma solidity 0.5.16;
2
3 import "./DiplomaStandard.sol";
4 import "./UniversityRegistry.sol";
5
6 /// @author Thomas Reite
7 /// @title DiplomaRegistry a contract that stores valid
8     diplomas.
9 contract DiplomaRegistry is DiplomaStandard {
10
11     /// @notice this mapping contains all the valid diplomas.
12     mapping(bytes32 => Diploma) diplomas;
13
14     /// @notice this mapping contains all the recipients.
15     mapping(address => Recipient) recipients;
16
17     /// @notice Contract that represents the
18     UniversityRegistry.
19     UniversityRegistry private UR;
20
21     /// @notice The Diploma Registry needs to have the
22     address of the UniversityRegistry.
23     constructor(address UniversityRegistryAddr) public{
24         UR = UniversityRegistry(UniversityRegistryAddr);
25     }
26
27     /// @dev Should check the msg.sender with the
28     UniversityRegistry before allowing a diploma to enter
29     the verified diplomas list.
30     function issueDiploma(bytes32 diploma, address recipient,
31         string memory _ipfs) public onlyTrustedUniversity{
32         diplomas[diploma] = Diploma({valid: true, issuerAddr
33             : msg.sender});
34         recipients[recipient] = Recipient({ofDiploma: diploma
35             , ipfs: _ipfs});
36         emit DiplomaIssued(diploma, msg.sender);
37     }
38
39     /// @dev Should check the msg.sender with the
40     UniversityRegistry before setting valid to false.
41     function revokeDiploma(bytes32 diploma) public
42         onlyTrustedUniversity{
43         require(msg.sender == diplomas[diploma].issuerAddr, "
44             DiplomaRegistry: NOT AUTHORIZED");
45         diplomas[diploma].valid = false;
46         emit DiplomaRevoked(diploma, msg.sender);
47     }
48 }
```

```

37
38     /// @dev gets a diploma used for verifying that the
        diploma is authentic
39     function verifyDiploma(bytes32 diploma) public returns (
        bool, address){
40         return (diplomas[diploma].valid, diplomas[diploma].
            issuerAddr);
41     }
42
43     /// @notice getter for recipient
44     function getRecipient() public returns (address, bytes32,
        string memory){
45         return (msg.sender, recipients[msg.sender].ofDiploma,
            recipients[msg.sender].ipfs);
46     }
47
48     /// @notice access modifier ensuring that the msg.sender
        is in the UniveristyRegistry.
49     modifier onlyTrustedUniversity() {
50         require(UR.isUniversityTrusted(msg.sender), "
            DiplomaRegistry: Not Authorized");
51         -;
52     }
53
54 }

```

Listing 5: DiplomaRegistry.sol

## A.4 The University Interface

```
1 pragma solidity 0.5.16;
2
3 /// @author Thomas Reite
4 /// @title interface for university smart contracts
5 /// @notice interface containing all the function a
    university contract needs to comply with in order to issue
    diplomas to the DiplomaRegistry.
6 /// the interface requires that ERC173 is implemented since
    there needs to be an owner or superuser if you like in
    this contract.
7 interface UniversityStandard{
8
9     /// @dev This should be used to represent an employee
    that is allowed to handle diplomas.
10
11    /// You should implement this as a mapping.
12    struct TrustedEmployee{
13        bool trusted;
14        string employee_identifier;
15    }
16
17    /// @dev This emits when permission is given.
18    event PermissionGiven (address indexed account, string
    employee_identifier);
19
20    /// @dev This emits when permission is revoked.
21    event PermissionRemoved (address indexed account, string
    employee_identifier);
22
23    /// @dev This emits when a diploma is issued.
24    event DiplomaIssued(bytes32 diploma, string
    employee_identifier, string studentID);
25
26    /// @dev This emits when a diploma is revoked.
27    event DiplomaRevoked(bytes32 diploma, string
    employee_identifier, string studentID);
28
29    /// @dev Check that the owner/superuser/admin of this
    contract is the msg.sender before adding the employee.
30    function addTrustedEmployee(address addr, string calldata
    identifier) external;
31
32    /// @dev Check that the owner/superuser/admin of this
    contract is the msg.sender before removing the
    employee.
33    function removeTrustedEmployee(address addr) external;
34
```

```

35     /// @dev Should check if the sender is a trusted employee
    , then call the DiplomaRegistry contract with the
    diploma.
36     function issueDiploma(bytes32 diploma, string calldata
    studentID, address recipient, string calldata ipfs)
    external;
37
38     /// @dev Should check if the sender is a trusted employee
    , then call the DiplomaRegistry contract with the
    diploma.
39     function revokeDiploma(bytes32 diploma, string calldata
    studentID) external;
40
41
42 }

```

Listing 6: UniversityStandard.sol



## A.5 The University Contract

```
1 pragma solidity 0.5.16;
2
3 import "./UniversityStandard.sol";
4 import "./ERC173.sol";
5 import "./DiplomaRegistry.sol";
6
7 /// @author Thomas Reite
8 /// @title Contract that represents a university.
9 /// @notice This contract is used to handle which employees
    can interact with a DiplomaRegistry on behalf of the
    university.
10 contract University is UniversityStandard, ERC173 {
11
12     /// @notice the owner of this contract.
13     address private university;
14
15     /// @notice mapping of trusted employes
16     mapping(address => TrustedEmployee) trusted_employees;
17
18     /// @notice the address of the DiplomaRegistry that you
    want to publish to.
19     DiplomaRegistry private diplomaRegistry;
20
21     /// @notice sets the owner address and the address of the
    DiplomaRegistry to be used.
22     constructor(address _diplomaRegistry) public{
23         university = msg.sender;
24         diplomaRegistry = DiplomaRegistry(_diplomaRegistry);
25     }
26
27     /// @notice getter that fetches the address of the
    univeristy (owner account).
28     function owner() view external returns (address){
29         return university;
30     }
31
32     /// @notice function for transferring ownership.
