

Sander Osvik Brekke
Ivan Norderhaug
Kristian Røren Svanholm

License Management in Closed Offline Networks Using Modern Cryptographic Solutions

In cooperation with Nevion Europe AS

Bachelor's thesis in Bachelor in Computer Science and Bachelor in
Programming

Supervisor: Steven Yves Le Moan

May 2023

Sander Osvik Brekke
Ivan Norderhaug
Kristian Røren Svanholm

License Management in Closed Offline Networks Using Modern Cryptographic Solutions

In cooperation with Nevion Europe AS

Bachelor's thesis in Bachelor in Computer Science and Bachelor in Programming
Supervisor: Steven Yves Le Moan
May 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science



Abstract

Title:	License Management in Closed Offline Networks Using Modern Cryptographic Solutions
Date:	May 20, 2023
Participants:	Sander Osvik Brekke Ivan Norderhaug Kristian Røren Svanholm
Supervisor:	Steven Yves Le Moan
Employer:	Anders Dale, Nevion Europe AS
Keywords:	Offline Digital Rights Management, Licenses, Chain of Trust, Certificates
Pages:	105 without appendix 222 with appendix
Appendices:	15
Availability:	Open

Nevion wanted a concept solution to a system that incorporates a time-based licensing model, where the customer can generate sublicenses, from a larger top level license, on the behalf of Nevion. This should happen without an internet connection. Through this project, there has been developed a concept to the wanted solution through combining technologies like X.509, AES-256 and PKI. Furthermore, there have been developed and implemented a proof of concept, proving the practicality aspect of the system. Managing licenses and digital rights in off-line environments is a risk regarding security. Because of this, Nevion wanted a security review to shed light on the security aspects of the developed concept. The total solution gives Nevion a base for implementing a similar system and utilizing this report for further research and development.

Sammendrag

Tittel:	License Management in Closed Offline Networks Using Modern Cryptographic Solutions
Dato:	20. mai 2023
Deltakere:	Sander Osvik Brekke Ivan Norderhaug Kristian Røren Svanholm
Veileder:	Steven Yves Le Moan
Oppdragsgiver:	Anders Dale, Nevion Europe AS
Nøkkelord:	Digitale rettigheter i frakoblede miljøer, lisenser, tillitskjede, sertifikater
Antall sider:	105 uten vedlegg 222 med vedlegg
Antall vedlegg:	15
Tilgjengelighet:	Åpen

Nevion ønsket en løsning til et system som benytter seg av en tidsbasert lisensieringsmodell der kunden selv skal kunne generere del-lisenser av en større lisens på Nevions vegne, uten internettilkobling. Gjennom dette prosjektet har det blitt utviklet et konsept til det ønskede systemet ved å benytte seg av ulike teknologier som X.509, AES-256 og PKI. Videre har det også blitt implementert et bevis på at dette konseptet er mulig å gjennomføre i praksis. Håndtering av lisenser og digitale rettigheter i miljøer uten internettilkobling innebærer en stor sikkerhetsrisiko. Derfor ønsket Nevion å vite om et slikt system er sikkert, og etterspurte en sikkerhetsanalyse av det utviklede konseptet. Den totale løsningen gir Nevion et godt utgangspunkt til å implementere et liknende system og benytte prosjektrapporten til videre forskning og utvikling.

Preface

This is a bachelor thesis written at the Norwegian University of Science and Technology in Gjøvik, Faculty of Information Technology and Electrical Engineering, Department of Computer Science during the spring of 2023. The bachelor thesis has been written as a part of the Bachelor's degree program in Computer Science and the Bachelor's degree in Programming, written by Ivan Norderhaug, Sander Osvik Brekke, and Kristian Røren Svanholm.

We would like to extend our sincere appreciation to our supervisor, Steven Yves Le Moan, for his unwavering support, insightful feedback, and constructive discussions. Additionally, we would like to express our gratitude to Anders Dale, our Nevia representative, for his valuable contributions, helpful guidance, and reflective discussions throughout the process.

There are many additional people who deserve a thank you; first and foremost our classmates who have been motivating and guiding us through deep, both relevant and irrelevant, conversations. Countless hours at the reading hall went by more quickly in your presence. Also, it is important not to forget the most valuable help; each other! It has been an interesting, rewarding, and fun process.

Last but not least, we would like to thank our supporters at home: Pernille, Marthe, and Laura.

Contents

Abstract	iii
Sammendrag	v
Preface	vii
Contents	ix
Figures	xiii
Tables	xv
Code Listings	xvii
Acronyms	xix
Glossary	xxi
1 Introduction	1
1.1 Background	1
1.2 Problem Area	2
1.3 Problem Statement	2
1.4 Scope	3
1.5 Assumptions	3
1.6 Goals and Constraints	4
1.6.1 Constraints	4
1.6.2 Result Goals	5
1.6.3 Effect Goals	5
1.7 Target Audience	6
1.8 Group Background	6
1.9 Project Structure	6
1.9.1 Project Roles	6
1.9.2 Technical Areas of Responsibility	7
1.10 Report Structure	8
2 Development Process	11
2.1 Development Model	11
2.2 Meetings	12
2.3 Scrum Sprints Summary	12
3 Concept Review	17
3.1 State of Technology	17
3.2 Theory	19
3.2.1 Cryptography	19

3.2.2	Digital Certificates	22
3.2.3	Summary	23
3.3	Offline License Management (OLM)	23
3.3.1	Certificate Authority	23
3.3.2	Network Management System	24
3.3.3	License File Aggregator	24
3.3.4	Communication	25
3.3.5	How Does it Work?	26
3.4	Security Aspects	27
3.5	Adapted Chain of Trust	28
3.6	Discussion	29
3.6.1	Challenges	30
3.6.2	Alternative Solutions	31
4	Security Review	33
4.1	Introduction	33
4.2	Theory	33
4.3	Methodology	37
4.3.1	Structure	38
4.3.2	Assets Identification	38
4.3.3	Threat Identification	38
4.3.4	Risk Assessment	38
4.3.5	Risk Control	38
4.4	Assets Identification	39
4.4.1	Sensitive Data & Third Party Components	39
4.4.2	Critical Assets	39
4.4.3	CIA Classifying of Critical Assets	40
4.5	Actors & Roles	42
4.5.1	Root License Administrator	42
4.5.2	Client Employee	42
4.5.3	Client Network admin	42
4.5.4	External Actors	42
4.5.5	Malicious Actors	43
4.6	Risk Appetite	43
4.7	Threat Identification: Stride	44
4.7.1	Spoofing	44
4.7.2	Tampering	44
4.7.3	Repudiation	45
4.7.4	Information Disclosure	45
4.7.5	Denial of Service	46
4.7.6	Escalation of Privilege	46
4.8	Risk Assessment: DREAD	46
4.9	Risk assessment: Threat Matrix	51
4.10	Risk Control: Bowtie Modelling	52
4.10.1	Spoofing	52

- 4.10.2 Tampering 53
- 4.10.3 Repudiation 54
- 4.10.4 Information Disclosure 55
- 4.10.5 Denial of Service 56
- 4.10.6 Escalation of Privilege 57
- 4.11 Residual Risk 57
- 4.12 Discussion 58
 - 4.12.1 How Does the Risk Stand? 58
 - 4.12.2 Is it Viable? 58
- 4.13 Conclusion 59
- 5 Proof of Concept 61**
 - 5.1 Introduction 61
 - 5.2 Requirements 62
 - 5.2.1 Functional Requirements 62
 - 5.2.2 Non-functional Requirements 62
 - 5.2.3 Operational Requirements 62
 - 5.3 Technologies 62
 - 5.3.1 OpenSSL 62
 - 5.3.2 Network Management System 63
 - 5.3.3 License File Aggregator 63
 - 5.3.4 Frontend 63
 - 5.4 Design 64
 - 5.4.1 Structure 64
 - 5.4.2 Communication 64
 - 5.5 License Signing 65
 - 5.6 Network Management System 65
 - 5.6.1 API 67
 - 5.6.2 LFA Registry 71
 - 5.6.3 Client 72
 - 5.6.4 License Parsing 73
 - 5.6.5 Security Features 73
 - 5.6.6 Pools 75
 - 5.6.7 Persistence 76
 - 5.6.8 Sub Level License Generation 76
 - 5.6.9 Logging 76
 - 5.6.10 Project Initialization 77
 - 5.7 License File Aggregator 78
 - 5.7.1 Server 78
 - 5.7.2 OpenSSL 80
 - 5.7.3 Client 82
 - 5.8 Frontend 82
 - 5.9 Secrets Handling 84
 - 5.10 Licenses 84
 - 5.11 Hosting and Version Control 86

5.11.1	Distribution	86
5.11.2	Deployment	86
5.11.3	Git	88
5.12	Quality Assurance	88
5.13	Discussion	90
5.13.1	Administrative Decisions	90
5.13.2	Proof of Concept VS. Concept	91
5.13.3	Security	91
5.13.4	Is the Concept Proven?	92
6	Discussion	95
6.1	Project Retrospective	95
6.2	Project Plan	96
6.3	Version Control & QA	97
6.4	SCRUM	99
6.5	Meetings	100
6.6	Thesis	100
6.7	Engineering Values and Perspectives	101
7	Conclusion	103
7.1	Further Work	104
	Bibliography	105
A	Project Agreement	111
B	Task Description	119
C	Project Plan	121
D	Gantt Diagram	139
E	Group Rules	141
F	Status Report	143
G	Meeting Minutes	149
H	Time Sheets	193
I	Frontend Graphical User Interface	197
J	Proof of Concept Initial Setup Guide	199
K	Network Management System ReadMe	203
L	License File Aggregator ReadMe	205
M	Frontend ReadMe	209
N	License File Signing ReadMe	211
O	Checkstyle Ruleset	213

Figures

2.1	Cumulative flow diagram of SCRUM (snapshot taken on May 11, during the sprint period of April 28 – May 12.	12
3.1	Architectural diagram for OLM.	25
3.2	Applied chain of trust diagram.	29
4.1	Risk management summary figure [3, p. 120].	34
4.2	Bowtie model for spoofing threats.	52
4.3	Bowtie model for integrity threats.	53
4.4	Bowtie model for Repudiation threats.	55
4.5	Bowtie model for Information disclosure threats.	56
4.6	Bowtie model for Denial of Service threats.	57
5.1	Initial PoC Architecture.	61
5.2	NMS API Overview.	67
5.3	Screenshot from front end website (NMS part).	83
5.4	Screenshot from front end website (LFA part).	83

Tables

1.1	Result goals of the project.	5
1.2	Development Roles Summary.	8
3.1	Example of XOR cipher encryption.	19
3.2	AES S-box [15].	20
4.1	DREAD category ranking.	35
4.2	Average-score-to-ranking conversion using DREAD modelling [25, p. 215].	36
4.3	NTNU assets evaluation table [28].	40
4.4	Assets CIA classification.	41
4.5	Security Review Actors & Roles.	43
4.6	Summarization of threats using DREAD modelling.	47
4.7	Value tuples for threat matrix visualization.	51
4.8	Threat matrix.	52
5.1	NMS API Endpoint Summary.	68
5.2	LFA API Endpoint Summary.	78
5.3	Proof of Concept Services Ports.	86

Code Listings

5.1	Entire LFS bash script utilizing OpenSSL.	65
5.2	NMS Source Code Tree.	66
5.3	NMS API Spring Web Endpoint Handler Shell.	68
5.4	NMS API Positive response body example.	69
5.5	NMS API Negative response body example.	69
5.6	NMS API /license/lfa/ endpoint body structure.	69
5.7	NMS API /lfa/ endpoint response body structure.	70
5.8	NMS API /lfa/licenses endpoint response body structure.	70
5.9	NMS API /pool/all endpoint response body structure.	71
5.10	LFA alive check from NMS.	72
5.11	Cryptography.generateKey.	74
5.12	Cryptography.applyCipher.	74
5.13	NMS Pool: Property change listener definition.	75
5.14	NMS Logging file example.	77
5.15	LFA Source code tree.	78
5.16	LFA API / endpoint response body.	79
5.17	LFA API /licenses endpoint response body.	79
5.18	LFA API /consume endpoint response body.	79
5.19	Chain of trust verification using the root and intermediate certificate.	80
5.20	Integrity validation of sub level licenses.	81
5.21	A simple client for sending a PUT request to NMS.	82
5.22	Calls to NMS endpoints using Axios.	84
5.23	Top-Level License example.	85
5.24	Sub-Level License example.	86
5.25	NMS source code repository CI/CD implementation (simplified).	89
5.26	LFA source code repository CI/CD implementation (simplified).	90
O.1	NMS Checkstyle Rulesheet.	213

Acronyms

AES Advanced Encryption Standard. xv, 19, 20, 27, 73

CA certificate authority. 22–24, 26, 28, 40, 41

CIA Confidentiality, Integrity and Availability. xv, 33, 34, 40, 41

DRM Digital Rights Management. 17, 18, 29, 101, 104

LFA License File Aggregator. xiii, xv, xvii, 13, 14, 23–28, 30–32, 39–43, 45, 46, 54, 56, 62–64, 67–72, 75–87, 89, 91–93, 95, 103

LFS License File Signer. xvii, 65, 84

NMS Network Management System. xiii, xv, xvii, 1, 2, 5, 13, 14, 23–28, 30–32, 39–46, 54–56, 62–73, 75–77, 79–87, 89, 91–93, 95, 103

OLM Offline License Management. xiii, 3, 13, 18, 23–25, 27, 28, 30, 32, 43, 57–59, 61, 62, 64, 85, 95, 101, 103

PoC Proof of Concept. xiii, xv, xxii, 1–5, 7, 8, 13–15, 61, 62, 64, 65, 70, 71, 73, 76, 79, 80, 82, 84–86, 88, 90–93, 95–98, 100

SLL Sub-Level License. xvii, xxii, 2, 5, 18, 24, 26, 31, 39, 40, 43, 45, 46, 54, 56, 62, 65, 68, 69, 71, 77, 80–86, 92, 103

TLL Top-Level License. xvii, xxii, 2, 5, 18, 24, 26, 27, 30, 31, 39, 40, 42–46, 48, 53, 54, 62, 64, 65, 68, 69, 71, 82, 84, 85, 92, 101, 103

UI User Interface. 82, 83

Glossary

API Application program interface, used to communicate with an application, as a point of contact for a software. xiii, xv, xvii, xxii, 13, 49, 63, 64, 67–72, 76, 78, 79, 82, 84, 86, 91

Bowtie model A bowtie-shaped diagram that visualizes the risk with a clear differentiation of proactive and reactive risk management. xiii, 34, 37, 39, 52, 53, 55–57, 59

certificate A unique file that binds an identity using the public key infrastructure. See Section 3.2.2 – Digital Certificates for more information. xvii, xxi, 13, 22–24, 26–32, 55, 63, 77, 80, 84, 85, 90

certificate authority A trusted entity that provides digital certificates to verify the legitimacy of people or organisations. xix, 22, 23, 26, 28, 49, 50

chain of trust A series of certificates and key-derivatives ensuring trust within digital security systems. See Section 3.2.2 – Digital Certificates for more information. xiii, xvii, 13, 23, 24, 26, 28–30, 42, 45, 49, 50, 55, 80, 85

cracked software Illegitimately unlocked or accessed licensed software. 17

cryptography The practice of concealing/unveiling data through systematic scrambling. 7, 18, 19, 22, 23, 62, 73, 74

Digital Rights Management Management or orchestration of legal access to, often licensed, digital assets, contents, or products. 2

digital signature A technique of validating data origin through a signature and the alleged source's public key. See Section 3.2.1 – Cryptography for more information. 22, 23, 27

DREAD A framework used to evaluate various threats by rating them on an ordinal scale. xv, 33–36, 38, 43, 46, 47, 51, 59

hash The product of the action of hashing. A unique fixed-size value representing the data inputted, regardless on input size. A specified hash function must be used. 26, 71

license functionality The functionality contained in the licenses, which are distributed through Top-Level Licenses (TLLs) & Sub-Level Licenses (SLLs). The functionality may be different things, such as streaming or conversion for media companies or usage time for software comapnies. xxii, 26, 62, 71, 77, 101

licensing company A company that issues software licensing to their customers. 2, 3, 23, 24, 26–28, 30, 32, 39, 40, 42–44, 47, 48, 54, 55, 57–59, 64, 65, 85, 91, 101, 103

media function In the Proof of Concept, Media Function has become the name of the license functionality of the implementation. The PoC has been implemented under the guidelines of Nevion, making mediafunctions a relevant name. 62, 68, 69, 71, 75, 76

Nevion Thesis cooperation company, whom the thesis is written on the behalf of [1]. xxii, 1, 2, 4, 6, 12, 23, 24, 62, 95, 100, 103

private key The private side of an assymetric key pair. Private keys can be used to encrypt data and to prevent repudiation. 20–24, 26–28, 30, 40–42, 44, 45, 47, 49, 50, 52, 55, 59, 65, 76, 80, 84, 87, 101

public key The public side of an assymetric key pair. Public keys can be used to encrypt and decrypt data. xxi, 20–24, 26–28, 73

RESTful API Architectural style for an application program interface (see API, where HTTP(S) requests are being used for accessing data. 64, 67, 91

STRIDE A threat model used to identify potential vulnerabilities and threats in a product. 33, 34, 36, 38, 44, 57, 59

threat actor The actor which is the threat. 2, 3, 31, 41–50, 52, 54, 55

Chapter 1

Introduction

1.1 Background

Nevion Europe AS is a company that makes virtual media production equipment [1]. Their focus is on transporting and processing media in real-time. One of their products, Virtuoso, is a software defined media processing node which works together with another system called VideoIPath, which is an Network Management System (NMS). The Virtuoso has different functionality such as compression, decompression and video conversion. These functions are provided by interchangeable cards, which can be inserted into the Virtuoso. The VideoIPath controls and manages all the Virtuosos in the network and assigns them tasks or connects them together using software patching [2]. An example of where such a system is applicable is a football stadium, where several camera streams need to be gathered at one place in order to be transcoded, compressed, and sent to the broadcaster's headquarters before being broadcasted to televisions at home.

Some of Nevion's customers have expressed a wish for licenses that unlock functionality for only a set amount of time, as opposed to the current lifetime license. This is mostly due to the fact that they only utilize the unlocked functionality for short periods of time and that the price of lifetime licenses is high. Nevion has considered the feedback, and expressed their wish of using the VideoIPath as a license manager, with responsibilities such as distributing licenses and keeping track of unallocated time left. This restructuring is also motivated by their wish to expand to customers who may not have the financial means to afford lifetime licenses.

Given the uncertainty of such a system, Nevion has requested a concept to be developed and with it, a security analysis, in order to find eventual security flaws. Furthermore, Nevion has expressed a wish for a Proof of Concept (PoC) to be developed with certain requirements.

1.2 Problem Area

This thesis addresses a complex problem encompassing multiple areas of concern such as license communication, license integrity, system integrity and license usage management. The purpose of the project is to develop an offline system, which will enable digital rights owners – such as Nevion – to protect their assets in offline environments. By using this project to develop a suggesting solution to the problem presented, a secure offline Digital Rights Management solution can be developed and researched. This will also gain the end users, as they will be offered digital rights protected assets in offline environments. By using this project to also research the problem area, a basis for further development in other, future, projects can be presented.

Firstly, ensuring the integrity and security of an information system is crucial, especially when in a closed offline network outside the premises of the licensing company. This involves challenges such as safeguarding confidential information, hiding secrets effectively, and preventing tampering with the software and physical hardware by threat actors seeking financial gain. Additionally, there are significant concerns, such as preventing license misuse within a customer's network and distribution to other customers to divide licensing costs amongst each other. Failure to address these challenges could lead to financial setbacks for the licensing company.

1.3 Problem Statement

The problem addressed in this thesis is formulated and written by Nevion. Their current system operates on a closed offline network and consists of an Network Management System (NMS) – VideoIPath – and one or more Virtuosos. As of today, Nevion offers lifetime licenses to their customer for their services. However, there is a demand for a more flexible licensing model that allows customers to purchase licenses and divide them into smaller licenses on demand, to be used within the customer's network. This involves the entire process from the customer receiving a bulk license, henceforth known as Top-Level License (TLL), from the licensing company, to the distribution of a smaller, divided license, henceforth known as Sub-Level License (SLL), within the customer's network. Furthermore, the objective is to implement a Proof of Concept (PoC) in Java and C++, showcasing a working example of the proposed solution, and perform a security review of the concept. The concept should address the challenges mentioned in Section 1.2, such as ensuring sufficient security and alleviating concerns of license generation in a closed offline network.