33     function transferOwnership(address _newOwner) public {
34         require(msg.sender == university, "NOT PERMITTED!");
35         emit OwnershipTransferred(university, _newOwner);
36         university = _newOwner;
37     }
38
39     /// @notice function for adding a employee to the trusted
    list allowing that person to act on behalf of the
    university.
40     function addTrustedEmployee(address addr, string memory
```

```

41     identifier) public onlyUniversity{
42         trusted_employees[addr] = TrustedEmployee({trusted:
43             true, employee_identifier: identifier});
44         emit PermissionGiven(addr, identifier);
45     }
46     /// @notice function for removing a employee from the
47     trusted list.
48     function removeTrustedEmployee(address addr) public
49     onlyUniversity {
50         trusted_employees[addr].trusted = false;
51         emit PermissionRemoved(addr, trusted_employees[addr].
52             employee_identifier);
53     }
54     /// @notice function used for issuing a diploma.
55     function issueDiploma(bytes32 diploma, string memory
56     studentID, address recipient, string memory ipfs)
57     public onlyTrustedEmployee{
58         diplomaRegistry.issueDiploma(diploma, recipient, ipfs)
59         ;
60         emit DiplomaIssued(diploma, trusted_employees[msg.
61             sender].employee_identifier, studentID);
62     }
63     /// @notice function used for revoking a diploma.
64     function revokeDiploma(bytes32 diploma, string memory
65     studentID) public onlyTrustedEmployee{
66         diplomaRegistry.revokeDiploma(diploma);
67         emit DiplomaRevoked(diploma, trusted_employees[msg.
68             sender].employee_identifier, studentID);
69     }
70     /// @notice getter for fetching an trusted_employee
71     function getTrustedEmployee(address addr) public view
72     returns (bool, string memory){
73         return (trusted_employees[addr].trusted,
74             trusted_employees[addr].employee_identifier);
75     }
76     /// @notice function used to take down this contract,
77     diplomas issued will then be locked in DiplomaRegistry
78     .
79     function deactivateContract() external onlyUniversity{
80         selfdestruct(msg.sender);
81     }
82     /// @notice access modifier allowing university account (
83     owner) to perform some action.

```

```
74     modifier onlyUniversity() {
75         require(university == msg.sender);
76         -;
77     }
78
79     /// @notice access modifier allowing trusted employees to
80     perform some action.
81     modifier onlyTrustedEmployee() {
82         require(trusted_employees[msg.sender].trusted, "
83             University: Not Authorized");
84     }
85 }
```

Listing 7: University.sol

## A.6 The ERC-173 Contract

```
1 pragma solidity 0.5.16;
2
3 /// @title ERC-173 Contract Ownership Standard
4 /// @dev See https://github.com/ethereum/EIPs/blob/master/
5     EIPS/eip-173.md
6 interface ERC173 {
7     /// @dev This emits when ownership of a contract changes.
8     event OwnershipTransferred(address indexed previousOwner,
9         address indexed newOwner);
10
11     /// @notice Get the address of the owner
12     /// @return owner The address of the owner.
13     function owner() view external returns (address);
14
15     /// @notice Set the address of the new owner of the
16     contract
17     /// @param _newOwner The address of the new owner of the
18     contract
19     function transferOwnership(address _newOwner) external;
20 }
```

Listing 8: ERC173.sol

## B Appendix B: BlockDiploma Screenshots

Appendix B contains all the screens from the implemented BlockDiploma system.

### B.1 Public Website

Screenshots of the public parts of the BlockDiploma DApp.

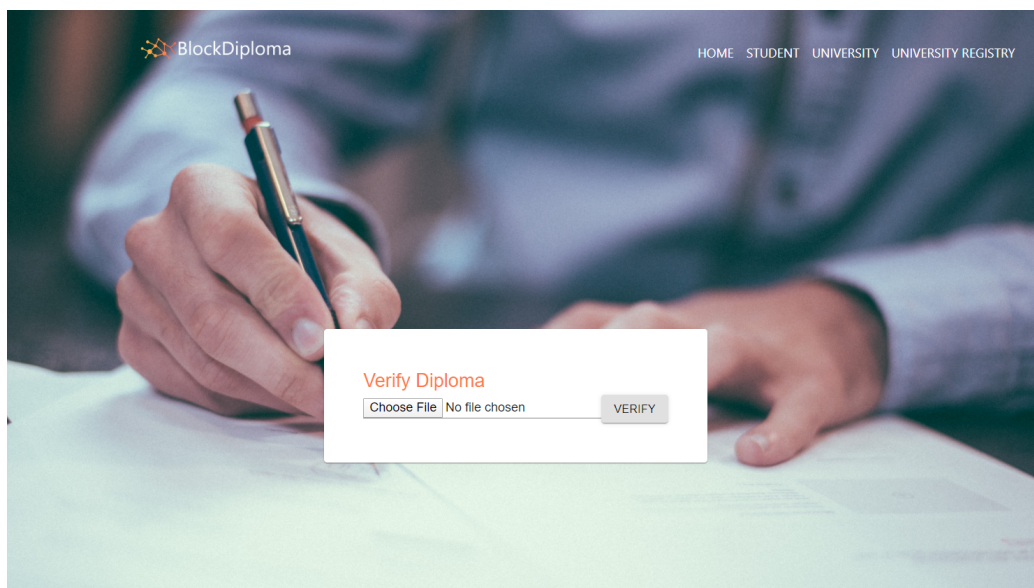


Figure 32: The public website for validating diplomas.

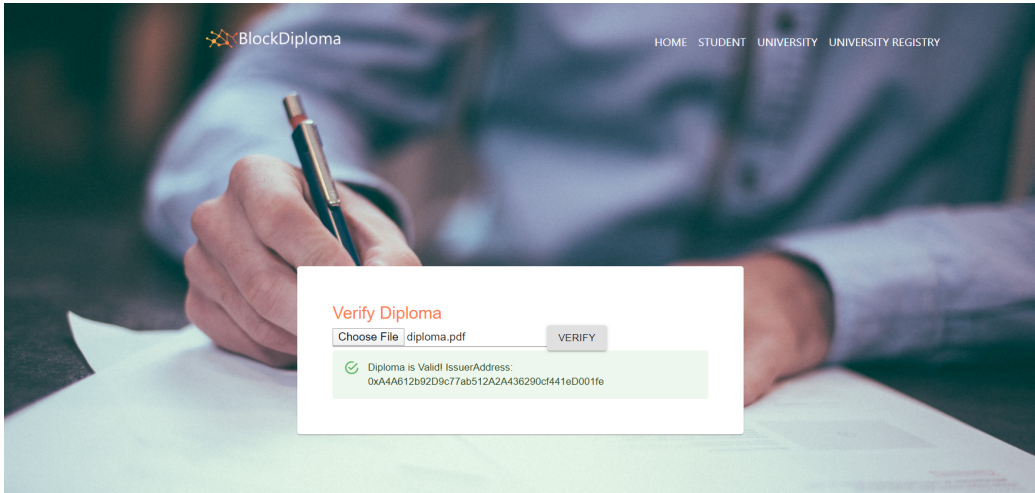


Figure 33: The public website when a diploma successfully validated.

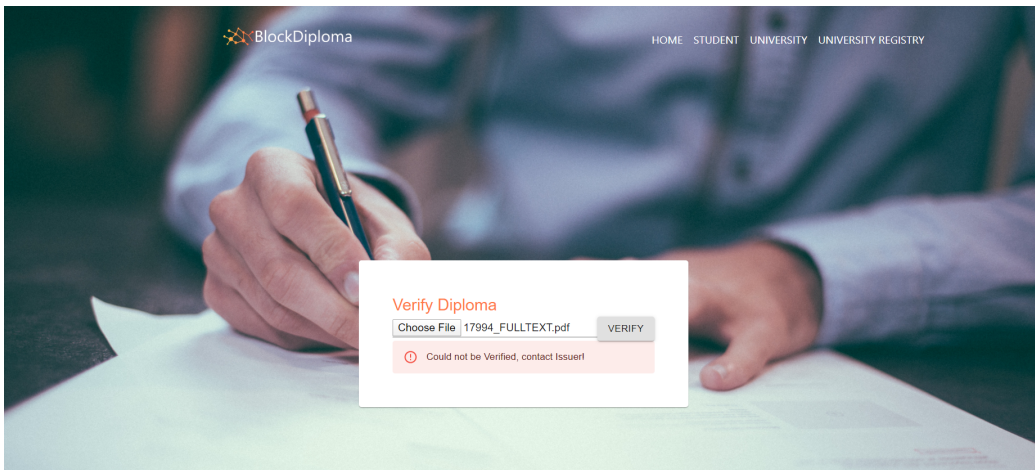


Figure 34: The public website when diploma could not be validated.



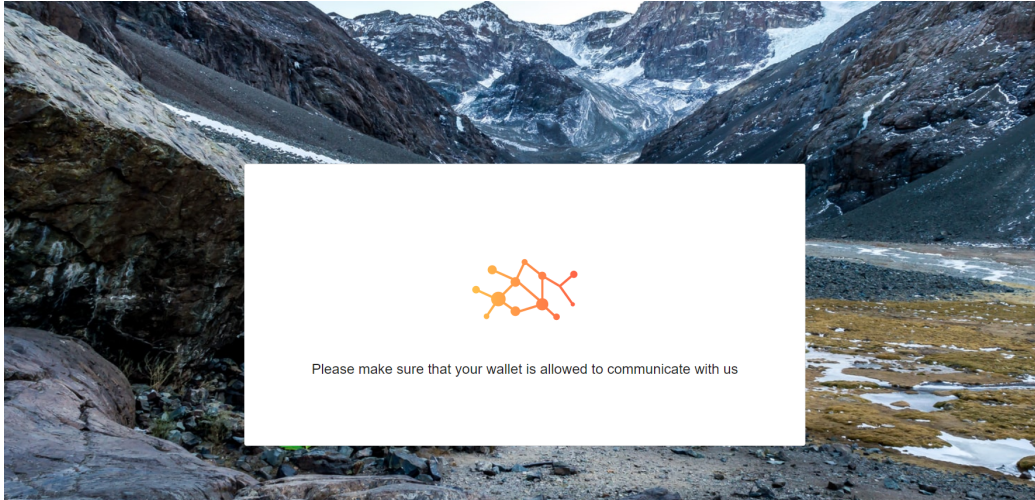



Figure 35: Loading screen displayed when no wallet is connected to the DApp.



  
**Restricted Area**

In order to access this site you must have been granted access. If access has been given to you, do the following:

1. Sign in to your wallet extension in the browser you use.
2. Check that your wallet is allowed to communicate with us.
3. Refresh the page, if you are authorized you should be granted access now.

Copyright © BlockDiploma 2020.

Figure 36: Page displayed when user is not authorized to perform any actions.

## B.2 University Registry

The University Registry pages is related to all actions performed in the University Registry-contract.

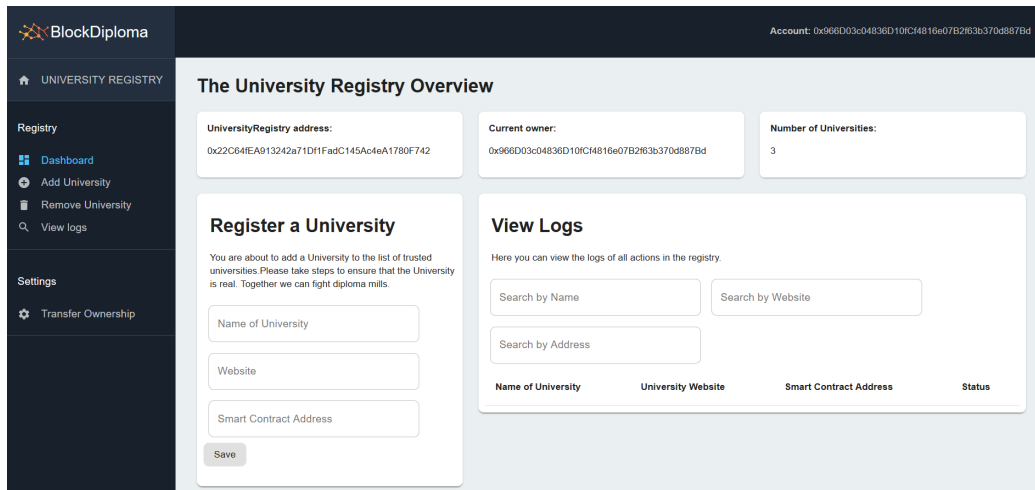


Figure 37: The University Registry Dashboard.