1.4 Scope

The scope of this project includes developing a concept solution for the described offline digital rights management system, henceforth named Offline License Management (OLM). This involves designing and defining the necessary components to implement a functional solution. The implementation will also include a Proof of Concept (PoC), which serves as an example of how the concept can be put into practice.

There are certain aspects that fall out of scope and will not be considered. This includes securing the source code and considering the security of the hardware, where Virtuoso software is run. Furthermore, the authentication process of system users is also excluded from the scope, as it is not necessary for the system to work independently, although it is strongly encouraged in a production environment of the concept.

In addition to the concept development and PoC implementation, a security review will be conducted. The purpose is to assess the level of cybersecurity within the OLM system and identify its risks. The security review will primarily focus on evaluating the concept and its overall security, rather than focusing on the limitations of the PoC.

1.5 Assumptions

The assumptions are set as a basis for the development of the OLM-concept, also valid for the security review. Firstly, the proprietary hardware on which the Virtuoso is deployed on, is assumed safe, as it is not available for customer insight and manipulation. As a result of this, that aspect of the OLM concept is exempt from the scope of the security review.

Secondly, throughout the project, the customer has been assumed as not trustworthy, as the company will need to secure the concept against customer misuse. This has had a major impact on the security implementations of the PoC and the security review. This means that external threat actors have taken a back seat in regard to security in the concept.

Finally, the licensing company is assumed to be a trustworthy organization where satisfactory cybersecurity and physical security measures are in place. This does not exempt it from being a target of Spear Phishing, Social Engineering, and similar attack vectors in the security review [3, p. 72, p. 426].

These assumptions have been taken into account while developing the concept and implementing the PoC, although to a varying degree; as with the specific implementation, compromises followed. Given enough time and resources, these compromises can be avoided.

1.6 Goals and Constraints

1.6.1 Constraints

Customer Internal Network

All hardware and software will be deployed within the customer's own internal network with direct physical access. Root access is also allowed for certain elements of the system.

Offline

Neither the concept nor the PoC source code will include any internet connectivity, as prescribed by the problem statement in Section 1.3. The networks in use will also be local, set within a venue or given premises.

Time

In the course of the project, the group has been given a limited time to develop and review the concept, implement the PoC, and write the thesis. The amount of time given is between 550 and 650 working hours per group member, divided over the course of 20 weeks. This results in approximately 1 800 hours total for the group. The deadline is the 22nd of May 2023.

Economics

The group does not have access to any additional funds other than the most necessary, small, purchases. This means that the group is constrained from the ability to hire external services or buy tools. The most necessary purchases will be covered by Nevision.

1.6.2 Result Goals

Table 1.1: Result goals of the project.

Thesis Components	Result Goals
The Concept	<ul style="list-style-type: none"> • A general concept is developed, adaptable by organizations such as Nevion. • A graphical architectural model of the concept.
The Proof of Concept	<ul style="list-style-type: none"> • The PoC will be based on and fulfil the concept. • It will demonstrate the feasibility and effectiveness of the concept. • The software will be secure when running offline. • It will be hard to exploit the source code for financial gain. • The NMS will be able to create Sub-Level Licenses. • The Top-Level Licenses will only be read once. • The Top-Level License is not reproducible and signed by the customer.
The Security Review	<ul style="list-style-type: none"> • A comprehensive list of discovered threats. • An in-depth ranking, modelling, and visualization of threats. • Various prevention and mitigation tactics for the threats. • An estimated risk appetite and residual risk, both compared and defined.

1.6.3 Effect Goals

- The customer will save money and experience improved customer satisfaction and trust in the system.
- Nevion and other organizations will be able to implement and utilize functionality based on the concept.
- Nevion will have increased confidence in the security of the system running within the customer's internal network.
- The customer will be able to utilize the software and gain the intended advantages of the software, with a high degree of usability.

1.7 Target Audience

The target audience for this report are technically educated individuals and organizations. We assume the reader has a background within IT with a sufficient understanding of technical terms, concepts, and programming languages. The target audience may include individuals with any cybersecurity or software security experience and knowledge, but this will not be necessary in order to understand and make use of the report.

1.8 Group Background

Every member of the group has nearly three years of education within computer science at NTNU Gjøvik. Whereas Ivan & Sander is completing a bachelors degree in Computer Science, Kristian is completing a bachelors degree in Programming. In addition to this, Kristian has been attending the course Software Security, while Ivan and Sander has been attending a study program with a specialization within programming and cybersecurity. The academic overlap between the two study programs is sufficient to allow the students to work together on a shared thesis of this matter. When the project started, Sander had been working at Nevion for approximately six months, making it easier for the group to get a grasp of the requirements and the level of security necessary.

1.9 Project Structure

1.9.1 Project Roles

Group Structure and Hierarchy

The group is aiming for the internal group structure to be as flat as possible. This means that there will be no-one in the group who has a higher degree of decision-making power than the others, except when the group rules and routines states it. This is because a fully flat structure is impractical, for example, in the event of an internal conflict. For this reason, a group leader is necessary.

The group leader is Sander Osvik Brekke.

Meeting Minutes

During the course of the project, there will be a number of meetings taking place. To make sure all these meetings have their corresponding meeting minutes, the group will have a meeting minutes responsible. The responsibilities consist of writing the meeting minutes, or delegating this responsibility when necessary, and making sure the meeting minutes are stored compliantly.

The meeting minutes responsible is Ivan Norderhaug.

Documentation

While working with the bachelor thesis, there is a large amount of documentation required. The documentation responsible is responsible for making sure the correct documentation is being created according to the requirements, within the related deadlines, and that the created documentation is being stored compliantly. The documentation responsible is also responsible for making sure the required hand-ins are being handed in within the set deadlines, even though it is the group's common responsibility to make sure the products and material needed to hand in is done within the set deadlines.

The documentation responsible is Sander Osvik Brekke.

Process Framework

The process framework chosen for this bachelor project is Scrum. For the process framework Scrum to work optimal, a Scrum master is necessary. This is a responsibility that, amongst other things, consist of leading the scrum meetings, both the sprint planning, sprint review and retrospective.

The Scrum master is Kristian Røren Svanholm.

1.9.2 Technical Areas of Responsibility

Cryptography Researcher

The largest part of the project is developing a concept using cryptography. The main responsibility of the cryptography researcher is researching cryptographic solutions for the concept development. Additionally, the responsibility includes making sure the necessary progress is present, making sure the concept follows the given requirements, making sure the necessary research is performed and making sure the tasks this development requires are delegated within the team.

The concept responsible is Ivan Norderhaug.

Proof of Concept

A part of the project description, thus also the thesis, is an implemented Proof of Concept (PoC). The PoC responsible has the responsibilities of making sure the PoC is developed according to requirements and making sure the tasks necessary to make the PoC complete are being delegated within the team.

The PoC responsible is Kristian Røren Svanholm.

Development roles

Underneath the PoC responsibility role, there are developers. Different parts of the PoC is written in two different languages, Java and C++, where the team members have different knowledge in the two programming languages. As a result of this, team members will have different development roles.

Table 1.2: Development Roles Summary.

<i>Kristian</i>	C++ developer
<i>Sander</i>	Java developer
<i>Ivan</i>	Java developer

1.10 Report Structure

Chapter 1 – Introduction

In this chapter, the task description and scope of the project is introduced, and the goals and objectives are outlined. Context and preliminary work is also presented.

Chapter 2 – Development Process

Development Process describes the development process of the project, including process framework, sprint summaries, and the different meetings throughout the project.

Chapter 3 – Concept Review

The Concept review presents and discusses the conceptual framework with its design and theory.

Chapter 4 – Security Review

The security review describes aspects of the proof of concept (PoC) in order to identify potential threats and develop prevention and mitigation strategies for them. The review also discusses the viability of the concept from a security standpoint.

Chapter 5 – Proof of Concept

Proof of Concept describes the real-world implementation of the concept from the previous chapter and its different modules. It also discusses different design decisions that differ from the prescribed concept and following security challenges.

Chapter 6 – Discussion

The discussion chapter discusses the project's outcomes, contributions and its overall viability. In addition, a summary of future work is described here.

Chapter 7 – Conclusion

The conclusion describes the objective results of the thesis and concludes the thesis.

Chapter 2

Development Process

2.1 Development Model

The development model SCRUM was used during the thesis project. This was meant to allow for a more granular and systematic approach to completing bite sized tasks organized into sprints with a larger enveloping goal.

SCRUM was organized with an Atlassian Jira¹ instance, which allowed for quickly organizing new sprints and moving issues between the different columns on the accompanying kanban board. Within the board, any issue could exist under any of five different columns, which signified the state of the issue. The columns were as follows:

To do	Work has not yet started on this issue.
Working on it	Work is under way on this issue.
Help needed	The assigned individual requires assistance issue.
Review	The issue is ready for group member to review
Done	The issue has successfully been completed.

Throughout the utilization of SCRUM, a few rules defined in Appendix C – Project Plan were utilized to minimize misunderstandings and double work. Double work entails a situation where an issue has not been moved out of the “To do” column even though someone has started to work on it. By not moving it, someone else might also “take” the job, which results in two items of work being completed where only one is necessary. The list of rules can be found under Section 4.1 in the appendix for the Project Plan.

¹<https://www.atlassian.com/software/jira>

2.2 Meetings

Throughout the project period, there have been held three main categories of meetings. Firstly, there have been held weekly meetings with the project supervisor which mostly regarded project status, questions about future steps, and feedback from the supervisor.

Secondly, there have been held bi-weekly meetings with Nevion's representative, which have been more focused on technical details and updating the company of the group's progression. This meeting has also been utilized for technical questions to gain a deeper understanding of the problem area of the thesis.

Finally, sprint planning, sprint review, and sprint retrospective meetings have been held. These have only been attended by the group members and regarded the organization and progression of the group's chosen development model, SCRUM.

2.3 Scrum Sprints Summary

Throughout the project, the group adhered to a consistent weekly sprint cycle as originally planned. The group held consecutive sprint-review and sprint-planning meetings on Friday mornings for the completed sprint and the upcoming week's sprint. The sprint-planning meetings allowed for planning of which items that needed to be worked on, while the sprint-review meetings allowed for reviewing the success of the current sprint. Additionally, a sprint retrospective meeting was held every three weeks to discuss the success of the last three sprints and address any issues that were discovered.

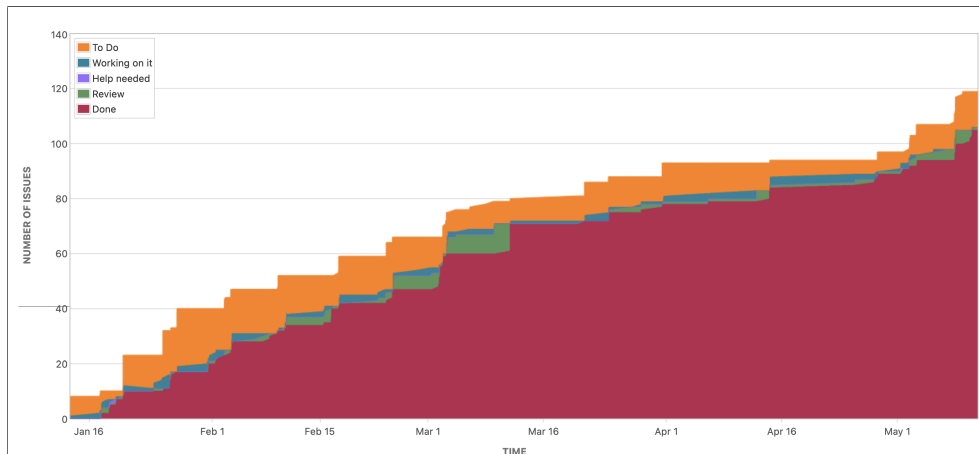


Figure 2.1: Cumulative flow diagram of SCRUM (snapshot taken on May 11, during the sprint period of April 28 – May 12).

The cumulative flow diagram in Figure 2.1 depicts the progress of weekly sprints over time. The different colours within the diagram represent the amount of issues

in various states at that moment in time. Orange means the issue has not yet been started on, and red means the issue is completed. As is very readily apparent by the 'staircase' look of the figure, the group first worked in weekly sprints and then later in bi-weekly sprints as the plateaus of the 'staircase' gets longer. This cumulative flow diagram gives a deep look into how the group has generated new work items at the beginning of each sprint in the sprint planning meeting and throughout the sprint have slowly moved the different issues from 'To do' to 'Done'.

#1 – Project plan first draft

13/1 – 20/1

The first sprint of the project consisting of writing the first draft of the Project Plan, constructing a group agreement and creating the Gantt chart for the project.

#2 – Project plan refinement

20/1 – 27/1

This sprint was oriented around refining the Project Plan, beginning the research and initializing the development environments for the PoC.

#3 – Project plan final delivery

27/1 – 03/2

Sprint #3 was about creating a first draft of the structure for the final thesis, implementing API communications for both LFA & NMS and updating & delivering the project plan.

#4 – Infrastructure development 1

03/2 – 10/2

Sprint #4 concentrated on setting up basic license pool functionality for both NMS & LFA. Research was also done on Java code obfuscation and the license file format provided by Nevion.

#5 – Infrastructure development 2

10/2 – 17/2

Sprint #5 consisted of adding OpenSSL, HTTPS & certificate verification within the LFA and implementing integrity functionality, networking, and chain of trust functionality for the NMS. This was the first sprint with unfinished issues at the end, as one issue was under-estimated in the sprint-planning meeting at the start of the sprint. In total two large and one medium issues were left unfinished which all regarded the implementation of OLM in the NMS.

#6 – Testing and tools

17/2 – 24/2

The sixth sprint in our project was about adding unit testing to the codebase and creating a control panel to interact with the system. In the end, three medium-sized issues regarding unit testing for the NMS was never completed and went back to the backlog.

#7 – Testing and LFA verification

24/2 – 03/3

Sprint seven was one of the larger sprints, with thirteen finished issues and one unfinished issue. During of which the codebases came to a good enough completion that the group considered them as a minimal viable product and NMS unit testing was further expanded. The PoC control panel was also further expanded with the ability to generate sublicenses from available NMS license pool and viewing all the connected License File Aggregators (LFAs) respective license pools.

#8 – Testing and code completion

03/3 – 10/3

Sprint #8 consisted of finishing the codebase to a more stable and feature complete state, which was going largely to plan. Only one issue was started on and not finished, wish was about adding testing to the LFA. After this sprint the codebase has largely been untouched as it has reached the status of PoC.

#9 - Exam sprint

10/3 – 17/3

During this sprint, two of the group members were busy with an exam in another course, so the last member was largely left to their own devices and worked on the thesis.

#10 – Security review writing 1

17/3 – 24/3

Sprint #10 was the beginning of the work on the Security Review. The chapter would turn out to be one of the most tenacious chapters in the thesis, taking an entire months worth of sprints to reach a first draft. Five issues were created, of which two were finished.

#11 – Security review writing 2

24/3 – 31/3

Similarly to sprint #10, sprint #11 was about writing the Security Review and saw several key pillars of the review take its first form. However, this sprint was also slow as several aspects of the review had to be debated several times over to find a suitable answer the group were happy with. In the end, four out of six issues were completed during this sprint.

#12 – Security review first draft

31/3 – 14/4

Given the previous two sprints' rate of completion, it was decided that sprint #12 would be two weeks long. This turned out to be a good decision, as a first draft of the Security Review was completed and ready for review by the thesis supervisor at the end of the sprint. From here on out, all sprints were two weeks long as a standard.

#13 – Process, theory, PoC

14/4 – 28/4

Sprint #13 regarded doing a lot of work in several parts of the thesis and saw sizeable restructuring. The process section was finalized for the first draft, the theory section was combined with the concept section into 'Concept review' which saw a lot of work and the PoC section was nearly finalized for the first draft. At the end of this sprint the thesis document had reached 20000 words and 79 pages without the appendix. In the end all but one issue regarding the PoC was finished which was sent into the next sprint for finalization.

#14 - First Draft

28/4 – 12/5

Sprint #14 saw the completion of the first draft and several more quality updates to the final report. The group received valuable feedback from the thesis supervisor and revised several aspects of the document. The several code-bases of the project also received updates to their README files with updated knowledge and explanations. Overall, a lot was finished, and the thesis had reached a high level of maturity.

#15 – Final delivery

12/5 – 26/5

Sprint #15 was the sprint where detailed feedback was received and applied, proofreading was performed, and other detailed, minor, alterations were applied. During the sprint, the thesis was also exported and handed in as the final report.

Chapter 3

Concept Review

This chapter aims to present, describe, and discuss the concept developed during the course of the project. The concept will be presented in terms of components, composition, requirements fulfilment, and functionality, and discussed by the same terms and aspects. Security aspects of the concept will also be presented.

3.1 State of Technology

In the field of offline Digital Rights Management, several approaches and technologies have been developed to address the unique challenges posed by closed offline networks. This section provides an overview of the state of the technology in this domain.

The problem domain of this project is close to the problem domain of offline Digital Rights Management (DRM). DRM is a way of managing several things in relation to digital media, for example managing access, redistributions, and utilization [4]. An example of this, is the sharing of paid stock photos.

Computer software products that require a license, but do not need an internet connection, are also handling Digital Rights Management. An example of this is the game Farming Simulator 2022 ¹. This game, and several other similar computer software products, experience the spread of *cracked* versions of their software [5]. A cracked version of a computer software product, often called cracked software, is a piece of software that has got its copyright protection, license requirements, or similar removed [6]. The fact that software products, often created by large software companies, have issues with cracked software versions signals the fact that the problem domain is still relevant, and no general – completely secure – solution have been created.

Some companies have created computer software products which do not suffer

¹<https://www.farming-simulator.com/>, visited April 28th, 2023

from distributed cracked versions. What is common between these products is the fact that they require an internet connection in order to be run. This allows for the license information and other pieces of software legitimacy information to be checked with a trusted central server before the software is run. An example of this is computer games with DRM solutions created by companies such as Denuvo [7].

Spotify and Netflix are two examples of computer software products that allow the end user to stream digital media through a subscription-based license^{2,3}. The two services also allow their end users to download media to their devices, in order to listen to music or watch series & movies while offline. Their solution to offline Digital Rights Management is to make it online; at certain intervals, the user is required to connect to the internet in order to maintain access to their downloaded media [8][9]. For Spotify, this interval is set to 30 days, which is also the subscription interval of the license; this way, it is not possible for an end user to cancel their subscription and keep access to their media, which is offline. For Netflix, the online connection requirement is set to irregular intervals, as it varies by the media which is downloaded. If the end user does not connect to the internet within the required time, the downloaded media will be voided. The downloaded media will also be voided if the end-user alters the time and date of the mobile phone manually, and the phone is no longer getting its current time from an NTP server [10].

Common DRM solutions utilize techniques such as cryptography and hashing, which the OLM system also relies on. However, most similarities end there, as these other solutions also make use of an internet connection, either at certain intervals or constantly, to enforce their integrity. This makes the solutions included in the compared technologies hard to adapt for this concept, as a completely offline environment is a requirement. This project includes the aggregation of TLLs and generation of SLLs, which makes it further different from the examples of technologies and solutions, as the examples does not handle this. However, the examples should still be deemed relevant, as they are examples of computer software handling licenses and digital rights in an – often – offline environment.

²<https://support.spotify.com/no-en/>, visited April 28th, 2023

³<https://www.netflix.com/no-en/>, visited April 28th, 2023

3.2 Theory

3.2.1 Cryptography

Cryptography is a practice used to secure sensitive information from unauthorized access, tampering, and interception. By utilizing mathematical algorithms, it provides a range of security services, such as confidentiality, integrity, and authentication. One common approach is symmetric key encryption, which employ the same key for both encryption and decryption. The plaintext is converted into ciphertext using mathematical operations with a secret key, which can also retrieve the original plaintext when used in reverse [11].