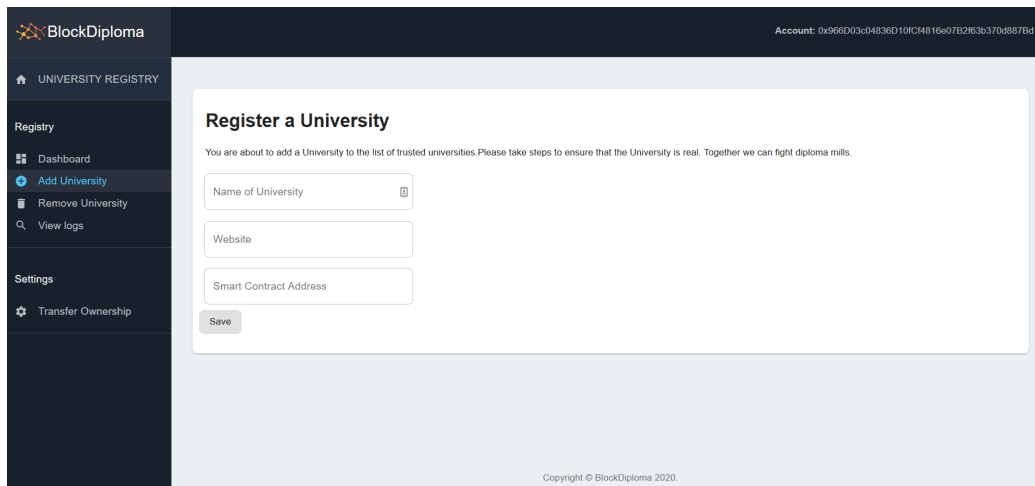


Figure 38: Page for adding a university to the university registry.



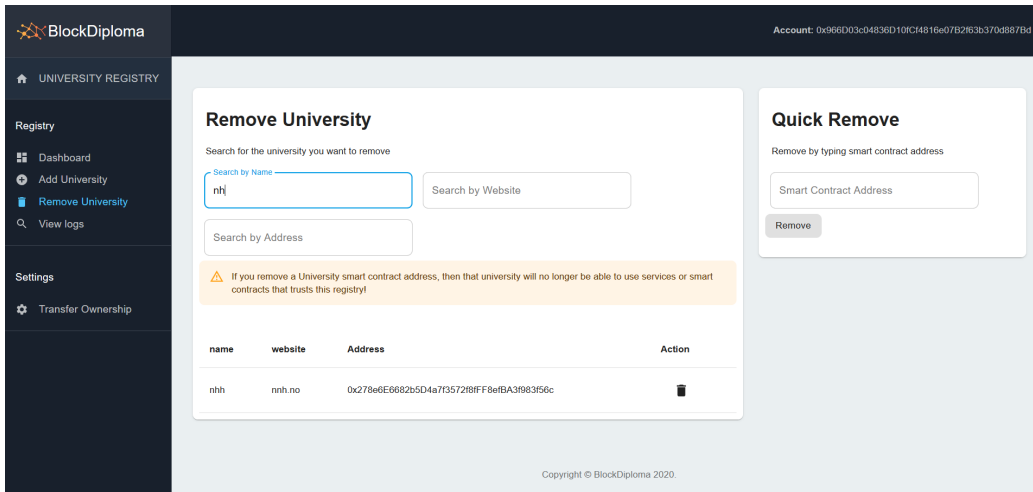


Figure 39: Page for removing a university from the university registry.

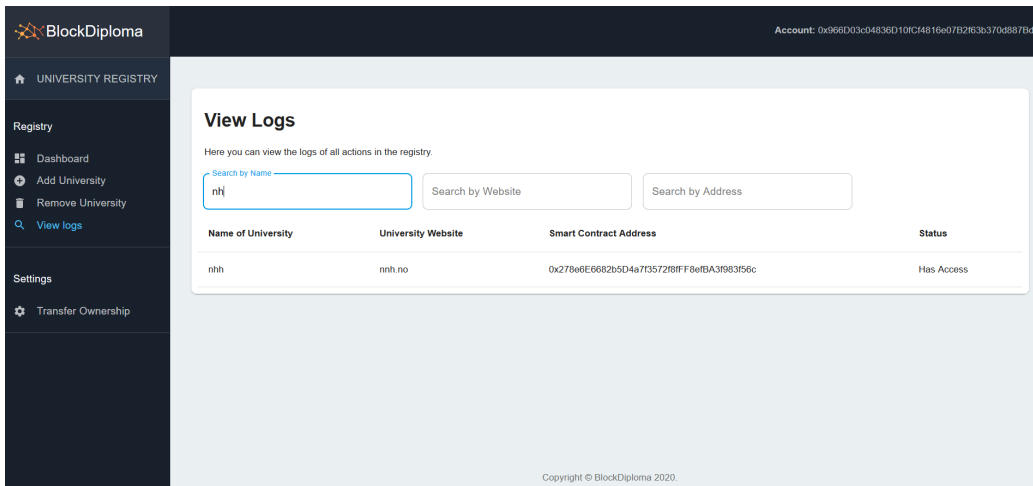


Figure 40: Page for viewing event logs of actions in the University Registry.

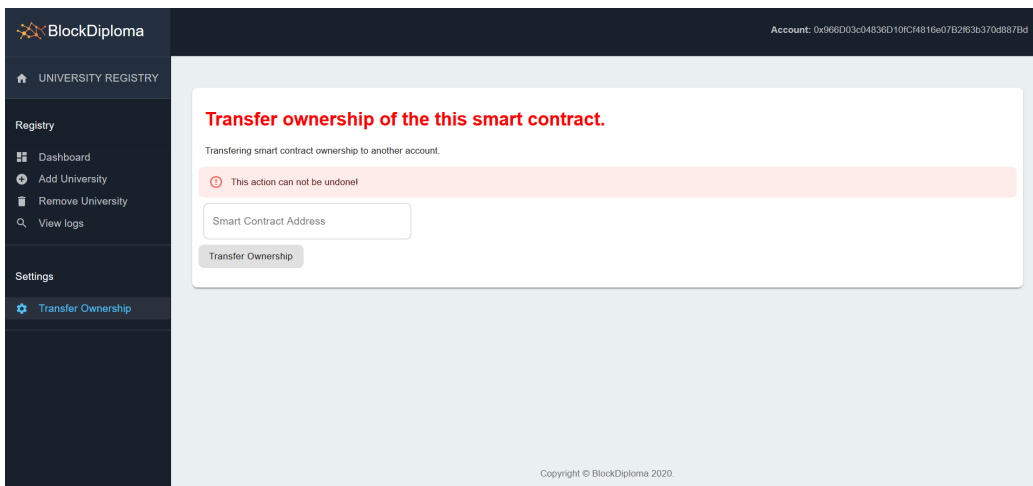


Figure 41: Page for transferring ownership of the University Registry contract.

## B.3 University

The University pages is related to all actions performed in the University contract.

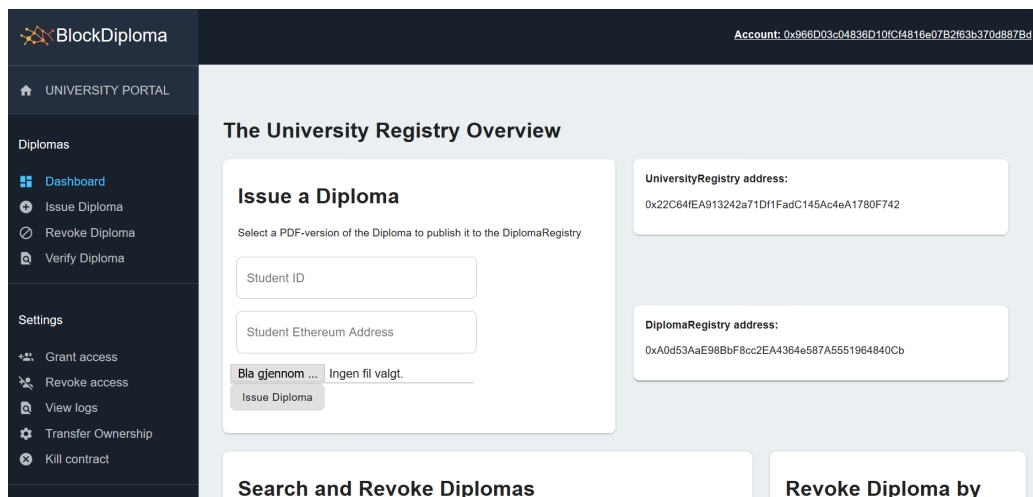


Figure 42: The University Portal Dashboard

### Manual Registration of Diploma

**Student details**

Family Names <input type="text"/>	Given Names <input type="text"/>	Date of Birth <input type="text"/>	Student ID <input type="text"/>	Address <input type="text"/>	<input type="button" value="Lock Student"/>
-----------------------------------	----------------------------------	------------------------------------	---------------------------------	------------------------------	---------------------------------------------

**Add Courses**

<input type="button" value="v"/>	Course code and name <input type="text"/>	Semester <input type="text"/>	Credits <input type="text"/>	Grade <input type="text"/>	Distribution <input type="text"/>	<input type="button" value="Add"/>
----------------------------------	-------------------------------------------	-------------------------------	------------------------------	----------------------------	-----------------------------------	------------------------------------

Current Courses

**Common Courses**

**Programme Courses**

**Specialization Courses**

**Elective Courses**

### Issue a Diploma

Select a PDF-version of the Diploma to publish it to the DiplomaRegistry

Student ID <input type="text"/>	Student Ethereum Address <input type="text"/>	<input type="button" value="Bla gjennom ..."/> Ingen fil valgt.
---------------------------------	-----------------------------------------------	-----------------------------------------------------------------

Figure 43: Page for issuing a Diploma.

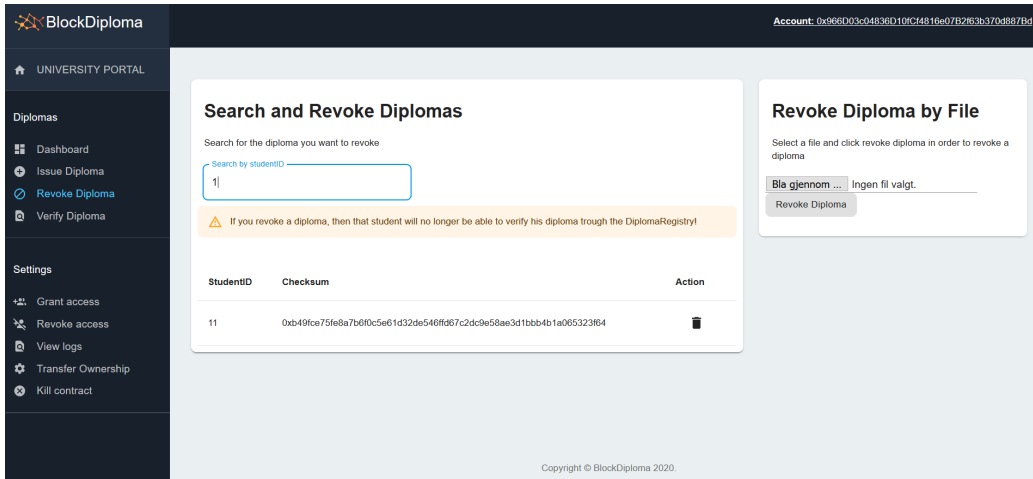


Figure 44: Page for revoking a diploma from the diploma registry.