An early form of encryption, is the substitution cipher, which involves substituting each letter in a plaintext with another letter based on a fixed rule. The Caesar cipher is a well known example of this, in which each letter in the plaintext was replaced with another letter, x number of positions down the alphabet [12].

One of the most widely used operations in symmetric key encryption is XOR (exclusive OR), a binary operation that outputs a 1 if the input bits are different from each other and a 0 if they are the same [13, p. 32-34]. To securely encrypt a message using the XOR operation, a key that is at least as long as the plaintext needs to be present. This is because a smaller repeating key could easily be broken using frequency analysis [14]. As shown in Table 3.1, the process involves XOR-ing each bit of the plaintext with the corresponding bit of the key.

Table 3.1: Example of XOR cipher encryption.

Plaintext	H	E	L	L	O
Plaintext ₍₂₎	01001000	01000101	01001100	01001100	01001111
Key ₍₂₎	00000001	00000010	00000011	00000100	00000101
XOR	01001001	01000111	01001111	01001000	01001010
Ciphertext	I	G	O	H	J

Today, the most widely used symmetric cipher is the Advanced Encryption Standard (AES). The algorithm employs a block cipher⁴ which operates on fixed-size blocks. The block size is determined by the key size, e.g., 128-bit blocks for AES-128. The key size can either be 128, 192 or 256 bits, with bigger key sizes offering greater security. The AES algorithm consists of many rounds of substitution and permutation operations that convert the plaintext into ciphertext. For each round, a *round* key is derived from the original key using a key expansion algorithm. The key is then XOR-ed with the plaintext block before being subjected to many substitutions and permutations. During the substitution step, each byte of the input block is replaced with the corresponding byte from a pre-defined lookup table known as the S-box (e.g., Table 3.2) [16]. The S-box is designed to withstand a variety of cryptographic attacks, such as linear and differential cryptanalysis [15].

⁴A cipher that encrypts an entire block of plaintext bits at a time with the same key [13, p. 30].

Table 3.2: AES S-box [15].

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
10	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
20	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
30	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
40	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
50	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
60	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
70	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
80	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
90	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a0	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b0	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c0	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d0	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e0	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f0	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

During the permutation step, also known as the shift rows step, the bytes of the input block are rearranged according to a pre-defined permutation pattern. This ensures that the output is resistant to attacks relying on the ordering of bytes within the input block. Additionally, the AES algorithm has a mixing step. It involves performing a matrix multiplication operation on the columns of the input block, which provides diffusion of the plaintext and strengthens the security of the algorithm [16].

Given the strong security properties of the Advanced Encryption Standard (AES) and its resistance to known attacks, it presents itself as a viable choice for securing sensitive data.

Asymmetric key encryption involves two distinct keys, a public key and a private key, for encryption and decryption respectively. RSA is a frequently used asymmetric encryption method in secure communication protocols. Ron Rivest, Adi Shamir, and Leonard Adleman developed RSA in 1977 and named it after their initials. The algorithm generates both a public and a private key. The difficulty of factoring huge prime numbers is the foundation of RSA's security, as a big composite number is thought to be computationally impossible to factor into its prime elements in a practical timeframe [13, p. 174].

Key Generation [13, p. 175-176]:

To generate an RSA key pair, the first step is to choose two large prime numbers p and q . These should be chosen randomly and should be half the bit length of n . The modulus n is then computed by multiplying the primes together.

$$n = p \cdot q$$

The totient function of n , represented by $\phi(n)$, is then computed.

$$\phi(n) = (p - 1)(q - 1)$$

Then an integer e is chosen, which must satisfy the condition where the greatest common divisor of e and $\phi(n)$ is 1:

$$1 < e < \phi(n) - 1, \\ \gcd(e, \phi(n)) = 1$$

After e has been chosen, the integer d must be computed such that d is the multiplicative inverse of e modulo $\phi(n)$:

$$d \cdot e = 1 \pmod{\phi(n)}$$

Output:

The public key is the pair of integers (n, e) , which can be shared with anyone.

$$\underline{\underline{k_{\text{pub}} = (n, e)}}$$

The private key is the integer d , which is kept secret.

$$\underline{\underline{k_{\text{pr}} = (d)}}$$

Encryption

To encrypt a message M using the recipient's public key, the message is raised to the power of their public exponent e and modulo the recipient's public modulus n , which results in the ciphertext C [13, p. 174-175]. This can be expressed with the following equation:

$$C = M^e \cdot \text{mod } n,$$

Decryption

To decrypt a message using the recipient's private key, the ciphertext C is raised to the power of their private exponent d and modulo the recipient's private modulus n , which results in the message M [13, p. 174-175]. This can be expressed with the following equation:

$$M = C^d \cdot \text{mod } n,$$

Integrity

Apart from encryption, cryptography also facilitates the verification of data integrity through the use of digital signatures. The Basic RSA Digital Signature Protocol is one example of this [13, p. 265]. In this protocol, a signer uses their private key to compute the signature s of the message x as:

$$s \equiv \text{sig}_{k_{pr}} = x^d \cdot \text{mod } n$$

The message and the signature are both sent to the recipient, where the signature is validated with the signer's public key.

$$x' \equiv s^e \cdot \text{mod } n$$

If x and x' match, the recipient can be certain that the author of the message possessed the signer's private key. This ensures that the authenticity of the message, thus also the legitimacy of the message sender, is preserved.

3.2.2 Digital Certificates

An X.509 certificate is a public key certificate which binds an identity, such as an individual or organization, to a public key using a digital signature, either signed by a certificate authority or self-signed. A signed certificate, from a trusted source, allows for validation of digitally signed documents.

There are two types of certificates in an X.509 system; CA certificates and end-entity certificates. The certificate authority, either the root certificate authority or one of its children, have the ability to issue other certificates, whereas end-entity certificates serve the goal of identifying users. This hierarchical structure creates a certificate chain.

In addition to issuing certificates, a certificate authority (CA) also plays a crucial role in certificate revocation. When a certificate needs to be invalidated before its expiration date, the CA can revoke it. This is typically done in situations where the private key associated with the certificate is compromised, the certificate holder's privileges have changed, or the certificate is no longer trusted for other reasons

Through the use of a certificate chain, a chain of trust can be established. Using a process called Certificate Path Validation⁵, a target certificate's signature is checked against the next certificate's public key, and so on, until the top of the chain is reached. As the certificate at the top is the source of truth, reaching it means that the target certificate can be trusted and therefore authenticated [17].

3.2.3 Summary

In short, cryptography provides a range of security services such as confidentiality, integrity, and authentication by using mathematical operations to secure information from unauthorized access, tampering, or interception. Symmetric key encryption uses the same key for both encryption and decryption, while asymmetric encryption uses two distinct keys, a public key for encryption and a private key for decryption. Furthermore, these keys can be used to ensure integrity through the use of digital signatures. By signing a message with the private key, the recipient can verify the authenticity by checking the signature against the public key. Digital certificates are utilized for establishing authentication and sources of trust within cybersecurity systems and through the use of a digital certificate chains, trust can be established in distributed systems.

3.3 Offline License Management (OLM)

Offline License Management (OLM) is a system created for closed networks where an internet connection is not available. OLM serves as an intermediary licensing process, relying on a chain of trust between various entities. The OLM system is based on Nevion's system but expanded upon to meet the requirements mentioned in Section 1.3. From here on, VideoIPath will only be referred to as Network Management System (NMS) and the Virtuoso will only be referred to as License File Aggregator (LFA).

3.3.1 Certificate Authority

The certificate authority (CA) is the fundamental source of truth and authority within the system. It is owned, operated, and secured by the licensing company which utilizes it to sign licenses for different customers. Given the sensitivity of data stored within the CA, security is of the utmost importance. Should any single key leak, an entire customer relationship would be rendered compromised. The

⁵<https://datatracker.ietf.org/doc/html/rfc5280>

security around the CA should include both digital and physical security measures, as well as a required high security clearance for access.

3.3.2 Network Management System

The Network Management System (NMS) is a software solution which serves the purpose of acting on the licensing company's behalf. Preferably, the NMS is to be deployed on proprietary hardware provided by the licensing company as this would ensure only one instance of the software existing, thus reducing the potential exploitability of the system. However, in this specific case, the focus is on accommodating Nevion's request. Therefore, the focus will be on the NMS deployed on a customer's computer.

To enable the NMS to act on the licensing company's behalf, the system incorporates a certificate, which will henceforth be known as the intermediate certificate, derived from a trusted root. This certificate enables the NMS to sign the generated SLLs, thereby enabling their acceptance by the License File Aggregator (LFA). This is achieved by the NMS redeeming a Top-Level License (TLL) issued by the licensing company, which in turn allows the NMS to generate and sign SLLs which are subsequently issued to various LFAs across the internal network. To ensure secure operations under the specified circumstances, each customer relationship in the OLM system necessitates a root key pair. The private key is securely stored within the licensing company, while the public key is integrated into the NMS software that is distributed to each customer.

For the Offline License Management (OLM) system to function properly, it requires the secure storage of the intermediate private key, as this component is critical for the NMS to sign a license. Furthermore, due to local storage, the OLM system requires encrypted persistence at the NMS to store the active and redeemed licenses.

3.3.3 License File Aggregator

The License File Aggregator (LFA) is software that is run on proprietary hardware, and in turn, considered secure. This is where the actual functionality of a given product lies, and needs to be unlocked by NMS in the form of an SLL. The LFA requires the certificate of the root CA to be embedded in the software, which allows for verification of received licenses. As with the NMS, the LFA also requires local storage to store the active and redeemed licenses as checksums. By storing the certificates in an immutable storage space — also known as embedding — the certificates will be rendered more resilient to attacks. This acts as a preventative measure for certificate swapping, which, if happens, may render the entire chain of trust compromised, as the system relies on the LFA to act as a source of truth.

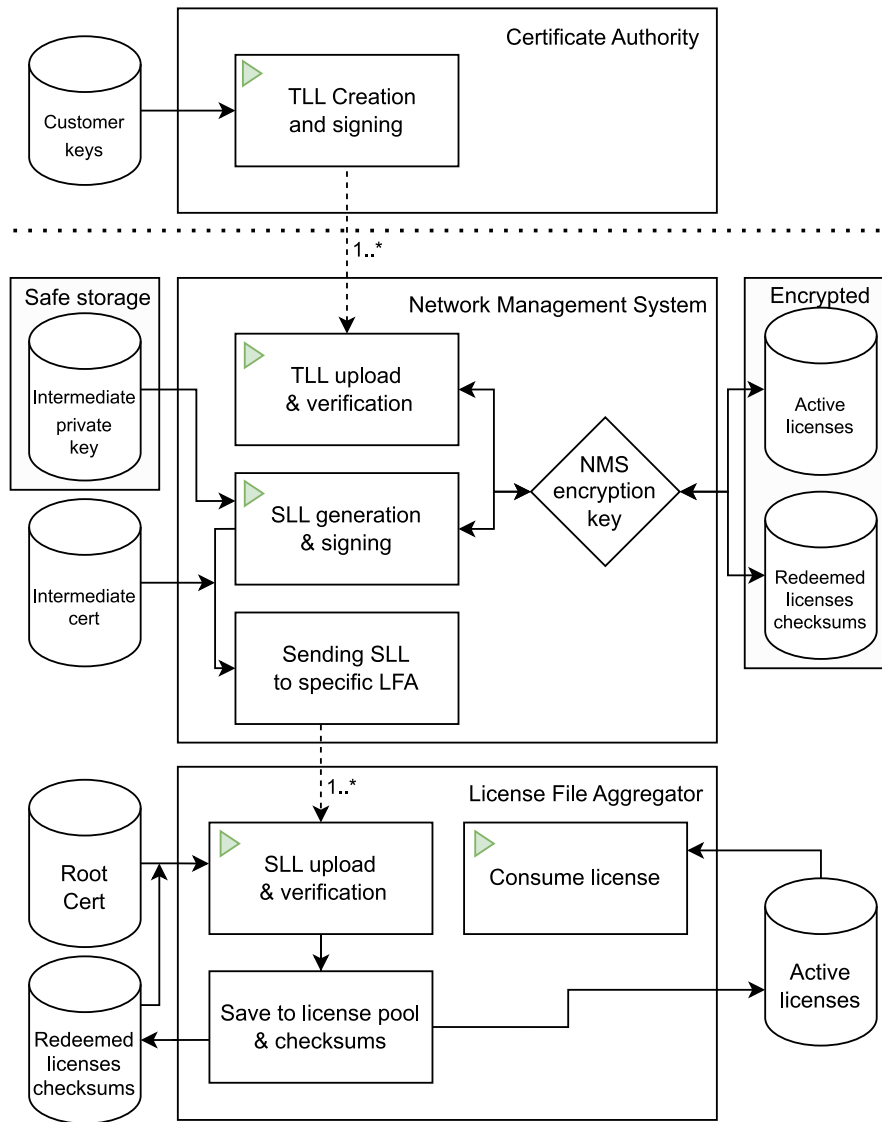


Figure 3.1: Architectural diagram for OLM.

3.3.4 Communication

To be able to make all components of the concept work, communication is necessary. Firstly, communication between the customer’s actors and the NMS is necessary to upload licenses. This can be done in numerous ways, for example through a Command-Line Interface⁶ or an Application Programming Interface⁷. In addition to this, it is necessary for the NMS to be able to communicate with the LFA. This can be done in a number of ways over a network and does not require

⁶https://en.wikipedia.org/wiki/Command-line_interface, visited April 27th, 2023

⁷<https://en.wikipedia.org/wiki/API>, visited April 27th, 2023

a particularly secure connection, as if the payload has been altered it will be detected in the LFA through the use of signatures and certificates. As a result of this, an example of the means of communication is, similar to the first required communication channel, an Application Programming Interface (API). The network in use may be any kind of network, as long as it is not connected to the internet; LAN⁸, VLAN⁹, or similar.

3.3.5 How Does it Work?

It all starts with the licensing company which maintains their own certificate authority (CA). Within the CA the licensing company keeps a series of keys, key-derivatives, and certificates per customer relationship stored. When an order for a new TLL arrives at the licensing company, the company creates a new license and then signs it using their pre-existing customer root private key. This simple technique of giving each customer relationship its own keys enforces that the new license will only ever be redeemed by that specific customer. The licensing company then sends the new TLL to the customer via their preferred means, such as an e-mail.

On receiving an ordered TLL, the customer redeems the license at the NMS, where a verification process will start. The TLL and its signature are verified against the embedded public key, where only validly signed TLLs will be parsed. If the license is verified successfully, the license functionality is added to its corresponding pool located in the NMS. During the redeeming process of the TLLs, a hash of the TLL file will need to be appended to the storage of TLL hashes if it does not already exist there. This is done to prevent multiple redemptions of the same license.

When a satisfying amount of TLLs are redeemed at the NMS, SLLs can be generated. The NMS is then able to subtract given functionality from the respective pools, generate a license file and write a corresponding signature file which results in a new SLL. When creating an SLL, an LFA unique identifier is necessary and will be added to the license file. The necessary files, including the license file, the signature, and the intermediate certificate of the NMS, is then communicated to an LFA to be redeemed there.

When the LFA receives an SLL from the NMS a series of steps are taken to verify the legitimacy of the license. First, the accompanying intermediate certificate is verified against an embedded copy of the root certificate. This ensures that the alleged intermediate certificate in fact is part of the chain of trust. If successful, the LFA continues by validating the integrity of the license by deriving the intermediate public key from the now-validated intermediate certificate. With this public key, it is possible to validate that the license signature is from the trusted intermediate source and that the signature matches the license payload. Once all these

⁸https://en.wikipedia.org/wiki/Local_area_network, visited May 10th, 2023

⁹<https://en.wikipedia.org/wiki/VLAN>, visited May 10th, 2023

facts have been established, the LFA can continue knowing a legitimate license has been uploaded.

Once the legitimacy and integrity of a license has been validated, the LFA checks the license payload for an identifying factor which only this specific LFA has attributed to it. This could be a unique ID, the device's MAC address, or similar. If the attribute matches, the LFA knows that the specific license was not created for another LFA. Finally, the checksum of the license payload is checked against a locally stored list of previously uploaded licenses. This prevents re-using licenses that match the LFA, but has already been redeemed before. Once the license is verified and redeemed, it is possible for the LFA to utilize the included functionality by *consuming* the license redeemed.

Within Figure 3.1, four boxes have green arrows in them to signify them as 'entry-points'. This means that they are starting points which engage core functionality within OLM.

3.4 Security Aspects

Integrity and confidentiality are crucial aspects of the Offline License Management, ensuring that the licensing data remains both accurate and protected from unauthorized access. By prioritizing these two security principles, the OLM system aims to maintain trust between the licensing company and the customers, as well as to safeguard sensitive information.

Integrity in the OLM system is primarily maintained through the use of digital signatures. By utilizing RSA to create unique root key pairs for each customer relationship, the OLM system establishes a strong foundation for the system's integrity. The private key, held by the licensing company, is used to sign TLLs and create intermediate certificates. The corresponding public key, stored at the NMS, is used to verify these signatures and ensure the authenticity of the licenses as explained in Section 3.2.1. In addition to digital signatures, the use of X509 certificates – explained in Section 3.2.2 – further strengthens the system by only allowing authenticated uploads. This is done by ensuring that the trust is only in the root certificate and the child certificates derived from it, which prevents other sources from being accepted.

Confidentiality in the OLM system is achieved through the use of a strong encryption algorithm, such as the AES-256 algorithm mentioned in Section 3.2.1. Its purpose is to encrypt the active licenses and redeemed licenses' checksums located at the NMS, keeping the license data protected from unauthorized parties.

3.5 Adapted Chain of Trust

OLM is built around the chain of trust standard, with certain modifications and adaptations to allow for the system to be deployed in an offline environment. As described in Section 3.2, the concept of chain of trust is built upon an intermediary certificate authority which will approve or disapprove any end entity certificates on the behalf of the root certificate authority. The Offline License Management concept includes an adapted version of chain of trust, where the division of responsibility and roles is adapted to this purpose, and the number of links is reduced. The NMS becomes an end entity of the chain, and the LFA becomes a client of the chain of trust, as seen in Figure 3.2. This way, the intermediary link of the *chain* is removed, and therefore shortened.

As seen in Figure 3.2, the intermediate key pair is distributed from the certificate authority to the end entity and the certificate authority public key is distributed to the client. This is done before the delivery to the customer, for example, by embedding the data during the system setup. This is necessary to do in this order due to the fact that the devices will not be connected to the internet, as it is an offline solution. The figure also shows the chain, as described earlier, where the licensing company is the certificate authority, the NMS is the end entity and the LFA is the client.

An important aspect is the fact that the OLM system is offline. This means that an important part of the chain of trust concept is rendered impossible; certificate revocation. If the certificate of an end entity, i.e. the NMS, is deemed invalid by the certificate authority, the client has no way of accessing this information. As certificate revocation is an important part of chain of trust, as described in Section 3.2, this is a rather large departure from the standard chain of trust concept. Essentially, the trust can only be given and never taken away. This means that the integrity and confidentiality of the NMS, its data, and private key is essential.

As opposed to most normal deployments of chain of trust, this concept attempts to protect the licensing company from malicious customers instead of the licensing company protecting the customers from other malicious users. In this way, the purpose of the utilization is inverse.

Despite the adaptations done in the concept's rendition, the concept is still regarded as a utilization of chain of trust, as the key components are still present. The key components, as described in Section 3.2, consisting of a chain of derived key pairs and a hierarchical structure of validation, can be identified. For example, certificate path validation is present, as the LFA verifies the intermediate certificate against the public certificate authority embedded in the hardware. As a result of this, it is clear that the key components are utilized in a way such that OLM can be called a rendition of adapted chain of trust.