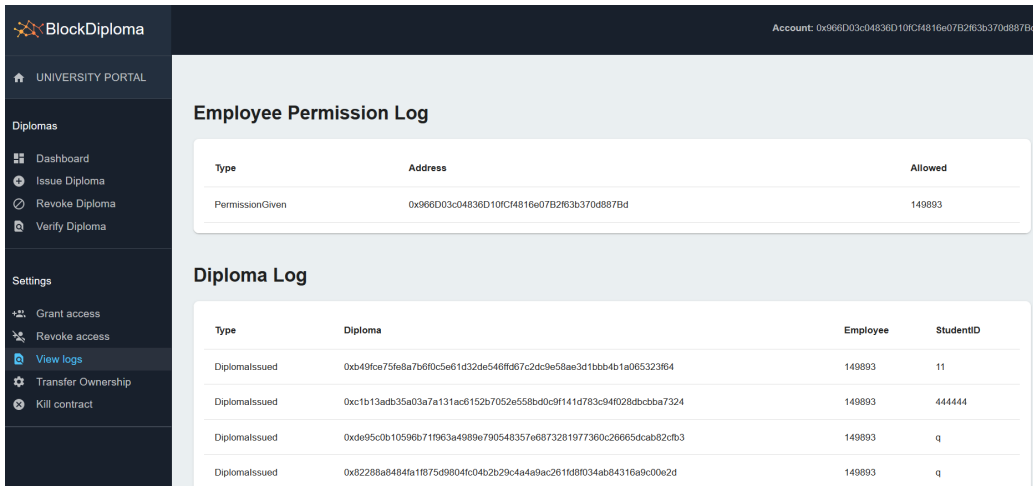


Figure 45: Page for viewing event logs of diploma and employee actions.

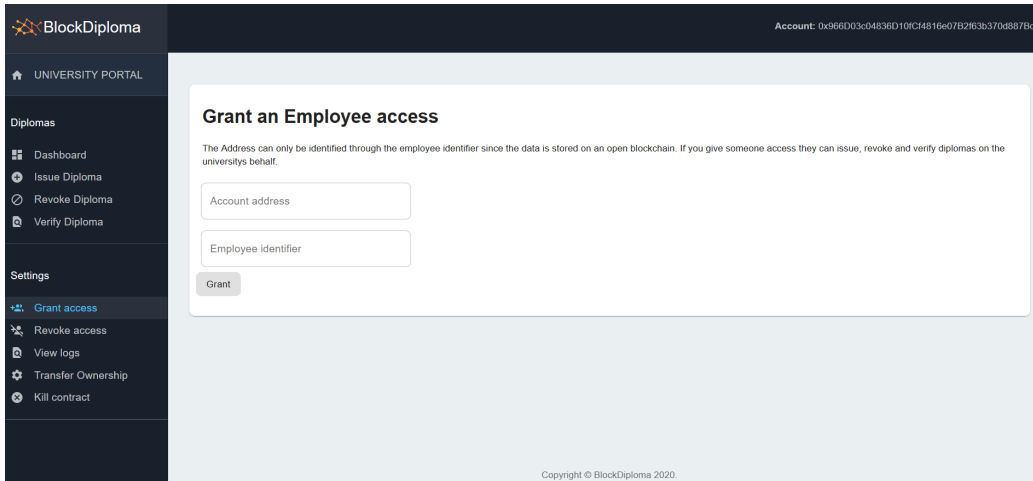


Figure 46: Page for giving an employee access to the university-contract.

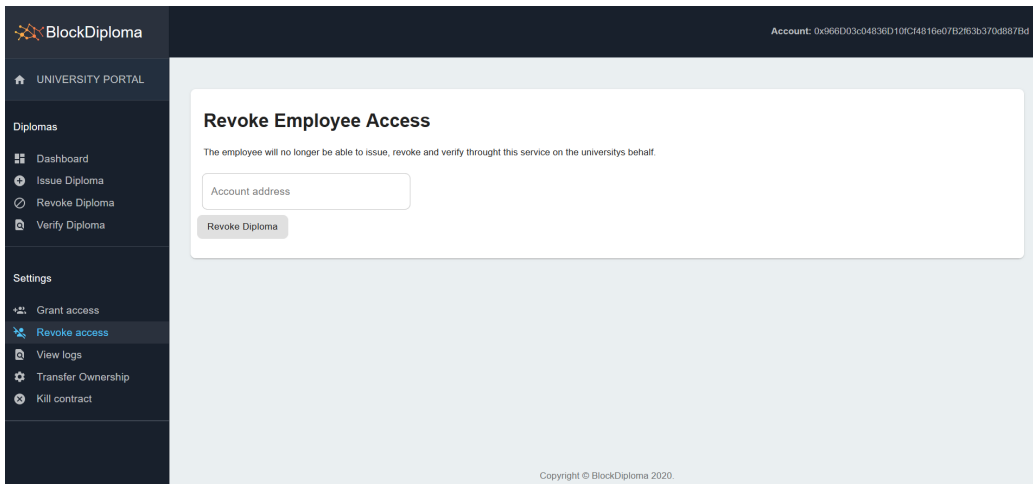


Figure 47: Page for revoking access for an employee from the university contract.

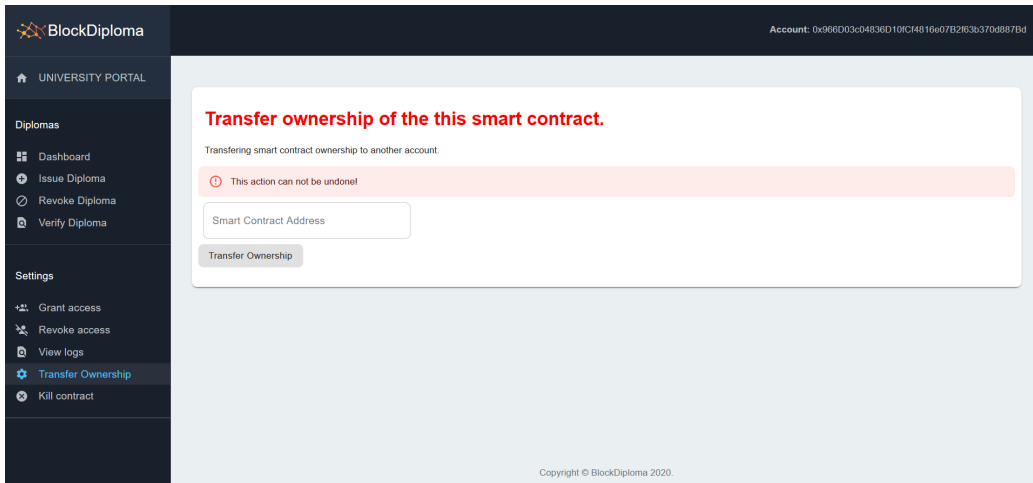


Figure 48: Page for transferring ownership of the university contract.