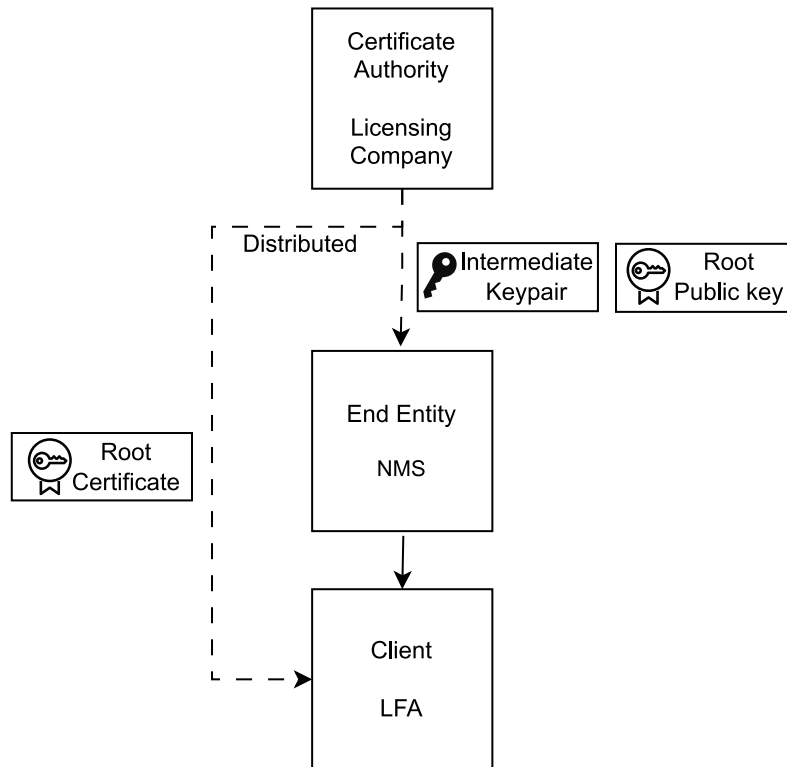


Figure 3.2: Applied chain of trust diagram.

3.6 Discussion

The concept is a comprehensive and detailed general solution to the issue of handling licenses in an offline environment. It contains solutions to and covers the problems and the requirements sketched up as a basis for the project: with an adapted version of *chain of trust*, the concept works in an offline environment; with a combination of encrypted local storage, secure storage, certificates, and keys, the concept has its integrity secured; with well-defined components and modules, an implemented instance of the concept will easily maintain a high cohesion and loose coupling between the different components.

As mentioned in Section 3.1, similar challenges are present in similar products, where different techniques have been used to solve them. Some products, for example Spotify and Netflix, utilize an “online at intervals” approach to the issue, while the DRM solutions of Denuvo have a “keep the connection online” approach. These solutions are, as mentioned, not possible to utilize in this concept, as it is required to be kept solely offline. This makes it hard to take inspiration or to look at similar technology.

When implementing a new or altered solution, there are many pitfalls to look out

for. For example, it is important to maintain the principles related to modularity¹⁰, coupling¹¹, and cohesion¹². The goals are to maintain a high degree of modularity, loose coupling and a high degree of cohesion, which has been kept in mind during the development of the concept, and is considered to be fulfilled. This is due to, for example, the division of responsibilities between the NMS and LFA, where the LFA only need to communicate with the NMS, and not any potential end user.

The security of this concept is crucial to whether or not it will cover the requirements. As a consequence of this, the security has been paramount in the development, running reviews, and iterations. As mentioned in Section 3.4, the integrity has been especially important in the effort of securing the concept among other security details, such as securing the confidentiality of important private keys within the NMS. By altering the chain of trust and effectively giving the customer control over and physical access to a source of authority, such as the NMS within the OLM system, a plethora of new issues arise. For a more detailed review of the security of the concept, please see Chapter 4 – Security Review.

3.6.1 Challenges

Although the concept covers the requirements to a satisfying degree, there are some challenges which makes it hard to perfect the concept. Firstly, when an intermediate certificate has been given to an NMS, a TLL has been sent to the customer, or an LFA with an embedded root certificate has been given to the customer, they are impossible to revoke. This means that if there is a breach in the security, where, for example, a license can be redeemed several times, it is hard to invalidate the licenses that are given to the customer. As a result, it is hard to remove the TLL the customer is redeeming several times, giving the customer, in theory, indefinite amounts of functionality time for the functionality included in the TLL. Similarly to this, if the root private key of a customer relation is leaked from the licensing company, it is impossible to revoke the intermediate certificate in the NMS and the root certificate in the LFA, making it possible to generate and validly sign TLLs. The solution to the difficulty of revoking licenses and certificates is to make it impossible for the leaks and cracks to happen. Of course, it is not possible to make it impossible, thus there are security holes. Even if it was possible to revoke these things, it would be hard to discover the misuse, and when to revoke the said certificates, licenses, or keys.

This leads us into a different challenge with the concept; it is hard to discover any potential cracks, leaks, or broken chains inside the customer's offline system. When this is the case, it is hard for the licensing company to, firstly, discover the misuse; secondly, act on the misuse; and finally prove the fact that it happened. This means that the licensing company accepts a risk when the system is deployed

¹⁰<https://en.wikipedia.org/wiki/Modularity>, visited May 2nd, 2023

¹¹[https://en.wikipedia.org/wiki/Coupling_\(computer_programming\)](https://en.wikipedia.org/wiki/Coupling_(computer_programming)), visited May 2nd, 2023

¹²[https://en.wikipedia.org/wiki/Cohesion_\(computer_science\)](https://en.wikipedia.org/wiki/Cohesion_(computer_science)), visited May 2nd, 2023

in the customer's internal network and TLLs are sent to the customer.

Due to all the security challenges within the system, a lot of impracticality is also present. Swapping a customer's certificate would require a technician from the licensing company to physically enter the premises of the customer to do their work. The balance act between security and usability is also important to take into account, as this could impact adoption. For example, if a completely secure system were to exist, it is probable that the security measures have been implemented on the cost of usability and practicality to a degree where the system is deemed impractical.

Combining the facts that revocations are not possible and that it is hard to discover any potential breaches, if any breaches first occur, it is difficult to end the misuse.

3.6.2 Alternative Solutions

A large amount of the issues that have been proven hard to solve in offline environments, are intuitive to solve in online environments. An example of this is integrity checks of the software, or source code, towards a central server. This does not necessarily require a constant internet connection, and can, for example, utilize the Spotify model mentioned in Section 3.1, where an internet connection is required on certain time intervals. When doing so, a trusted clock can also be accessed, thus making it possible to utilize time- or duration-restricted certificates. When an internet connection is made, the certificates may also be automatically swapped out, making the action of performing certificate rotations possible. The duration of the certificates should line up with the intervals of the internet connection requirements, similarly to how the internet connection requirement intervals line up with the subscription intervals in the Spotify model. Given the fact that the LFA is deemed as secure hardware, a possible solution may also be to add a trusted clock to it. This can, although, go against the principle of high cohesion and loose coupling; the NMS will rely on the clock of an LFA, while the LFA is reliant on the NMS for SLLs. This creates a circular dependency, which is unwanted [18].

For the Spotify model to work, the system infrastructure will need to be connected to network infrastructure allowing for communication with the World Wide Web. If the infrastructure is already present, why not keep the online connection running constantly? By doing so, the concept may not need to verify any licenses locally, where a central server can be used. This does, of course, open the system to a different kind of threat actors: online threat actors. The advantages are still comprehensive; the integrity of the system is ensured, making it much harder to crack the system for any gains. However, due to the danger of online threat actors, some industries prefer their hardware and networks to be entirely offline to prevent any digital attacks during broadcasting. This means that an entirely disabled or extremely limited internet connection might be required, which makes all above examples pointless.

When an offline solution is required – such as this project – a central verification server is, of course, not possible. A possible improvement of the concept, when keeping it offline, may be a physical visit from a technician from the licensing company at certain intervals. The technician could rotate and swap certificates at regular intervals. The immediate drawback of this is the fact that programmed functionality for changing certificate makes it harder to secure the certificates from the customer; the certificates are then no longer embedded.

As mentioned earlier in Section 3.3.2, it is preferable for the NMS to exclusively run on proprietary hardware, in the same manner as the LFA. This is, as mentioned, considered more secure as it would ensure only one instance of the software existing. Thus reducing the potential exploitability of the system. Furthermore, it would provide enhanced protection against unauthorized access, tampering, or software piracy as the closed nature of proprietary systems makes it harder for potential attackers to gain an advantage [19].

These are only some possible adaptations, each with their own advantages and disadvantages. When using OLM as inspiration for an implemented instance of the concept, it is always possible to perform adaptations, as it is flexible.

Chapter 4

Security Review

4.1 Introduction

This chapter aims at performing a cybersecurity review of the developed concept in Chapter 3 – Concept Review. The review will be performed as a risk management analysis, through utilizing modern and up-to-date cybersecurity analysis frameworks, modelling concepts, and visualization tools.

The security review will be limited in scope to the concept solely. This means that the threats that are identified will only be threats that are a part of the concept developed in the course of this project, and not any threats that are part of already existing systems or other out-of-scope aspects.

Furthermore, the purpose of this security review is to identify, assess, categorize, prevent, and mitigate all distinguishable security threats, before comparing the residual risk to the risk appetite defined for the concept.

4.2 Theory

This section presents the relevant theory and best practices concerning security reviews, which primarily involve the identification, assessment, and handling of risks, collectively known as risk management (see Figure 4.1) [3, p. 119]. Risk management relies on established frameworks and methodologies in the field of information security, such as the CIA triad and security frameworks like DREAD and STRIDE.

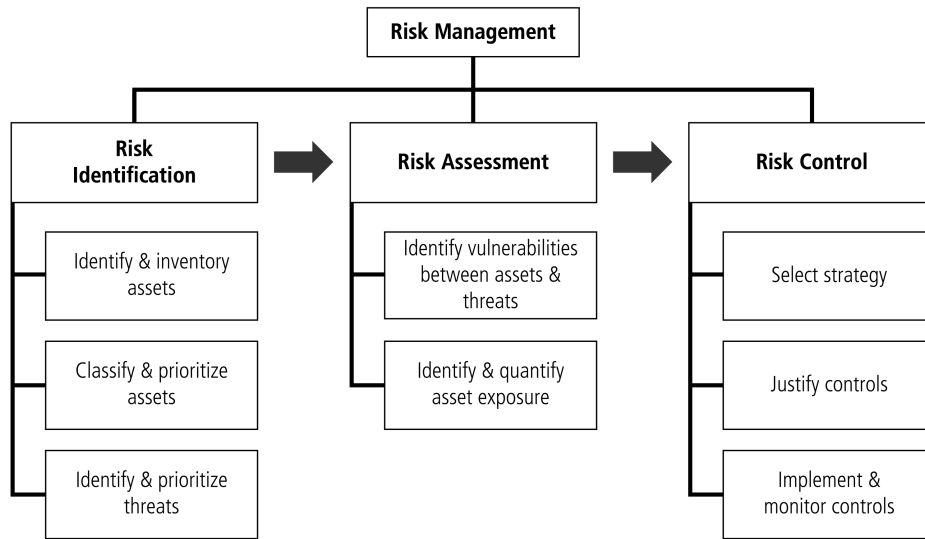


Figure 4.1: Risk management summary figure [3, p. 120].

At its core, computer security revolves around protecting valuable assets in a system, which may include hardware, user data, or source code [20, p. 8]. Consequently, the first step in efficiently protecting these assets is their identification [3, p. 122].

The CIA serves as the foundation of information security, outlining three primary objectives: Confidentiality, Integrity and Availability [21, p. 21-23].

Confidentiality	Preserve the confidentiality of sensitive information
Integrity	Maintain the integrity of data and systems
Availability	Ensure the availability of resources for authorized users

These principles form the basis for identifying critical assets. Each asset is assigned a value per attribute and ranked numerically, with those possessing high CIA values considered crucial to protect from threats [3, p. 123-125].

Subsequently, threat identification involves recognizing, describing, and categorizing threats against a security system. The threat identification process is crucial for performing further security modelling, such as DREAD or the Bowtie model [22][23].

The STRIDE framework serves as a valuable tool for identifying threats to a system and pinpointing targeted security aspects. While encompassing elements from the CIA triad, STRIDE expands the scope by integrating additional aspects, such as repudiation. The acronym STRIDE represents Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege, addressing the primary types of threats to Confidentiality, Integrity and Availability within a

security system [24].

Once risks have been identified and defined, the next step is to evaluate their severity along with the likelihood of their occurrence. This process is known as risk assessment [3, p. 140-145]. Conducting a risk assessment of identified threats enables the suggestion and implementation of more effective measures, ultimately aiming to prevent threats and mitigate potential damages [3, p. 146].

DREAD is a model used to quantitatively assess the severity of cybersecurity threats, where a scaled rating system is used to assign numerical values to five different risk categories. The five risk categories all have the same weighting, where the average of all categories per threat is treated as the severity rating for the given threat [22].

The score may be within the interval of $[0, 3]$, where the score translates to:

Table 4.1: DREAD category ranking.

Score	Category rank
0	Low
1	Medium
2	High
3	Critical

The explanation of the different aspects of DREAD, in addition to an explanation of the scale used in the threat modelling, is presented below:

Damage Potential

“Ranks the damage that will be caused when a threat is materialized or vulnerability exploited” [25, p. 214].

0. No damage is done, and minor downtime may be present
1. Serious downtime
2. User data is compromised
3. Full system destruction

Reproducibility

“Ranks the ease of being able to recreate the threat and the frequency of the threat exploiting the underlying vulnerability successfully” [25, p. 215].

0. Impossible to reproduce the threat
1. Very hard to reproduce the threat
2. One or two steps necessary to reproduce the threat
3. Only the address bar, without any authentication, is necessary to reproduce the threat

Exploitability

“Ranks the effort that is necessary for the threat to be manifested and the preconditions, if any, that are needed to materialize the threat” [25, p.215].

0. Impossible to exploit the threat
1. Advanced programming or custom tools are necessary to exploit the threat
2. Malware exists, allowing exploitation of the threat
3. Only a web browser is necessary to exploit the threat

Affected Users

“Ranks the number of users or installed instances of the software that will be impacted if the threat materializes” [25, p. 215].

0. No customers are affected
1. One customer relation is affected
2. Several customers relations are affected
3. All customer relations are affected

Discoverability

“Ranks how easy it is for [...] researchers and attackers to discover the threat, if left unaddressed” [25, p. 215].

0. Impossible to discover
1. Inside knowledge or source code access is necessary
2. Guessing or network traffic monitoring is necessary
3. Information is visible in the web browser, in a form or similar, or the threat is common knowledge.

The average score of the different aspects is used to define the rank of the threat, using the conversion table seen as Table 4.2.

Table 4.2: Average-score-to-ranking conversion using DREAD modelling [25, p. 215].

Rank	Score
Low	0.0 - 1.0
Medium	1.1 - 2.0
High	2.1 - 3.0

Following the identification and assessment of threats using methods such as DREAD, a threat matrix provides a means to communicate the risk of these threats effectively. Threats discovered in STRIDE are deconstructed into pairs of consequence and probability, which are then plotted into a threat matrix, where the

coordinates convey the risk each threat poses to the system [26]. By utilizing a threat matrix, a security team can visually determine which threats to prioritize when implementing measures and intervention techniques for prevention and mitigation.

As the security team moves towards implementing these measures, understanding the risk appetite becomes essential. Risk appetite is defined as the type and amount of risk a product owner is willing to pursue or retain [27, p. 3]. It serves as the starting point for securing a system, acknowledging that it is challenging, if not impossible, to prevent all potential threats. Establishing a risk appetite enables setting a goal for how secure the system needs to be at the end of development. Often, finding a balance between lowering the risk appetite and maintaining the system's usability is necessary [3, p. 163-164].

With a clear understanding of the risk appetite, risk control comes into play. This practice involves identifying and cataloguing various prevention and mitigation methods to help reduce the severity of threats to a security system [27, p. 24]. Risk control can be achieved by preventing threats altogether, reducing the damage resulting from a successful attack, or, more commonly, a combination of the two. Examples of risk control measures include adding a VPN to a network to prevent unauthorized access or encrypting confidential data to mitigate the potential harm of data leaks to the system's confidentiality [3, p. 146-150].

Risk Prevention

Risk prevention is controlling the risks that are already identified [27, p. 23]. This often consists of functionality or culture-related measures [3, p. 155].

Risk Mitigation

Risk mitigation is reducing the effects of an already successfully completed threat to a security system. This will often consist of encrypting confidential data and implementing session/authorization invalidation functionality.

Bowtie Model

The Bowtie model is a way to visualize prevention and mitigation measures, in correlation to their related threats and their consequences [23]. By visually 'blocking' threat vectors, an individual can quickly surmise where a security system is lacking either prevention or mitigation measures.

4.3 Methodology

This section provides a description of the methodologies used to conduct the security review. The section will explain the rationale behind selecting these methodologies.

4.3.1 Structure

The structure of the security review will be according to the best practice of risk management performance and reporting, as seen in Figure 4.1 and outlined in ISO27005:2022, with a pipeline of identifying, assessing, and handling [27].

4.3.2 Assets Identification

Asset identification is the first step in performing a security review, according to ISO27005:2022 [27]. It has been used in the security review because it made it possible to identify the critical assets of the system, including hardware, software, and data. Identifying these assets made it possible to determine the potential risks associated with each asset and develop mitigation strategies accordingly. This approach helped focus the efforts on protecting the most critical assets of the system and ensuring the continuity of operations in case of any security breaches.

4.3.3 Threat Identification

Threat identification is the second step, where methods are utilized in order to explore and identify threats. The STRIDE model was selected to identify and evaluate potential security threats because of its structured approach. As mentioned in Section 4.2, STRIDE stands for Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege. By focusing on these six categories of threats, the STRIDE model allows for a comprehensive evaluation of potential security threats towards different aspects of a security system.

4.3.4 Risk Assessment

Risk assessment was used to assess and prioritize potential security risks systematically. DREAD was one of the two methods utilized to do so. This method includes scoring and prioritizing threats based on their potential damage, reproducibility, exploitability, affected users, and discoverability. It allows for a comprehensive evaluation of the security threats associated with the system and enabled the team to develop appropriate mitigation strategies to address these threats. DREAD was paired with a threat matrix, as a second method, which is based on the probability and consequence of each threat plotted out in a 2D matrix. This method represents a less comprehensive assessment, but generates an improved visualization, compared to DREAD.

4.3.5 Risk Control

Risk control is the last step in the security review. It was used to maintain, mitigate and prevent the threats found in the threat identification phase of the security review. Prevention and mitigation strategies were developed based on the risk assessment and threat modelling results. These strategies aim to prevent security

breaches and minimize the potential impact of any successful attacks. For this purpose, the Bowtie model has been used, enabling the group to visualize the different threats along with their potential consequences. Moreover, this approach allowed for the identification of barriers and controls that could be implemented to prevent or mitigate these threats [23]. Because of this, the group was able to develop a comprehensive set of prevention and mitigation strategies that provide maximum protection to the system. Risk control, including both mitigation and prevention, is always a balance between keeping the residual risk at a level lower than the risk appetite and maintaining a necessary degree of usability for the end users of the system [3, p. 163-164].

4.4 Assets Identification

4.4.1 Sensitive Data & Third Party Components

The system does not contain any sensitive data, nor does it manage customer data. The concept does not rely on any third-party components and should be possible to implement in several programming languages and environments.

4.4.2 Critical Assets

Critical assets are the organizational resources necessary for continued business operations [25, p. 64]. This includes, for example, all software components, hardware devices, and documents necessary to keep the intactness of availability, confidentiality, and integrity of the system. In the case of this conceptual system, the assets necessary to maintain the integrity of the system are of especially high importance, thus are the most critical assets. The threats assessed in this security review are centred around the critical assets of the system.

In this system, the critical assets are as follows:

Network Management System

The Network Management System (NMS) is the heart of the system, as this is where the customer can upload new licenses bought from the licensing company. The software is deployed directly onto customer-owned hardware within the local network, and there is very little control of the software's deployment environment. Here, the new licenses can be validated if both the signature is coming from the correct authority and if the signature matches the accompanying license payload. It is also asserted that the license has never been redeemed before by checking it against an encrypted file of checksums of all previously redeemed licenses. This file is required untouched for the system to boot. The NMS can also generate SLLs from their available pool of TLLs with any amount of time within the bounds of available time and send the SLL to any connected LFA.

License File Aggregator

The LFAs are where licenses are sent to be consumed. Any one LFA can hold a series of licenses with varying amounts of license durations associated with them. Throughout this project, it has been assumed that the LFAs are *safe*, as they are required to be proprietary hardware with safe-boot functionality. This means that the LFA is regarded as “truth-asserters” with unbreakable integrity. All licenses uploaded here have their integrity and authority checked as in the NMS and will only accept licenses from the NMS.