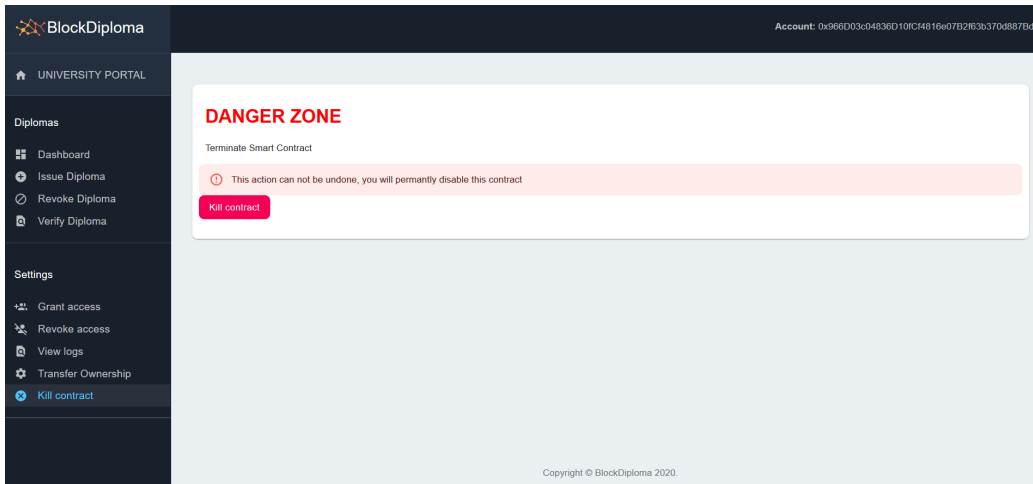


Figure 49: Page for terminating the University contract.

## B.4 Student

The student pages is related to all actions performed in the Student Portal.

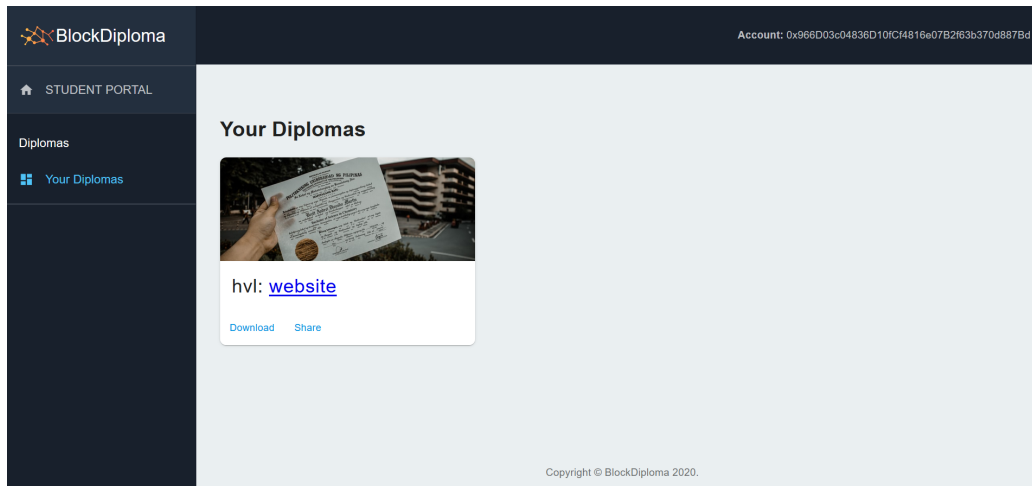


Figure 50: The Student Portal

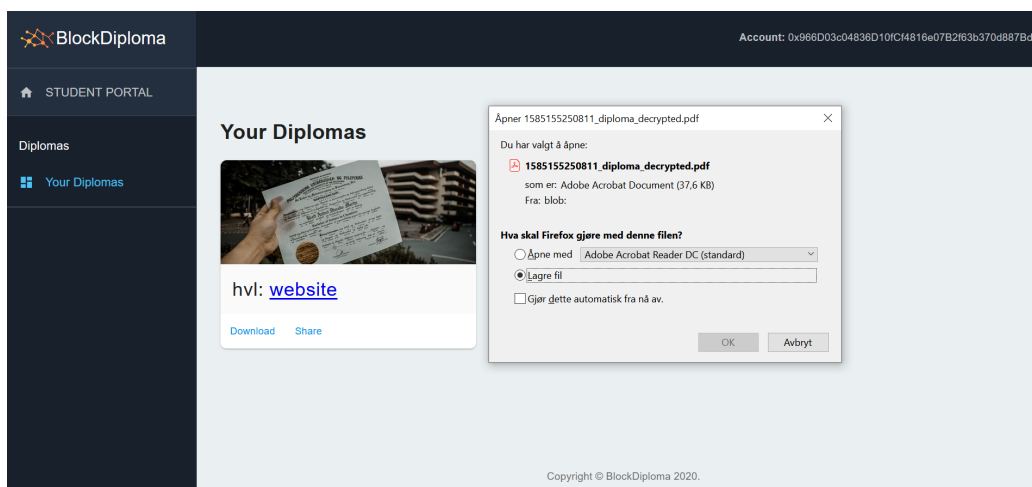


Figure 51: Page showing the download of a diploma from IPFS.





Name: q q  
Date of Birth: q

## Higher Education

### Western Norway University of Applied Sciences

Degree: Bachelor of Engineering (180 ECTS) achieved 25/06/2018

Study programme: Computing

Note: Bachelor i ingeniørfag

Note: Studiet er fullført ved studiested Bergen.