Root Certificate Authority

The root CA is the licensing company. Here, all TLLs are created per customer order. The TLLs are then signed with the customer-company relationship private key and sent by email to the customer, who then uploads them to their network.

Software Licenses

Software licenses, both TLLs and SLLs, are defined as a critical asset. These are defined as either the license file itself, in JSON format, and its signature, in sign format, or the ZIP file containing both of the aforementioned files. A TLL is always distributed from the licensing company to the customer using a file in ZIP format, containing the two mentioned, and necessary, files. To create a signature file, uniquely created in association with each license file, it is necessary to either have access to the root private key, contained with the licensing company, or the intermediate private key, contained within the NMS. If the license file is altered at any point in time, the associated signature file will become invalid, and one of the two private keys is necessary to create a new, valid, signature file.

4.4.3 CIA Classifying of Critical Assets

The CIA rank is the average rank of the results for the confidentiality, integrity, and availability of the asset as described in Table 4.3. The numbers express how large of a security risk the asset is within the system and ranges from 1 to 4, where 1 is low and 4 is critical.

Table 4.3: NTNU assets evaluation table [28].

Rank	Confidentiality	Integrity	Availability
1	Public	No requirement	No requirement
2	Internal	Expected	2 days
3	Confidential	Required	4 hours
4	Strictly confidential	Critical	Immediately

Table 4.4: Assets CIA classification.

Asset	Properties			Average Ranking
	C	I	A	
NMS	3	4	4	3.7
LFA	2	4	4	3.3
Root CA	4	4	1	3.0
Software Licenses	2	4	4	3.3

NMS

The NMS was given an average ranking of 3.7 which makes it a high-risk asset. It was ranked a 3 on the importance of its confidentiality, as it contains information that is confidential. It was ranked a 4 on integrity, as its integrity is critical to the security of the system. If any threat actor breaches its integrity, a lot of value is lost. Finally, it was ranked a 4 on availability, as it should respond to requests immediately.

LFA

The LFA was given an average ranking of 3.3 which also makes it a high-risk asset. Firstly, it was ranked as a 2 on confidentiality, because its information is only internal, as there are no mission-critical data on it. Secondly, it was ranked as a 4 on integrity because if breached, the customer would be able to use the product indefinitely. This, however, is highly unlikely as the LFA is expected to be proprietary hardware. Finally, it was ranked as a 4 on availability, as it should respond to license distribution and similar immediately.

Root CA

The root CA was given an average ranking of 3.0 which makes it a high-risk asset, but at the lower end. The root CA was ranked as a 4 on confidentiality as if any single root private key is leaked an entire customer relationship would be compromised. It was then ranked as a 4 on integrity as well because if a single root key were to be compromised or altered, an entire customer relationship would need their keys renewed if they want to buy a license again. Finally, a score of 1 on availability was given, as there is no requirement on how fast the root CA should respond to purchases with a new license.

Software Licenses

The software licenses were given an average ranking of 3.3 which makes it a high-risk asset as well. Firstly, it was given a confidentiality rank of 2 as the structure of a license is not secret, only internal. Secondly, it was given an integrity ranking of 4 as if compromised a threat actor could in theory create their own faux licenses.

Finally, it was given an availability rank of 4 as it has to be immediately available to use once it has been received into a system. This last requirement is not difficult to achieve as it is a piece of data and not a system made to respond to anything.

4.5 Actors & Roles

4.5.1 Root License Administrator

The root license administrator is an employee of the licensing company whose roles within the system is to process customer purchases, create new TLLs, sign them with the corresponding customer relation private key, and send it via email to the customer. Any individual with this role could if compromised leak the secret private keys and effectively destroy the established chain of trust. For this reason, only people with a very high security clearance should have access to this role.

4.5.2 Client Employee

The client employees' roles include, but are not limited to: uploading top-level licenses to the NMS, generating sublevel licenses for the LFA, and consuming the licenses by using the software functionality. The client employee does also have to be regarded as a threat actor. The reason is that if the actor can breach the security of their locally deployed NMS, they can effectively re-use licenses forever. Often, the integrity of a security system benefits the customer and is present for the customer. Yet, in this situation, the customer is the one who could gain from breaching the security of their own system.

4.5.3 Client Network admin

The client network admin manages the NMS and the locally deployed network. They have direct physical and root access to the NMS. The client network admin does also have to be regarded as a threat actor, similarly to the client employee.

4.5.4 External Actors

As a result of the fact that the system is placed in a closed offline network, not connected to the internet, a potential external actor will need physical access or gain access through someone who has access to the closed network. This may, for example, happen through a breach of the physical security of the licensing company compound or through social engineering affecting someone who already has access to the closed network. In effect, the external actors are anyone outside the licensing company and the customer company who have the competency and motivation to gain access.

4.5.5 Malicious Actors

All actors listed in Table 4.5 have the potential to become malicious for different reasons. For instance, a client employee or network admin could be considered malicious should their intention be to compromise the system. Additionally, an external actor also holds the potential to become malicious for the same reason. Furthermore, the licensing administrator with root access holds significant power and can inflict severe damage on the system should there be an incentive such as financial gain.

Table 4.5: Security Review Actors & Roles.

Actor	Roles
Root license administrator	<ul style="list-style-type: none"> • Process customer purchases • Create and sign new TLLs • Sending TLLs via electronic mail to customer
Client network admin	<ul style="list-style-type: none"> • Maintains NMS & LFAs on client network
Client employee	<ul style="list-style-type: none"> • Uploads TLLs to NMS • Generates SLLs to LFAs • Consumes licenses by using software functionality

4.6 Risk Appetite

In the case of OLM as a concept, the group finds that the risk appetite should be low. The nature of the concept means that a threat actor, on behalf of – or as – a customer, can achieve significant monetary gain and net loss for the licensing company. This can be done in order to achieve net savings for the customer or private monetary gain.

To be able to deem the concept as viable and secure, the following DREAD modelling and Threat Matrix requirements must be fulfilled, where no threat can be:

- DREAD modelled to a higher rank than low,
- placed in the risk matrix at a higher risk degree than *green*.

4.7 Threat Identification: Stride

STRIDE is a threat modelling technique used to identify and classify various threats to a system and what aspect of the system the threat would violate [25, p. 65]. In this section, the different STRIDE threats are listed as subsections and their accompanying discovered threats with description is written below.

4.7.1 Spoofing

Spoofing is the activity of impersonating fake human or technical identities, in order to gain privileges that the user normally would not have available.

S1 Signing of Fake Licenses

If either the root private key or the intermediate private key is leaked or obtained in any other way, as described in Section 4.7.4, it is possible to sign fake licenses. This will make it possible for a threat actor to impersonate either the licensing company or the NMS, creating faux licenses that will still redeem as if authorized by the licensing company.

S2 Malicious license files

If a threat actor purporting to be the licensing company sends the customer a malicious license file, damage can be done to the customer's network and infrastructure. When the license file upload happens, for example as a compressed package, it is possible to upload malicious scripts or similar, attacking the host machine before security measures can react. This could as an example manifest as a zip-bomb [29].

4.7.2 Tampering

Tampering is the action of performing unauthorized changes that affect the integrity.

T1 Ledger Files Replacement

When a ledger file containing the checksums of the already redeemed TLLs is being used, it can be replaced at strategic moments in time for license duplication. This may be done by copying the file at an early stage before any license have been redeemed. By later replacing a ledger file full of consumed redeemed licenses with the earlier copy, the customer can effectively reset the list and redeem the licenses again. This attack also applies to the save-file containing the serialized pool. By copying the file while the pools contain active licenses, the customer can swap back to the populated license pool after spending the initial pools, effectively gaining unlimited licenses.

T2 Encrypted Ledger Files Modification

If the NMS secret key is leaked or obtained, it is possible to modify either the file containing the serialized pool or the file containing a list of the checksums of the redeemed licenses. By doing this, the serialized pools may be added to in size or the list of checksums may be emptied in order to make it possible to redeem TLLs several times.

T3 Source Code Modifications

If a threat actor is able to reverse engineer the code, modify the source and compile a new version of the software, it will be possible to bypass the integrity checks, license file parsing, or other features that guarantee the software operates correctly.

4.7.3 Repudiation

Repudiation is the inability to prove claim or guilt, where the actions performed may be denied, as a consequence of lack of tracing, logging, authentication, or similar.

R1 Repudiable License Distribution

Given the lack of authentication and authorization within the scope of the concept, there is no non-repudiability within the concept regarding the distribution of licenses from the NMS to the LFAs. Anyone with digital access, for example the IP address to the NMS or LFAs, is regarded as an admin.

4.7.4 Information Disclosure

Information disclosure is the action of unveiling, disclosing, or revealing confidential information to unauthorized users or processes.

I1 Intermediate Private Key Leak

Should the intermediate private key for any deployed NMS be obtained by a threat actor, the chain of trust will be compromised. The threat actor will be able to sign their own SLLs. This key can either be gained through social engineering or through discovering it in the local installation of the NMS.

I2 Root Private Key Leak

Should the root private key for a customer relationship be disclosed, either on accident or on purpose, the chain of trust will be compromised. The threat actor will be able to sign their own TLLs. This leak can, for example, be achieved through techniques such as social engineering or a digital breach.

4.7.5 Denial of Service

Denial of service is the action of restricting an authorized user or process the access to one or more resources that they should have access to.

D1 Power Outage

If the power is compromised in any customer systems, the customer will effectively be locked out and any use of licenses will be impossible. This can be achieved by, for example, physically breaking the building's power inlet.

D2 Network Equipment Failure

The licenses are distributed from the NMS to the LFA through the local network. If the network equipment, in any way, breaks or fails, either intentionally or unintentionally, it will no longer be possible to distribute SLLs. It will only be possible to redeem TLLs at the NMS.

D3 NMS Host Unit Downtime

If the host unit device where the NMS is running has downtime, it will not be possible to redeem TLLs or distribute SLLs from the NMS to the LFAs. This can happen through both intended and unintended downtime, for example through unintended downtime caused by broken hardware or intended downtime through sabotage performed by a threat actor.

4.7.6 Escalation of Privilege

Escalation of privilege is where an authorized or unauthorized user or process gains access to resources that the process or user normally does not have access to. This is performed by an unauthorized increase in the user's or process' rights. As there is no authentication system within the concept, Escalation of privilege is not topical.

4.8 Risk Assessment: DREAD

In this section, DREAD modelling has been performed. The threats are given a score for the five risk categories, assessing their severity. In addition to this, the reasoning behind the assessments is presented. The letters of DREAD are explained in Section 4.2, where dread is short for *Damage potential*, *Reproducibility*, *Exploitability*, *Affected users*, and *Discoverability*.

The average score to ranking conversion can be seen in Table 4.2.

¹ Depends on the source code language of the actual concept implementation.

Table 4.6: Summarization of threats using DREAD modelling.

Threat	D	R	E	A	Di	Avg.	Rank
S1 Signing of Fake Licenses	2	2	3	1	1	1.8	Med
S2 Malicious license files	1	1	2	1	1	1.2	Med
T1 Ledger Files Replacement	2	2	3	1	1	1.8	Med
T2 Encrypted Ledger Files Modification	2	2	1	1	0 ¹	1.2	Med
T3 Source Code Modifications	3	1 ¹	2 ¹	2	1	1.8	Med
R1 Repudiable License Distribution	2	3	3	1	2	2.2	High
I1 Intermediate Private Key Leak	2	2	2 ¹	1	2	1.8	Med
I2 Root Private Key Leak - Singular	2	1	1	1	2	1.4	Med
I2 Root Private Key Leak - Multiple	3	1	1	2	2	1.8	Med
D1 Power Outage	0	1	1	1	2	1.0	Low
D2 Network Equipment Failure	0	1	2	1	2	1.2	Med
D3 NMS Host Unit Downtime	0	2	2	1	2	1.4	Med

S1 Signing of Fake Licenses (Medium)

The threat is considered to be medium as the damage potential and affected users scores are low, but the exploitability score is higher. The threat is not easily discoverable, but once discovered, holds the possibility of being easily reproducible.

Getting a hold of one of the private keys allows the threat actor to sign their own licenses on the behalf of the company, which will damage the company's earnings. However, as the keypairs are different between company-customer relations, only that specific relation will be affected.

S2 Malicious license files (Medium)

This threat has been given medium damage as such an attack could, for example, result in the system shutting down or the local files being encrypted. Because the network is entirely offline, no data can be stolen, but it could, for example, be held for ransom. Reproducibility has been ranked as medium as well, as this attack would likely only work once in a while, as it likely requires insider access to the licensing company or social engineering. Both of which often results in the threat actor being discovered and losing their authentication, or a revamp of the security training of on-site employees to prevent future social engineering attacks. Exploitability has been ranked as high, as there exists malware online that will do the work for a threat actor, leaving only the task of getting it into the system to them. Affected users and discoverability was ranked as medium as it will only affect one customer relationship and the threat actor would require insider knowledge of how the licensing company sends licenses to a customer.

Overall, the threat was ranked as medium.

T1 Ledger Files Replacement (Medium)

This threat is defined with a critical exploitability ranking and high damage and reproducibility ranking, while the affected users and the discoverability rating are medium. The damage rating is set because the threat will completely avoid the restrictions prohibiting the redeeming of TLL files several times. The reproducibility rating is set because of the ease of reproducing; if one customer has exploited the vulnerability associated with the threat, it is easy for other customers to do the exact same thing, as there is no difference in the threat between different customers. There are no tools necessary to copy and replace files; the only knowledge necessary is the knowledge associated with the fact that the files are replaceable and which files that need to be replaced. This gives the threat a critical exploitability ranking and a high discoverability ranking. The affected users are only the current customer relation, giving the threat a medium ranking in regard to the affected users.

All in all, this threat is given a medium ranking.

T2 Encrypted Ledger Files Modification (Medium)

This threat was given a high damage rank as if successful, the customer could reuse old licenses indefinitely, effectively releasing the customer from their need of ever buying a license again from the licensing company. It was also ranked high for reproducibility, as once it has been done it is rather simple to do again. It has a medium exploitability ranking, as it will require advanced IT knowledge; it is necessary to create a program that manipulates the ledger files properly in order to compromise the files. Furthermore, the threat has a medium rated affected users score as it will only affect the customers who perform the attack themselves, given the nature of the secure environment. Finally, its discoverability was rated as low, as the threat actor would require intimate knowledge of the source code.

This results in an overall medium rank for the threat.

T3 Source Code Modifications (Medium)

This threat has been given a high damage rank because the ability to remove or avoid integrity checks, both for the local files and the licenses that will be parsed, will make it obsolete to spend any more money on licenses from the licensing company. Although it depends on the source code language chosen, most relevant languages make it difficult to reverse engineer, making the threat difficult to reproduce. If the threat actor is able to reproduce the threat, it is relatively easy to exploit and materialize the threat; there are tools and applications available to alter and compile source code. If the threat is exploited, it will only affect the user who altered the source code. Although, if the altered version is distributed to other customers, it will be possible for several users to become affected by the threat.

The threat is hard to discover, where it is necessary to have inside knowledge about the source code or the source code itself to discover this threat.

All in all, the threat is ranked as a medium.

R1 Repudiable License Distribution (High)

As the concept itself does not contain any specification for user-authentication, the threat of unauthorized users is the highest ranked threat. The damage has been ranked high, as a lot of licenses can be redistributed around the system or deleted if this were to happen. Reproducibility and exploitability has been ranked critical as all functionality can be accessed from an open API endpoint within the system, making this insecure. The affected users ranking has been set to medium, as the damage would be contained to a single customer network. Finally, discoverability has been ranked high, as a threat actor could gain knowledge of this simply by looking at network traffic or guessing.

This results in the threat receiving the rank 'high'.

I1 Intermediate Private Key Leak (Medium)

This threat has been given a high damage rank as it could severely compromise customer data by allowing customers to self sign their own custom licenses. It has been given a high reproducibility rank as well, as once it has been leaked, it would be rather easy to perform the threat multiple times. Exploitability is set to high, but could be critical, depending on the language used to implement the concept. The affected users ranking is set to medium, as it would only ever affect a single customer within their own network. Finally, the discoverability ranking is set to high, as chain of trust is a well known system within the industry, which would be easily recognizable to a threat actor.

This results in a medium rank for the threat.

I2 Root Private Key Leak – Singular (Medium)

This threat has been given a high damage rank, as if the private key has been obtained for a single customer relationship, it would allow the customer to self sign their own custom licenses. Following this, the reproducibility, exploitability and affected users are all limited to medium threats as the certificate authority should be a highly secure institution within the system. Finally, discoverability has been set to high, as the concept of a root certificate authority is a well known way of establishing chain of trust for secure systems within the industry. It would not take a threat actor long to realize that it exists.

Overall, the threat has been ranked as medium.

I2 Root Private Key Leak – Multiple (Medium)

This threat has been given a critical damage rank as if the private key has been obtained for several customer relationships it would allow all the customers to self sign their own custom licenses. Following this, reproducibility & exploitability are limited to medium threats, as the certificate authority should be a highly secure institution within the system. The affected users ranking has been set to high, as this would affect several users within the system as opposed to just one. Finally, discoverability has been set to high as the concept of a root certificate authority is a well known way of establishing chain of trust for secure systems within the industry. It would not take a threat actor long to realize it exists.

Overall, the threat has been ranked as medium.

D1 Power Outage (Low)

This threat has low damage, with medium reproducibility, exploitability and affected users, while the discoverability is high. The damage is chosen due to the fact that materialization of the threat do not damage the system other than affecting the availability. In regard to reproducibility, exploitability and affected users, the score is chosen due to the fact that it is hard to perform a power outage, although not impossible, and there are no affected users other than the one affected customer relation. As a consequence of the fact that it is common sense that a power outage affects the availability of a digital system, the discoverability is also set to high.

All in all, this threat is given a low ranking.

D2 Network Equipment Failure (Medium)

The threat has been given a low damage rating and a medium reproducibility and affected users rating, while the exploitability and discoverability has been set to high. The reason the damage has been set to low is the lack of real damage occurring when, or if, the threat is materialized. The availability will be affected, but no data will be compromised, and no systems will take damage. When the network equipment is fixed or swapped out for functioning equipment, everything will be back to normal. The reproducibility rank has been set due to how difficult it is for an external actor to make the network equipment malfunction, especially given the fact that the network is local and offline. The same goes for the exploitability rank. The discoverability has been set to the same rating, as it is not hard to figure out that the system is reliant on a network connection. As for the affected users, a network equipment failure will only affect the current customer relation.

The combination of the risk category assessments gives this threat a medium ranking.

D3 NMS Host Unit Downtime (Medium)

The threat is considered to be medium, as the reproducibility, exploitability and discoverability are high, while availability and damage potential are low. Should the system experience downtime, then its core functionality would become unavailable, thus impacting the customer. However, should the threat materialize, expected downtime is set to be minimum as it is possible to use the system application on multiple devices.

4.9 Risk assessment: Threat Matrix

In this section, the threats discovered in Section 4.7 are modelled into a threat matrix with a value for probability and consequence of threats. The threats from Table 4.7 have been plotted into the threat matrix in Table 4.8, allowing for swift visualization of factors such as priority and risk.

The threats have been given a consequence and probability grading according to the DREAD modelling done in Section 4.8.

Table 4.7: Value tuples for threat matrix visualization.

Threat	Consequence	Probability
S1 Signing of Fake Licenses	4	3
S2 Malicious license files	3	3
T1 Ledger Files Replacement	4	3
T2 Encrypted Ledger Files Modification	4	2
T3 Source Code Modifications	4	2
R1 Repudiable License Distribution	1	4
I1 Intermediate Private Key Leak	4	2
I2 Root Private Key Leak - Singular	4	1
I2 Root Private Key Leak - Multiple	5	1
D1 Power Outage	1	3
D2 Network Equipment Failure	1	3
D3 NMS Host Unit Downtime	1	3

Table 4.8: Threat matrix.

		Probability				
		1	2	3	4	5
Consequence	1			D1,D2, D3	R1	
	2					
	3			S2		
	4	I3-1	I2,T2, T3	S1,T1		
	5	I3-2				

4.10 Risk Control: Bowtie Modelling

4.10.1 Spoofing

To be able to prevent spoofing, it is important to have ways of making sure components and requests are legitimate, for example by finding a way to uniquely identify the correct and incorrect components and requests.

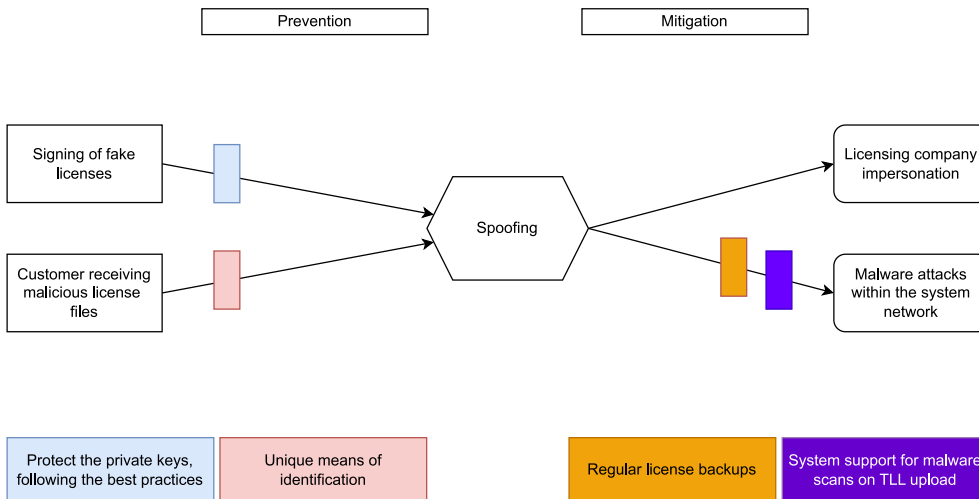


Figure 4.2: Bowtie model for spoofing threats.

One way of performing this identification is through the use of keys, where in this case, the root private key and the intermediate private key will need to be kept safe from threat actors. When these keys are kept secret, it is not possible to sign fake licenses.

A way of identifying requests and components is necessary also when sending and receiving TLLs meant to be uploaded into the system, where it is important to be able to make sure the TLL is legitimate and does not contain malicious files. One way of doing this can be to implement a system of identification, to make sure the files are from the correct sender.

To mitigate spoofing attacks, it is important to minimize the possible consequences of a spoofing attack. One way of doing this is through regular backups of data, where corrupt or lost data can be replaced by the backed-up data. In addition to this, if the customer has received malicious license files, support for malware scans during upload would mitigate this attack, albeit this would require some sort of online connection. Otherwise, the customer could perform a manual scan before uploading the payload. These mitigations do not have any effect if the malicious files contain undetected malware and the malware is aimed towards other aspects than data integrity and availability.

4.10.2 Tampering

As illustrated in Figure 4.3, potential threats may compromise system integrity. To prevent these risks, various precautionary measures should be implemented.

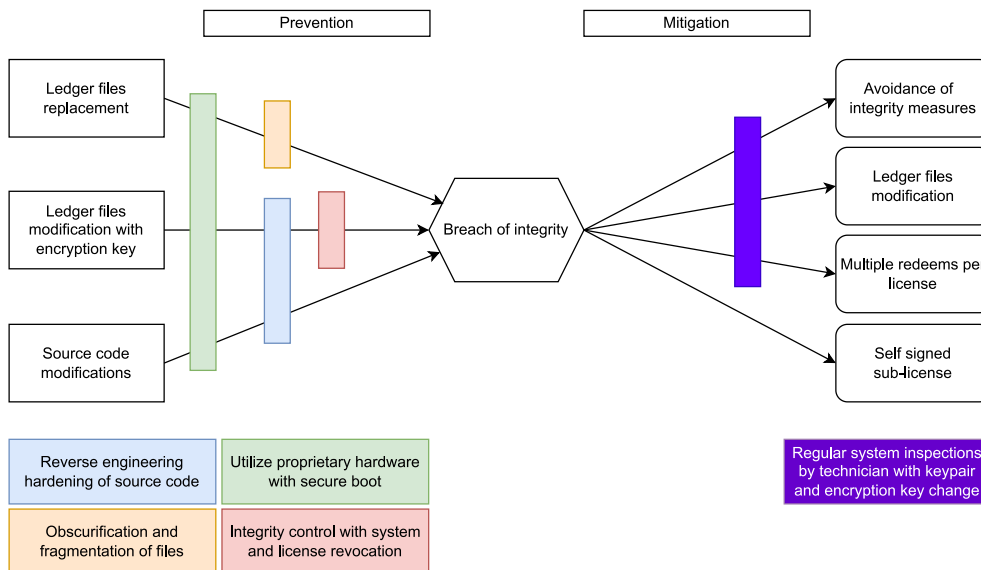


Figure 4.3: Bowtie model for integrity threats.

One effective approach to secure a system is to increase the difficulty of reverse engineering. This can be achieved by employing binary hardening techniques, which involves the strengthening of an application’s binary code in order to protect it against potential exploitations.

Another approach to consider is the incorporation of code obfuscation techniques. Code obfuscation is the deliberate modification of source code to make it harder

for humans and machines to understand, hindering the interpretation of the system's functionality.

Furthermore, the use of proprietary hardware with secure boot capabilities introduces an additional layer of security to the system. Secure boot is a technology that ensures only authorized and trusted firmware and software can be executed during the system's startup process. By incorporating proprietary hardware, it is possible to integrate custom security features and protocols that are specifically designed for the system in question. This makes it increasingly challenging for threat actors to access and manipulate system components, as they would need to overcome multiple security barriers tailored to the unique aspects of the hardware in use.

In the event that an integrity breach occurs, it is crucial to implement mitigation measures. The most effective approach involves dispatching a technician from the licensing company to perform a keypair change, which guarantees that only legitimate files are accepted and generated. This will restore the integrity of the system temporarily. The problem with this solution is that the customer can simply breach the system again, as the system is not more secure than before, simply *restored*. The solution will also drastically affect the usability of such a system, as the licensing company will experience a smaller financial setback due to increased number of technicians required and the customers will have to endure all the visits and downtime that follows.

4.10.3 Repudiation

As seen in Figure 4.4, there is only one threat, which is R1 Repudiable License Distribution. This is also coincidentally the only threat classified as high risk. Given the nature of the concept, there is no authentication within the system. This, however, does not prevent the licensing company to complement the system with authentication, which would prevent this threat from coming to fruition. By doing this, the threat would essentially be rendered inert, unless a threat actor were to bypass the added authentication.

Should the attack have already been successful, there are a couple of ways to mitigate the two defined outcomes. Firstly, there is adding comprehensive logging of all actions taken within the system. This would include time, IP-address, and – if available – user-ID from authentication added in prevention. Secondly, adding functionality to withdraw SLLs from an LFA, and add it to the NMS as a TLL. This recovery functionality would mitigate the damage done by an actor sending out an SLL to the wrong LFA.

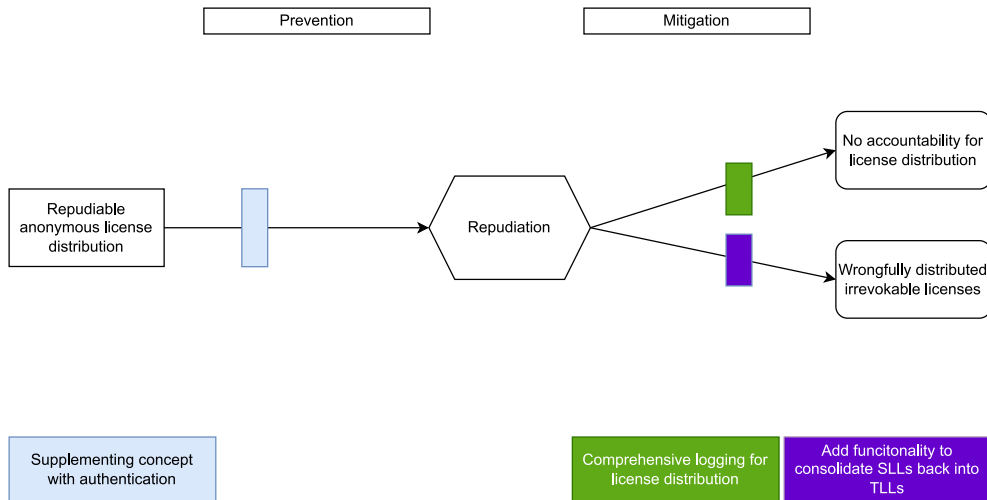


Figure 4.4: Bowtie model for Repudiation threats.

4.10.4 Information Disclosure

As seen in Figure 4.5, there are three threats attempting to achieve information disclosure. The first one is obtaining the root private key for one or more customer relationships, the second is obtaining the intermediate private key for one customer relationship, and the last one is obtaining information about how the licensing company protects the integrity of the data in their NMS. These three have three preventions that can be applied, the first of which is regularly re-educating authenticated personnel about social engineering and new security threats. The second is utilizing code obfuscation to prevent the source code from leaking the intermediate private key or the utilized integrity techniques. The third, is to store all the private keys throughout the system in secure offline key stores with strong passwords. Utilizing these three prevention techniques, all threat vectors have two barriers to get through before achieving information disclosure.

Mitigating an already successful Information disclosure attack is a more difficult situation as the entire network is offline. Keys and certificates cannot be made invalid after-the-fact as in normal chain of trust systems. The solution is then to regularly manually change customer key-pairs with a licensing company technician visiting the customer. This however does not really solve the problem when the customer is considered to be the threat actor. Nothing prevents the threat actor from finding the new intermediate private key and creating their own licenses again. Given the weak mitigation possibilities on an offline chain of trust system, a lot of the security is dependent on prevention rather than mitigation.

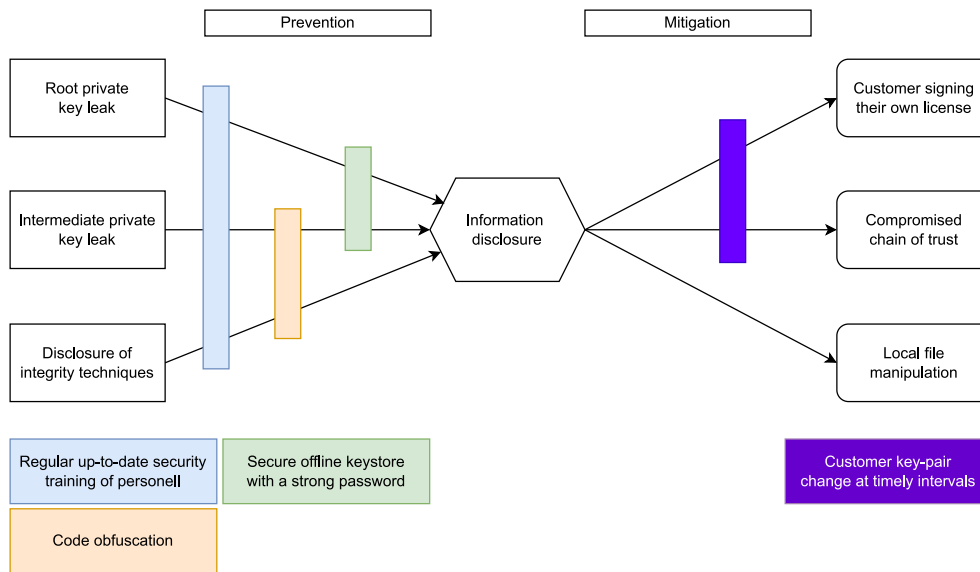


Figure 4.5: Bowtie model for Information disclosure threats.

4.10.5 Denial of Service

As illustrated in Figure 4.6, there are three threats which could result in Denial of service for the system: power outage, NMS host unit downtime either by failing hardware or sabotage, and network equipment failure. These three can all be prevented in a couple of ways. A power outage and the NMS host unit going down can both be prevented by ensuring strong on-site security routines with trained personnel around exposed assets. Second, the NMS host unit going down, and network equipment failure can be prevented in two ways. Firstly, by only utilizing quality hardware with built-in fail safes which can prevent natural equipment failure to a certain extent, and secondly by regularly maintaining all equipment within the system.

Should the threat have already succeeded, there are a few ways to mitigate the denial of service. Firstly, an on-site backup power supply can keep the system up for some time until it runs out of energy. Secondly, in the case of the NMS unit failing, a backup NMS readily available in the system could replace the one that has gone offline. Finally, the effects of the NMS going down can be mitigated by having scheduled the license distribution a while before the SLLs are needed in an LFA.

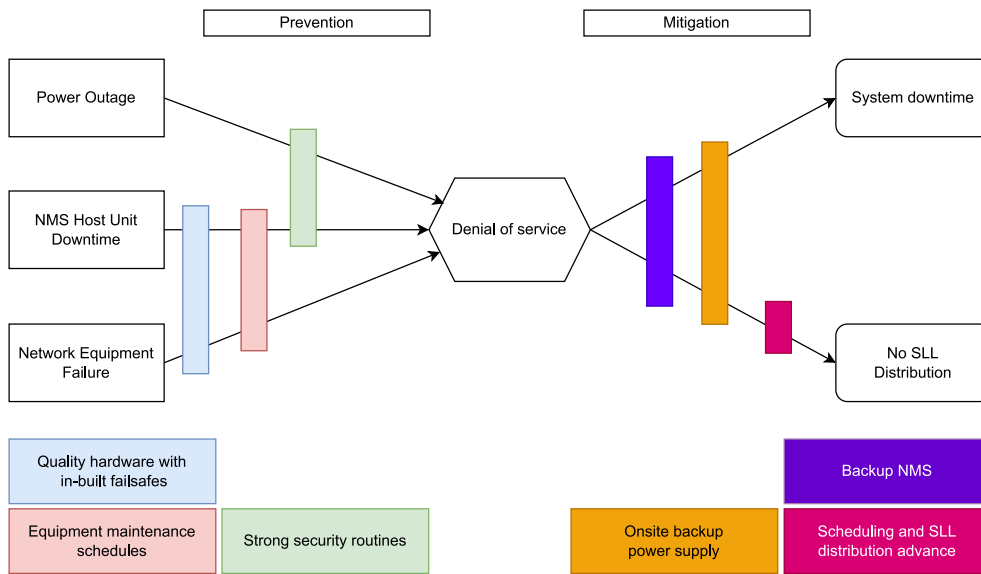


Figure 4.6: Bowtie model for Denial of Service threats.

4.10.6 Escalation of Privilege

As there are no threats under the STRIDE category of escalation of privilege, there is not presented any mitigation or prevention measures.

4.11 Residual Risk

The residual risk after discussing the several ways to prevent and mitigate security threats is set to medium. Given the nature of OLM, once the integrity is breached or information disclosed, there is little that can be done to prevent severe exploitation of the security system.

First of all, the entirety of the mitigation section of Bowtie 4.3 consists of customer key-pair changes. This is not a final solution as it will only temporarily mitigate the outcome of the original threats' success, as the door for it to happen again remains open. The solution also introduces a higher operational cost and reduction of usability, with a licensing company technician visiting the customer's network physically at a set interval. Second of all, the entirety of the mitigation section of Bowtie 4.5 consists of the same solution introducing the same operational costs, and cost-of-usability. Additionally, both these mitigation sections are missing coverage on a threat vector each, which means that even the small mitigation of swapping key pairs would not completely cover the outcome.

The upside of OLM is that if any exploit is deployed within a customer's network, the damage is retained within the network. This is the primary reason the risk assessment never reached much further than medium, mostly, as the number of affected customers was low.

4.12 Discussion

In the course of the security review, a number of threats have been identified and assessed. In addition to this, most of the threats have been prevented and mitigated in order to lower the risk of the system.

4.12.1 How Does the Risk Stand?

Most of the risks presented have been removed completely after the prevention and mitigation measures have been implemented. There are some threats, thus also a risk, that is unavoidable, even after implementing measures, at least not without affecting the usability to a degree that is experienced as too high.

Overall, the group believes the concept naturally comes with a high risk. There exists a multitude of good threat preventions that could be deployed, as presented in Section 4.10, but once the threat has been materialized, there is little a licensing company could do to mitigate it.

4.12.2 Is it Viable?

As the residual risk (medium) is higher than the risk appetite of the concept (low), the concept cannot be deemed viable in the sense of security. It is possible to implement or suggest further measures or additional security functionality in order to reduce the residual risk, although these measures are deemed to have a too high effect on the usability, interfering with the demands for the usability of the concept.

With the currently suggested implementable measures and security features, it is possible to utilize the system. The downside is the residual risk, which is higher than the desired residual risk, defined through the risk appetite. If the trust in the customers is improved, thus increasing the risk appetite, the system may be looked upon as viably secure. In a practical deployment, for example, the licensing company will be able to detect when a customer stops buying licenses, and the licensing company may end their customer support. The compromised customer system can run for an undetermined amount of time without support, thus making this mitigation only affective after the need for customer support arises.

The accepted risk appetite may be regarded as strict, yet it is necessary in order to protect the licensing company's assets due to the mentioned vulnerable nature of the concept. This makes sense in *business-to-consumer* relations, where the customer of the licensing company is a private consumer. In many cases, although,

the concept is made available and is handling licenses in a *business-to-business* relation. In these cases, the customer may be more trustworthy, as the customer wants a product that delivers the required functionality and wants to receive support on this functionality. If the customer were to crack the system, and stop buying licenses, the licensing company will notice, and support may not be offered. In addition to this, the customer may see less value in cracking the system than what a private consumer customer does.

4.13 Conclusion

Throughout the security review, the group utilized several threat modelling exercises such as STRIDE, DREAD, and the Bowtie model. During these exercises, there was uncovered a series of threats that could compromise the relationship of any company attempting to deploy this concept in their product line and their customers. Given the nature of a customer-hosted trusted security asset existing entirely offline, the licensing company can never revoke fraudulent licenses, update private keys, or prevent the customer from tampering with the system directly. All of which can result in a severed relationship between the customer and licensing company. With a high enough risk appetite OLM could be deployed in a real-world scenario, but with the current requirements it can not be defined as secure.

Chapter 5

Proof of Concept

5.1 Introduction

This chapter delves into the practical realization of the concept discussed throughout the thesis. The objective is to demonstrate an example of how the concept of OLM can be transformed into a functional software application, highlighting its key features and capabilities. A thorough examination of the software will be conducted, focusing on its architecture, essential components, and their interactions. This approach aims to provide an understanding of how the concept was brought to life, showcasing its potential and areas for improvement.

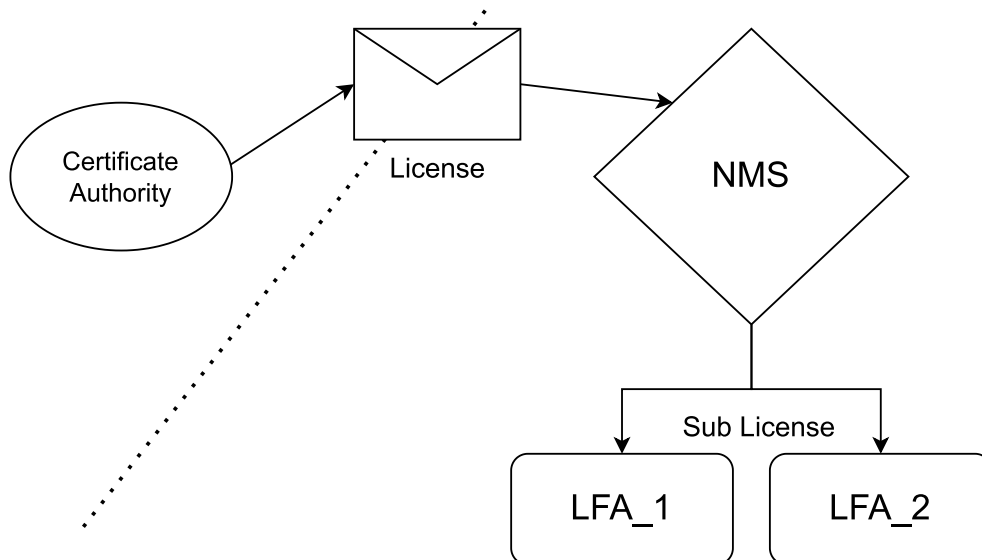


Figure 5.1: Initial PoC Architecture.