Course	Semester	Credits	Grade	Distribution
<b>Common Courses</b>				
DAT103 Databaser og os	Autumn 2015	10	A	n/a
<b>Programme Courses</b>				
<b>Specialization Courses</b>				
<b>Elective Courses</b>				

The distribution of grades is shown by the percentage for courses using the graded scale A-F. Fail (F) is not included in the distribution. All results from the last five years are included in the calculation. The distribution is also shown for courses that have been active for less than five years. There has to be at least 10 approved results during the period.

DOCUMENT GENERATED: MON, 23 MAR 2020 09:38:57 GMT

The document is electronically produced by BlockDiploma - the Decentralized and trustless diploma portal for educational results and contains results from the issuing institutions administrative system. The document can be verified by using the website of blockdiploma; <https://www.blockdiploma.com>

Figure 52: Example of a BlockDiploma PDF-generated diploma.

# Diploma Supplement

This Diploma Supplement model was developed by the European Commission, Council of Europe and UNESCO/CEPES. The purpose of the supplement is to provide sufficient independent data to improve the international 'transparency' and fair academic and professional recognition of qualification (diplomas, degrees, certificates etc.). It is designed to provide a description of the nature, level, context, content and status of the studies that were pursued and successfully completed by the individual named on the original qualification to which this supplement is appended. It should be free from any value judgements, equivalence statements or suggestions about recognition. Information in all eight sections should be provided. Where information is not provided, an explanation should give the reason why.

## 1 INFORMATION IDENTIFYING THE HOLDER OF THE QUALIFICATION

- 1.1 Family name(s): q
- 1.2 Given name(s): q
- 1.3 Date of birth (day/month/year): q
- 1.4 Student identification number or code: q

## 2 INFORMATION IDENTIFYING THE QUALIFICATION

- 2.1 Name of qualification and (if applicable) title conferred (in original language): Bachelor i ingeniørfag The title bachelor is protected by law in Norway.
- 2.2 Main field(s) of study for the qualification: Natural Sciences (Mathematics and Statistics. Physics, Chemistry and the Environment. Data processing). Social Sciences. Technical Subjects. Optional Subjects. Research Project.
- 2.3 Name and status of awarding institution (in original language): Høgskulen på Vestlandet, a public university college. The quality assurance system was evaluated and approved by the Norwegian Agency for Quality Assurance in Education in 2015.
- 2.4 Name and status of institution administering studies: See section 2.3
- 2.5 Language(s) of instruction/examination: Norwegian

Figure 53: Example of generated diploma supplement

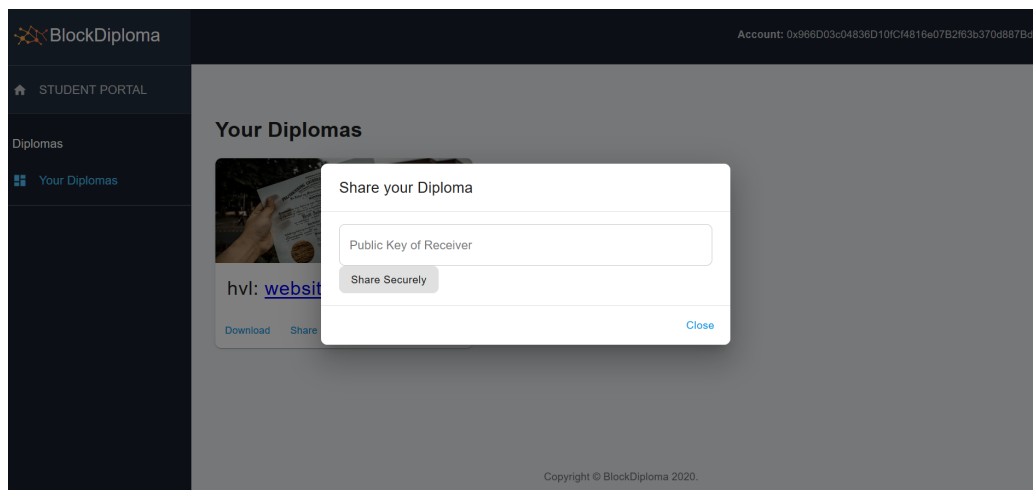


Figure 54: Private sharing using asymmetric encryption.

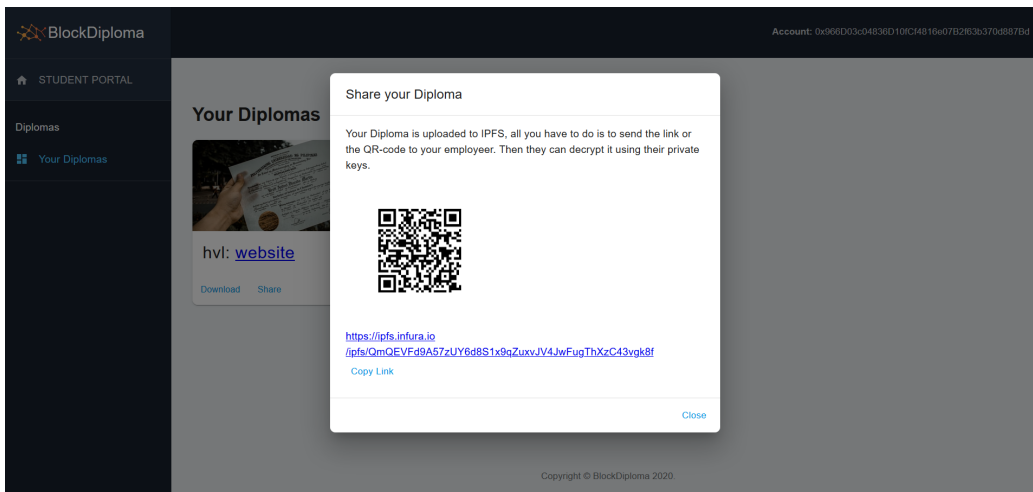


Figure 55: Sharing screen with link and QR-code that can be sent to an employer.