In regard to the PoC, under the influence of Nevion, the license functionality has been named “*media function*”. There is no difference between the license functionality discussed and presented in Concept Review and the media functions used in the PoC, except the fact that the media function implementation has been used as a part of the example implementation of the concept.

From the concept, as described in Chapter 3 – Concept Review, the initial idea of architecture is defined in Figure 5.1. This figure is added as a starting point for the architecture development of the PoC.

5.2 Requirements

5.2.1 Functional Requirements

- Redeem and accumulate functionality from TLLs on Network Management System (NMS).
- Generate SLLs.
- Send SLLs to LFAs.
- Consume licenses on LFAs.

5.2.2 Non-functional Requirements

- NMS coded in Java.
- LFA coded in C++.
- Secure implementation, with regard to integrity.
- Secure implementation, with regard to serialization.

5.2.3 Operational Requirements

- Reliability, whereas results should be consistent and accurate.
- User-friendly, with clear documentation and straightforward interfaces.
- Logging for diagnostic purposes.

5.3 Technologies

Implementing the concept of Offline License Management (OLM) into a product that can be used in practice requires a number of different technologies combined. These are the technologies that have been utilized in order to create this Proof of Concept.

5.3.1 OpenSSL

OpenSSL is a widely used, open-source toolkit implementing the Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols, as well as a general-purpose cryptography library. It has been used in this project to ensure secure

communication between different components and provide cryptographic functionality for certificates and authentication [30].

5.3.2 Network Management System

According to the non-functional requirements, the Network Management System (NMS) has been coded in Java. The Java version is set to 11, as the development team has the most experience with this version. Although Java 17 offers some newer features and general improvements, the benefits of using a familiar version in this project outweighs the potential gains of adopting the latest release. Java 11 also provides long-term support, which ensures an up-to-date JDK for the application even though it is the older version of the two [31].

Java Library Manager

MAVEN has been used as the Java library manager to streamline the build process, manage dependencies, and automate the packaging and deployment of the application [32].

API

The Java framework Spring Boot has been chosen for the API layer of the Network Management System (NMS), as it allows for rapid development, easy configuration, and seamless integration with other tools and libraries [33].

5.3.3 License File Aggregator

According to the non-functional requirements, the License File Aggregator (LFA) has been coded in C++. C++ was paired with the networking library *Oatpp* which includes all the different technologies required. It is a verbose library, requiring a lot of boilerplate for simple things, such as a client, but such is the nature of the 'beast'. Things which may be regarded as trivial in other higher-level languages require several lines of setup and configuration within *Oatpp*.

Additionally, the OpenSSL library has been used to provide custom cryptographic functionality for the system's various certificates and for HTTPS support. Finally, CMake is utilized to bundle the codebase into an executable properly with the g++ compiler.

5.3.4 Frontend

The frontend implementation was not originally a part of the thesis and did therefore not have any requirements applied to it, which allowed the group to build it in whatever they wanted to. The frontend implementation was constructed with the web framework Vue [34] and one of its plugins 'vuetify' [35] which allows for

good-looking standardized components for quick development. Axios was utilized for all API requests to the NMS because of its quick setup with little effort [36].

In addition, the code also utilizes the Node package manager (NPM) [37] and Node.js [38] for its long list of dependencies and its runtime environment. NPM allows for quickly fetching project dependencies, which allows for quicker development by utilizing open source packages that solve different problems. This is done in an attempt to never re-invent the wheel, but also comes with the plethora of issues that are associated with third party dependencies [39].

5.4 Design

5.4.1 Structure

The Proof of Concept is structured into 3 different code bases: one code base representing the NMS; one code base representing the LFA; and one code base representing the frontend. In addition, there is a script created for signing TLLs. The repositories are split for ease of deployment, difference of environments and separation of documentation.

The split is also attributed to the relationship between the different elements within a deployed OLM system. The NMS is, as its name indicates, a manager of the network and allows for managing several LFAs within a network. This one-to-many hierarchy is important to separate in code as well, as deploying an NMS per LFA is unnecessary.

5.4.2 Communication

Chapter 3 – Concept Review presents two channels where communication is necessary for an implemented instance of the concept. Communication is necessary between both the licensing company and the customer hosting the concept system, and between the NMS and the LFAs within the network.

As for this PoC, the concept requirements for communication have been chosen to be covered by two different RESTful APIs, one at the NMS and one at the LFA. These communicate by the use of requests built up of pre-defined structures and contents [40]. The choice of a RESTful API was made for its widely used architectural style for web services. It is also scalable, allowing high cohesion and low coupling between services [41]. The communication between the NMS and the LFA is done solely from the NMS to the LFA, as the NMS is the orchestrating node. An exception to this rule of direction is the initial contact between one LFA and the NMS, upon connection, which will be described in detail under Section 5.6.2. The rule of direction is deduced through a *one-to-many* relationship, where one NMS is potentially connected to several LFAs.

As for the communication between the licensing company and the customer hosting the instance of the system, no means of communication has been chosen in this example implementation of the concept. This is done because the means of communication is irrelevant to the implementation, as long as the Top-Level License can be redeemed at the NMS without an internet connection. The TLL can as an example be mailed to an online customer-device and then physically moved into the offline network for uploading into the NMS.

5.5 License Signing

License signing at the top level is performed using a simple bash script – called the License File Signer (LFS) – that accepts the license to be signed and the root authority’s private key as inputs. The script can be seen as Listing 5.1. The output is a file with the same name as the license file, appended with a '.signature' suffix. The purpose of the signature is to ensure that the license was signed by the correct authority, providing a secure way to validate the authenticity of the license. These two files are then shipped together to the customer within a zip file to be redeemed. Sub-Level Licenses are signed by the NMS itself with its intermediate private key automatically whenever a license is created by the NMS. More on this in Section 5.6.8 – Sub Level License Generation.

Code listing 5.1: Entire LFS bash script utilizing OpenSSL.

```

1  #!/bin/bash
2
3  # Check if the required parameters have been provided
4  if [ $# -lt 2 ]; then
5      echo "Usage: $0 <file_to_sign> <private_key_file>"
6      exit 1
7  fi
8
9  # Input file to be signed
10 file=$1
11
12 # Private key file used to sign the input file
13 key=$2
14
15 # Sign the file
16 if openssl dgst -sha256 -sign "$key" -out "${file}.signature" "$file"; then
17     echo "File signed successfully."
18 else
19     echo "File signing failed."
20 fi

```

5.6 Network Management System

The Network Management System (NMS) from the concept is implemented in the Proof of Concept as a Java application, the structure of which can be seen in Listing 5.2.

Code listing 5.2: NMS Source Code Tree.

```

.
|-- java
    |-- no
        |-- ntnu
            |-- nms
                |-- App.java
                |-- CustomerConstants.java
                |-- Init.java
                |-- api
                |   |-- BodyParser.java
                |   |-- Constants.java
                |   |-- client
                |   |   |-- Client.java
                |   |-- handlers
                |       |-- LfaRegistryHandler.java
                |       |-- LicenseHandler.java
                |       |-- PoolHandler.java
                |       |-- Root.java
                |-- domainmodel
                |   |-- Pool.java
                |   |-- PoolRegistry.java
                |-- exception
                |   |-- CryptographyException.java
                |   |-- ExceptionHandler.java
                |   |-- FileHandlerException.java
                |   |-- LedgerException.java
                |   |-- LfaRegistryException.java
                |   |-- LicenseGeneratorException.java
                |   |-- ParserException.java
                |-- filehandler
                |   |-- FileHandler.java
                |-- lfa
                |   |-- LfaRegistry.java
                |-- license
                |   |-- LicenseGenerator.java
                |   |-- LicenseLedger.java
                |-- logging
                |   |-- Logging.java
                |-- parser
                |   |-- LicenseParser.java
                |   |-- ZipUtil.java
                |-- persistence
                |   |-- PersistenceController.java
                |-- security
                |   |-- Checksum.java
                |   |-- Cryptography.java
                |   |-- KeyGenerator.java
            |-- resources
                |-- application.properties
                |-- keystore.jks

```

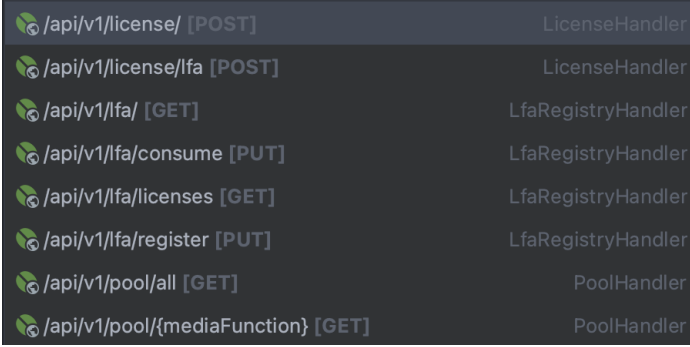
5.6.1 API

As mentioned in Section 5.4.2, a RESTful API is being used for communication between the NMS and the LFA. The API is run through an embedded web server in Spring Web [33], which launches at runtime and handles the endpoints defined by code.

The API consists of three groups of endpoints. The groups are *license*, regarding all licenses in the NMS; *LFA*, regarding all actions in connection with LFA; and *pool*, regarding the pools in the NMS.

For the implemented frontend to get access to the information and functionality necessary, the frontend makes use of this API. This is done due to the unreliability of network IP assignment for more dynamic devices, such as the LFA, where a connection to the NMS is more constant and secure. All requests that are meant to be sent from the frontend to an LFA are sent through the NMS, only to be translated and forwarded to the actual IP address. The API uses standard HTTP response status codes when returning responses to requests [42][43].

The API handlers are written as Rest Controllers, through Spring Web, where the different classes contain handlers for different endpoint groups. As seen in Listing 5.3, the root handler class is defined as a `@RestController` where it handles requests sent to the `BASE_URL`, defined as `"/api/v1"`. The example handler from Listing 5.3 handles a GET-request sent to the base URL, given the fact that the `@GetMapping` value is an empty string. The value string can also be non-empty, where the content is added to the `BASE_URL`, for example `"/root"`. In this case, the URL for the endpoint will be `BASE_URL_VALUE/root`. If the handler should handle POST-requests, the method will need a `@PostMapping` annotation, instead of the `@GetMapping` [33].



<code>/api/v1/license/ [POST]</code>	LicenseHandler
<code>/api/v1/license/lfa [POST]</code>	LicenseHandler
<code>/api/v1/lfa/ [GET]</code>	LfaRegistryHandler
<code>/api/v1/lfa/consume [PUT]</code>	LfaRegistryHandler
<code>/api/v1/lfa/licenses [GET]</code>	LfaRegistryHandler
<code>/api/v1/lfa/register [PUT]</code>	LfaRegistryHandler
<code>/api/v1/pool/all [GET]</code>	PoolHandler
<code>/api/v1/pool/{mediaFunction} [GET]</code>	PoolHandler

Figure 5.2: NMS API Overview.

Code listing 5.3: NMS API Spring Web Endpoint Handler Shell.

```

1  package no.ntnu.nms.api.handlers;
2
3  import no.ntnu.nms.logging.Logging;
4  import org.springframework.web.bind.annotation.*;
5
6  import static no.ntnu.nms.api.Constants.BASE_URL;
7
8  /**
9   * Root is a handler for the root API endpoint, which is not in use.
10  */
11  @RestController
12  @RequestMapping(value = {BASE_URL})
13  public class Root {
14
15      /**
16       * Root endpoint handler method for a request of method GET.
17       * @return {@link String} a message that the endpoint is not in use.
18       */
19      @GetMapping(value = {""})
20      public String rootEndpoint() {
21          Logging.getLogger().info("Root endpoint called");
22          return "This endpoint is not in use. " +
23              "Please check the documentation for available endpoints.";
24      }
25  }

```

Table 5.1: NMS API Endpoint Summary.

Endpoint	Method	Functionality
/license/	POST	Redeems a Top-Level License on the NMS.
/license/lfa/	POST	Generates a SLL and sends it to the LFA.
/lfa/	GET	Getter for all registered LFAs.
/lfa/consume	PUT	Consumes a certain amount of a certain media function from a certain LFA.
/lfa/licenses	GET	Getter for all the LFAs with their licenses.
/lfa/register	PUT	Registers a new LFA on the NMS.
/pool/all	GET	Getter for all NMS license pools.
/pool/{mediaFunction}	GET	Getter for a certain media function pool from the NMS.

General Response

Unless otherwise is described in this section, the response to the requests is of the content-type *application/json*. The response contains either a success message or an error message. An example of both a positive and negative response can be seen in Listing 5.4 and Listing 5.5.

Code listing 5.4: NMS API Positive response body example.

```
1 {  
2   "message": "File uploaded :)"  
3 }
```

Code listing 5.5: NMS API Negative response body example.

```
1 {  
2   "error": "No file uploaded"  
3 }
```

/license/

This endpoint functions as the upload point for the Top-Level License (TLL). After the upload, the files are checked for their integrity, the signature, and the contents. After the parsing, which is described in detail under Section 5.6.4, the TLL functionality is added to the pool, which is described in detail in Section 5.6.6. The license is uploaded as form-data in the request. The key to the data is *file*, and the name of the file is irrelevant, as it is discarded.

/license/lfa/

This endpoint makes it possible to generate an SLL, from the redeemed TLL, of a given length, from a given media function, and send it to a given LFA. The SLL will be packaged and signed, to keep the integrity of the license until it is received by the LFA. The body of the request needs to contain the information necessary to perform the actions triggered by the endpoint. The structure of the response and the JSON-keys are pre-defined. An example request can be seen in Listing 5.6, where all fields are required. The duration is given in seconds.

Code listing 5.6: NMS API /license/lfa/ endpoint body structure.

```
1 {  
2   "ip": "IP.TO.THE.LFA",  
3   "mediaFunction": "J2KHD",  
4   "duration": 300  
5 }
```

/lfa/

This endpoint acts as a getter for a list of all the LFAs registered at the given NMS. The list of registered LFAs is returned in a JSON-structure. An example of the JSON structure response body can be seen in Listing 5.7.

Code listing 5.7: NMS API /lfa/ endpoint response body structure.

```
1 {
2   "lfas": [
3     {
4       "ip": "127.0.0.1:8443",
5       "name": "License File Consumer - Alpha"
6     },
7     {
8       "ip": "127.0.0.1:8453",
9       "name": "License File Consumer - Omega"
10    }
11  ]
12 }
```

/lfa/consume

As a part of the frontend written to demonstrate the PoC, there is functionality enabling the end user to consume license time from an LFA. This endpoint allows for the frontend to utilize the consume functionality within an LFA. The request sent to this endpoint is converted into a request suitable for the given LFA, and sent to the device. The response body structure of this endpoint is identical to the /license/lfa/ endpoint, as seen in Listing 5.6.

/lfa/licenses

This endpoint gathers all the LFAs connected to the NMS, similar to /lfa/, but adds the LFAs licenses. The request body structure is the same as /license/lfa/, as seen in Listing 5.6. An example of the response body structure can be seen in Listing 5.8. It is similar to the response body seen in Listing 5.7, except it adds the licenses currently present at the given LFAs.

Code listing 5.8: NMS API /lfa/licenses endpoint response body structure.

```
1 {
2   "lfas": [
3     {
4       "ip": "127.0.0.1:8443",
5       "name": "License File Consumer - Alpha",
6       "licenses": [
7         {
8           "duration": 300,
9           "name": "J2KHDX",
10          "description": "J2K HD Encoders/Decoders"
11        }
12      ]
13    }
14  ]
15 }
```


/lfa/register

This is the endpoint where the LFAs register to the NMS as available and on-line. When registering, the LFA becomes a part of the *Lfa Registry*. The registry is described in detail in Section 5.6.2. For the endpoint, there are two URL query parameters; the name of the LFA and the port where the LFA server is running. The two query parameters have the keys *name* and *port*, where the name and the port are strings.

/pool/all

As described in Section 5.6.6, the NMS is keeping all redeemed TLLs in pools. This endpoint functions as a getter for all the pools within one NMS. An example of the response body structure can be seen in Listing 5.9.

Code listing 5.9: NMS API /pool/all endpoint response body structure.

```

1  {
2    "pools": [
3      {
4        "mediaFunction": "J2KHDX",
5        "timeLeftSeconds": 5700,
6        "description": "J2K HD Encoders/Decoders"
7      },
8      {
9        "mediaFunction": "UpstreamDataTransfer",
10       "timeLeftSeconds": 3600,
11       "description": "Data transfer for use upstream"
12     }
13   ]
14 }

```

/pool/{mediaFunction}

This endpoint works in the same way as /pool/all, except it returns only one of these pools. The pools returned are defined by the URL parameter *{mediafunction}*, where the media function is the name of the license functionality. The response body structure is close to the one presented in Listing 5.9, except there is no list, as there is just one pool returned.

5.6.2 LFA Registry

For the NMS to be able to distribute SLLs to the LFAs, it is necessary to keep track of the connected LFAs. This is done through the LFA registry. This registry is, through the API, also used for the front end of the PoC, when presenting menu choices and similar.

The LFA Registry is implemented as a singleton class of its own, containing a Java Hashmap as a field and the Singleton instance as a static class field. The Hash Map, named *lfaMap*, is defined with strings as both the key and the value, where

the key is the name of the LFA and the value is the IP address, including the port, of the LFA.

Every time the list of LFAs is requested from the registry, the map of the LFAs is refreshed. By refreshing, it is checking each entry to filter out the entries which are no longer online, or no longer connected for any other reason. The filtered LFAs are then returned. The refresh method can be seen in Listing 5.10. Every time an LFA server is started, it registers with the API of the NMS, which adds it as an entry to the LFA registry.

Code listing 5.10: LFA alive check from NMS.

```

1  /**
2   * Checks if the LFA is alive.
3   * @param ip The ip of the LFA.
4   * @return True if the LFA is alive, false otherwise.
5   */
6  public static boolean lfaIsAlive(String ip) {
7      String url = "https://" + ip + "/api/v1";
8      try (CloseableHttpClient httpClient = getHttpClient()) {
9          if (httpClient == null) {
10             Logging.getLogger().warning("Failed to create http client");
11             return false;
12         }
13         return httpClient.execute(ClassicRequestBuilder.get(url).build(),
14             (ClassicHttpResponse response) -> {
15             if (response.getStatusCode() == 200) {
16                 return true;
17             } else {
18                 Logging.getLogger().warning("Failed to connect to LFA: " +
19                     response.getStatusCode());
20                 return false;
21             }
22         });
23     } catch (Exception e) {
24         Logging.getLogger().warning(e.getMessage());
25         return false;
26     }
27 }

```

5.6.3 Client

To cover the need to communicate with the LFAs' API, a client has been implemented in the NMS. This client performs actions such as consuming licenses on a given LFA, gets information about the LFAs licenses, and similar. The client uses a standard Java HTTP client library created by Apache [44].

An example of how the client communicates with an LFA can be seen in Listing 5.10. In this example, a call is made to the LFA to check if it is still alive and online in the network. For a more detailed overview of the API calls possible to perform towards an LFA, Section 5.7.1 may be referenced.

5.6.4 License Parsing

The license parsing process is an essential component of the system, responsible for verifying, parsing, and handling license files. The main functionality of this process is implemented in two Java classes: *LicenseParser* and *ZipUtil*.

The *LicenseParser* class is responsible for handling the license files and their associated signature files. It starts by unzipping the provided input stream (MultiPartFile) containing the license and signature files using the *ZipUtil* class. Once the files are extracted, the class assigns the file paths to the respective instance variables. Next, the *LicenseParser* verifies the existence of both the files. It reads the content of these files and validates the signature against the embedded public key. This is crucial for ensuring the integrity and authenticity of the license files, and to prevent misuse and exploitation. If the verification fails, an exception is thrown, and the process is halted.

Upon successful verification, the *LicenseParser* checks if the license has already been used, and if so, it throws an exception and handles it accordingly. If the license has not been used, it proceeds to parse the JSON content of the license file. The parsing process extracts the relevant data, such as name, duration, and description, for each license key in the file. The parsed data is then used to update the *LicenseLedger* and *PoolRegistry* instances accordingly. Finally, the *LicenseParser* class cleans up the temporary files and folders created during the process to avoid clutter.

5.6.5 Security Features

The security of the NMS is a vital aspect of the system, as it manages sensitive data and communication between components. It is essential to use strong encryption algorithms and key generation techniques to ensure data confidentiality, which is why the PoC is using the AES-256 algorithm, which was covered in Section 3.2.1 – Cryptography.

The *cryptography* class generates a secret key with a key length of 256 bits for use in encryption and decryption using the *javax.crypto.KeyGenerator* and the AES algorithm. The key is stored as a byte array named KEY, and to ensure that the key is the same every time, the *generateKey* function specifies a preset seed for the secure random number generator.

Code listing 5.11: Cryptography.generateKey.

```

1 // The key used for encryption/decryption.
2 public static final byte[] KEY = generateKey();
3
4 /**
5  * Generates a key for encryption/decryption.
6  * @return The generated key.
7  * @throws RuntimeException if the AES algorithm is not available.
8  */
9 public static byte[] generateKey() throws CryptographyException{
10     try {
11         javax.crypto.KeyGenerator keyGen = KeyGenerator.getInstance("AES");
12         SecureRandom random = SecureRandom.getInstance("SHA1PRNG");
13         random.setSeed(123456L); // Set a fixed seed for the random number
14             generator, can be e.g. customer ID
15         keyGen.init(256, random); // key length of 256 bits
16         SecretKey secretKey = keyGen.generateKey();
17         return secretKey.getEncoded();
18     } catch (NoSuchAlgorithmException e) {
19         Logging.getLogger().warning("Failed to generate key: " +
20             e.getMessage());
21         throw new CryptographyException("Failed to generate key: " +
22             e.getMessage());
23     }
24 }

```

Furthermore, the *cryptography* class utilizes a method called *applyCipher* that takes a message in the form of a byte array and a Mode, which indicates whether to encrypt or decrypt the message. The method will then use the previously generated key to generate a `SecretKeySpec`¹ which will in turn be used to create the final ciphertext.

Code listing 5.12: Cryptography.applyCipher.

```

1 public class Cryptography {
2     public enum Mode {
3         ENCRYPT,
4         DECRYPT
5     }
6
7     /**
8     * Encrypts/decrypts a message using AES encryption.
9     * @param message The message to encrypt.
10    * @param mode The mode to use, e.g. ENCRYPT or DECRYPT
11    * @return The encrypted message.
12    */
13    public static byte[] applyCipher(byte[] message, Mode mode) throws
14        CryptographyException {
15        try {
16            SecretKeySpec keySpec = new SecretKeySpec(KEY, "AES");
17            Cipher cipher = Cipher.getInstance("AES");
18
19            int modeInt = mode == Mode.ENCRYPT ? Cipher.ENCRYPT_MODE :
20                Cipher.DECRYPT_MODE;
21            cipher.init(modeInt, keySpec);

```

¹<https://docs.oracle.com/en/java/javase/11/docs/api/java.base/javax/crypto/spec/SecretKeySpec.html>

```

20     return cipher.doFinal(message);
21   } catch (Exception e) {
22     Logging.getLogger().warning("Failed to apply cipher:" +
23       e.getMessage());
24     throw new CryptographyException("Failed to apply cipher:" +
25       e.getMessage());
26   }
}

```

5.6.6 Pools

Within the NMS, pools are responsible for managing the allocation of media functions. Each pool represents a media function and features a time value indicating the amount of time available for allocation. The Pool class, more specifically, stores information about the media function, duration, and a description of the pool. Furthermore, it offers methods for adding and subtracting time from the pool, allowing the NMS to stay up-to-date when time is allocated to an LFA.

Additionally, the *Pool* class has implemented a property change listeners, listening for changes in the pool object, when, for example, time is subtracted, licenses are added, or the descriptions are changed. When the property change listener is triggered, a log message is created and the pool is serialized. This way, the disk-stored version of the pool objects, defined as a Pool Registry, is always identical to the memory-stored pool objects, ensuring that all pools remain consistent even in the event of a restart or crash. The definition of the property change listener trigger function is seen in Listing 5.13, where the *@Override* annotation is added because the class implements the interface *PropertyChangeListener*.

Code listing 5.13: NMS Pool: Property change listener definition.

```

1  /**
2   * This method gets called when a bound property is changed.
3   *
4   * @param evt A PropertyChangeEvent object describing the event source
5   *           and the property that has changed.
6   */
7  @Override
8  public void propertyChange(PropertyChangeEvent evt) {
9     Logging.getLogger().info("Pool change listener triggered. Updating pool
10    registry.");
11    PoolRegistry.getInstance(false).updatePoolReg();
}

```

The PoolRegistry class manages all the pools in the NMS. As a singleton class, it ensures that there is only one instance of the registry throughout the entire system. The registry maintains a list of pools and provides methods to add, remove, and query pools based on their media function. The list is implemented as a Java ArrayList of the data type *Pool*.

5.6.7 Persistence

As mentioned in Section 5.6.6, the PoC employs persistent features, which is in the form of serialization. This process entails converting the data to a byte stream and encrypting it using the encryption *key* and the *Cryptography.applyCipher* method mentioned in Section 5.6.5. Lastly, the encrypted data is written to a file.

Ensuring integrity of data is important. As a safeguard against potential tampering, the NMS has a self-termination function built in. If decryption fails during the NMS startup, it means important data did not decrypt correctly, which is a big indicator of tampering or unauthorized access. This self-termination feature helps protect the system's safety and reliability, and to prevent misuse and exploitation.

5.6.8 Sub Level License Generation

The sublevel license generation is handled by the *LicenseGenerator* class, which is responsible for creating licenses for specific media functions and durations. The class uses a combination of methods to generate a license, sign it using the private key loaded from the key store, and upload it to the LFA.

The main method, *generateLicense*, accepts as inputs the IP address of the client, the media function, and the license duration. It then checks the input before retrieving the associated pool for the specified media function. The procedure then produces a unique license ID and creates the file path for the license file.

The license string is then generated by merging the pool information, duration, and other required data in a JSON format with the *generateString* function. The *writeToFile* method is used to save the JSON file to disk, and the *signFile* method is used to sign the license file using the private key loaded from the key store.

The REST API endpoint **/generateSubLicense** is responsible for handling the sub-license generation request, which takes a payload containing the LFA IP, media function, and duration. The payload is first parsed, and then the *generateLicense* method is called to create the sublicense. Once the sublicense is generated, it is uploaded to the LFA using the *Client.uploadLicense* method.

5.6.9 Logging

As a part of the operational requirements of the PoC, logging is introduced as a diagnostics tool.

In the package *Logging*, the Singleton class *Logging* is present. This is a class containing two methods and two class variables. The class variables, named *LOG_PATH* and *logger*. They are, respectively, the variable containing the path of the log files and the variable that keeps the singleton instance of the custom logger. To retrieve the logger from a different class, a *getLogger()* method is implemented. This method throws a *NullPointerException* if the second method of the class has

not been called firstly. The method that is required to be called first has the signature: `public static void setUpLogger(String logLevel) throws IOException`. This creates a Singleton logger object, with the given log level, where the log levels are the standard log levels for Java logging [45].

An example of a part of a log file can be seen in Listing 5.14. This log file snippet starts from the beginning of the application, where the logger is initialized. Further, a license is redeemed, where the license functionality does not exist in the NMS. This is why a new pool is created. For the sake of this snippet, the file containing the checksums of the redeemed licenses is failing to be read; this way, the log file does also contain *WARNING* statements. The log level for the logger that created this snippet is set to *ALL*.

Code listing 5.14: NMS Logging file example.

```

1 2023-04-27 09:41:56 INFO no.ntnu.nms.logging.Logging Logger initialized
2 2023-04-27 09:41:56 INFO no.ntnu.nms.parser.ZipUtil Found license.json
3 2023-04-27 09:41:56 INFO no.ntnu.nms.parser.ZipUtil Found license.json.signature
4 2023-04-27 09:41:56 INFO no.ntnu.nms.domainmodel.Pool Creating new pool for
   mediafunction J2KHDX
5 2023-04-27 09:41:56 INFO no.ntnu.nms.domainmodel.Pool Pool registry change
   listener successfully added
6 2023-04-27 09:41:56 INFO no.ntnu.nms.parser.ZipUtil Found license.json
7 2023-04-27 09:41:56 INFO no.ntnu.nms.parser.ZipUtil Found license.json.signature
8 2023-04-27 09:41:56 INFO no.ntnu.nms.parser.ZipUtil Found license.json
9 2023-04-27 09:41:56 INFO no.ntnu.nms.parser.ZipUtil Found license.json.signature
10 2023-04-27 09:41:56 INFO no.ntnu.nms.logging.Logging Logger initialized
11 2023-04-27 09:41:56 WARNING no.ntnu.nms.security.Checksum Failed to read file: test
12 2023-04-27 09:41:56 WARNING no.ntnu.nms.filehandler.FileHandler Failed to open and
   read file: null
13 2023-04-27 09:41:56 INFO no.ntnu.nms.logging.Logging Logger initialized
14 2023-04-27 09:41:58 INFO no.ntnu.nms.api.handlers.Root Root endpoint called

```

5.6.10 Project Initialization

Before the NMS software can be built, some code is run as a preliminary step to prepare the necessary files required for the NMS software to function correctly. The code creates a certificate file by converting the intermediate certificate from the key store to a PEM file format and saves it to a new file. This certificate is essential for sending SLLs to the LFAs as it is sent with the licenses and their respective signatures. Additionally, the code generates two critical files that the NMS software requires to operate. One of these files stores the encrypted serialized pools, while the other stores the encrypted checksums of the redeemed licenses. Should one of these files be removed, the self-termination mechanism activates as mentioned in 5.6.7.

5.7 License File Aggregator

Code listing 5.15: LFA Source code tree.

```

.
|-- CMakeLists.txt
|-- LICENSE
|-- README.md
|-- src
|   |-- App.cpp
|   |-- AppComponent.hpp
|   |-- client
|   |   |-- client.hpp
|   |-- controller
|   |   |-- Controller.cpp
|   |   |-- Controller.hpp
|   |-- dto
|   |   |-- DTOs.hpp
|   |-- error
|   |   |-- error.cpp
|   |   |-- error.hpp
|   |-- file
|   |   |-- fileHandler.cpp
|   |   |-- fileHandler.hpp
|   |-- shared.hpp
|   |-- ssl
|   |   |-- certificates.cpp
|   |   |-- certificates.hpp
|-- test
|   |-- ControllerTest.cpp
|   |-- ControllerTest.hpp
|   |-- app
|   |   |-- ApiTestClient.hpp
|   |   |-- TestComponent.hpp
|   |-- tests.cpp
|-- utility
|   |-- install-oatpp-modules.sh

```

5.7.1 Server

The primary function of the LFA is to act as a running license consumer which can take in any kinds of timed licenses for any functionality a customer would want. For all this to work, the server component of the LFA is essential.

Table 5.2: LFA API Endpoint Summary.

Endpoint	Method	Functionality
/	GET	Used to ping the LFA
/licenses	GET	Fetches currently available license pool
/consume	DELETE	Consumes n seconds from media-function x
/upload	UPLOAD	Accepts valid sublevel licenses and adds to pool

/

First there is the root endpoint which simply returns a “Hello World!” to any calls to it. This is used by the NMS for pre-checking that the LFA is still up and running, before performing requests to any of the other three endpoints. This is done to prevent larger, unnecessary requests to an offline LFA from the NMS.

Code listing 5.16: LFA API / endpoint response body.

```

1 {
2   "message": "Hello World!"
3 }
```

/licenses

The second endpoint simply lists all the LFA’s currently available licenses, their description, and how much time is left for the given license.

Code listing 5.17: LFA API /licenses endpoint response body.

```

1 {
2   "licenses": [
3     {
4       "name": "J3KHDX",
5       "duration": 100,
6       "description": "J3K HD Encoders/Decoders"
7     },
8     {
9       "name": "J2KHDX",
10      "duration": 100,
11      "description": "J2K HD Encoders/Decoders"
12     }
13   ]
14 }
```

/consume

The third endpoint allows for ‘consuming’ any valid submitted time from a registered license. This would in a real world scenario be the actual use of the given license’s accompanying functionality, but within the scope of this Proof of Concept it has been decided to simply allow for a direct ‘consumption’ – or rather deletion – of time from any active licenses through this endpoint. When correctly called upon, it returns the remaining duration of the license which was consumed.

Code listing 5.18: LFA API /consume endpoint response body.

```

1 {
2   "name": "J2KHDX",
3   "duration": 100,
4   "description": "J2K HD Encoders/Decoders"
5 }
```

/upload

Finally, there is the most important endpoint for the LFA which is the upload endpoint. This endpoint represents a very important part of the Proof of Concept, namely verifying the chain of trust, validating integrity, and allowing a license payload to be added to the project. A further explanation of how this is verified and the integrity of the uploaded license are verified can be found in 5.7.2.

5.7.2 OpenSSL

This overview of the implementation of OpenSSL in the PoC will mostly regard the upload endpoint and the steps deployed to verify both the chain of trust and the integrity of licenses.

Foremost, three files are required to be uploaded; the license, its signature, and the intermediate certificate. When called upon, the endpoint will proceed by first reading its own embedded copy of the root certificate and verifying the chain of trust between the supplied intermediate and root certificates.

Code listing 5.19: Chain of trust verification using the root and intermediate certificate.

```

1 // Verify that certificate file is valid
2 int correctCert = system(("openssl x509 -in "+certificate+" -text
   -noout").c_str());
3 OATPP_ASSERT_HTTP(correctCert==0, Status::CODE_400, "Certificate not valid");
4
5 // Verify intermediate cert as derived from root.
6 X509 * intCert = readCertFromFile(certificate);
7 X509 * rootCert = readCertFromFile("../cert/external/root.cert");
8 OATPP_ASSERT_HTTP(cert_verify(intCert, rootCert)==1, Status::CODE_401,
   "Certificate could not be validated!");
9 X509_free(intCert);
10 X509_free(rootCert);

```

The second step is then to validate the integrity of the license file. This is done to prevent man in the middle attacks [46] in the transit of an SLL from the NMS to an LFA. By also requiring the signature of the license in the upload, it can ensure that the payload is both unchanged in the period between now and the moment it was signed, and that it was signed by the previously verified chain-of-trust intermediate private key holder (the NMS). This step is critical to prevent fake or altered licenses from being uploaded and accepted by the LFA, as otherwise a malicious customer could intercept the license on the network and alter its content rather easily.

Code listing 5.20: Integrity validation of sub level licenses.

```

1 // Concatenate the system calls for later use
2 std::string createCommand = "openssl x509 -in "+certificate+" -pubkey -noout >
   intpubkey.pem";
3 std::string verifyCommand = "openssl dgst -sha256 -verify intpubkey.pem
   -signature " + signatureFile + " " + licenseFile;
4
5 // Execute the system calls
6 int createIntPubKey = system((createCommand).c_str());
7 int verifySignature = system((verifyCommand).c_str());
8
9 // Assert success of all previous system commands.
10 OATPP_ASSERT_HTTP(createIntPubKey==0, Status::CODE_400,
11   "Could not derive public key from certificate.");
12
13 OATPP_ASSERT_HTTP(verifySignature==0, Status::CODE_401,
14   "Could not verify license with license signature.");

```

As the source code is written, it allows for re-uploading the same SLL over and over again as long as it was once generated correctly, which represents a security risk. The reason it has not been implemented in the LFA is because the problem has already been proven 'solved' in the NMS and a similar solution would work to prevent the risk here as well. The technique works by saving a list of checksums of each license uploaded to the LFA on the disk, which is then parsed each time a 'new' license is uploaded. If a match were to be found, the license is a duplicate. More on this in Section 5.6.7.

As seen in listing 5.20, the approach to OpenSSL is quite different from listing 5.19 as the code now perform system calls rather than calling a custom-made function. This was done due to time constraints, as the OpenSSL library is infamously poorly documented [47] and the previous custom interpretation took quite some time, even though it should have been a simple implementation. Therefore, making system calls directly to the OS installation of OpenSSL proved much faster for development.

However, this could be a security risk as there is no control of what exactly 'OpenSSL' calls to in the OS. A simple alias to always return 0 for all system calls to 'OpenSSL' would entirely break the integrity check, as the result would always be 0 (meaning success). Blindly trusting system calls is short-sighted and for a real world secure implementation, all functionality should be custom-made in the code-base as in listing 5.19.

Finally, the LFA uses an implementation of OpenSSL to host the server on HTTPS to secure the line for an even stronger protection of the licenses payload between the NMS and the LFA. This is still vulnerable to attacks such as man-in-the-middle attacks and cannot be entirely relied on, which is why further integrity checks are performed in the upload endpoint.

5.7.3 Client

One slightly out of scope component of the LFA is the client. It only exists to make an initial call to the NMS at a static IP in the network at the very beginning of the program. This is done so that the NMS can create a list of connected LFAs for further ease-of-use to make mass-requests to all or any connected LFAs in the future. In the real world, this should be done properly through a fully fledged network management system. An issue with this approach is that if an LFA is launched before the NMS, the NMS will never know that the LFA is running, as the request was made earlier and failed. The LFA does not change its behaviour if it does not have any NMS available, and the NMS will not attempt to establish a connection to the LFA, as this would be even further out of scope.

Code listing 5.21: A simple client for sending a PUT request to NMS.

```

1 class Client : public oatpp::web::client::ApiClient {
2   API_CLIENT_INIT(Client)
3   API_CALL("PUT", "api/v1/lfa/register", getResource, QUERY(String, name),
4           QUERY(String, port))
};

```

The code in Listing 5.21 defines the client to perform a single PUT request with two variables in the URL; namely the name of the LFA and the port it exists on. The NMS also requires the IP for future request relays, but this is fetched from the request itself, as this presented as a far easier technique to grab the LFA IP. Initially it was attempted to fetch the port from the request, but it was discovered that the client itself would not run at the same port as the LFA server and would be randomly assigned an unassigned port on the system. Therefore, if the request port was cached, the NMS would never receive a response from the LFA as the client would have already been shut down and the server would be running on a different pre-defined port.

5.8 Frontend

For demonstration purposes, there was also developed a website hosted alongside the NMS to interact with the various API endpoints on the NMS.

This website was built with vue.js and the accompanying plugin Vuetify to quickly build UI with the design language of Google's Material Design. The UI allows for several key interactions with Proof of Concept functionality.

First the website allows for uploading TLLs which, if successful, adds the license data to the list of available licenses below. Next to this license pool, the website allows for generating SLLs and sending them to any connected LFAs. The UI for this can be seen in Figure 5.3.

When loading the website, a dynamic list of all connected LFAs are available at the bottom of the screen, with the license pool for each LFA. This list also reloads

with new data each time the user generates new Sub-Level Licenses. In the license pool of all the LFAs, the UI allows for consuming any amount of available duration from any license. The UI for this can be seen in Figure 5.4.

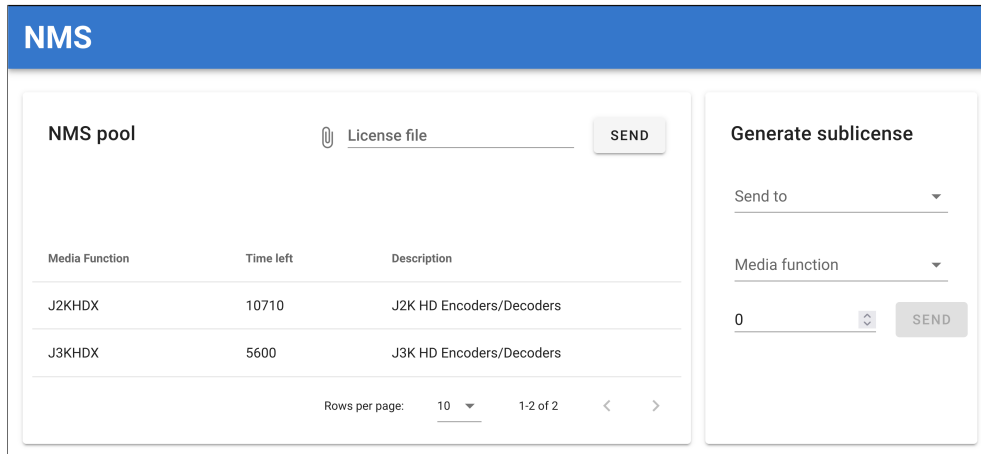


Figure 5.3: Screenshot from front end website (NMS part).

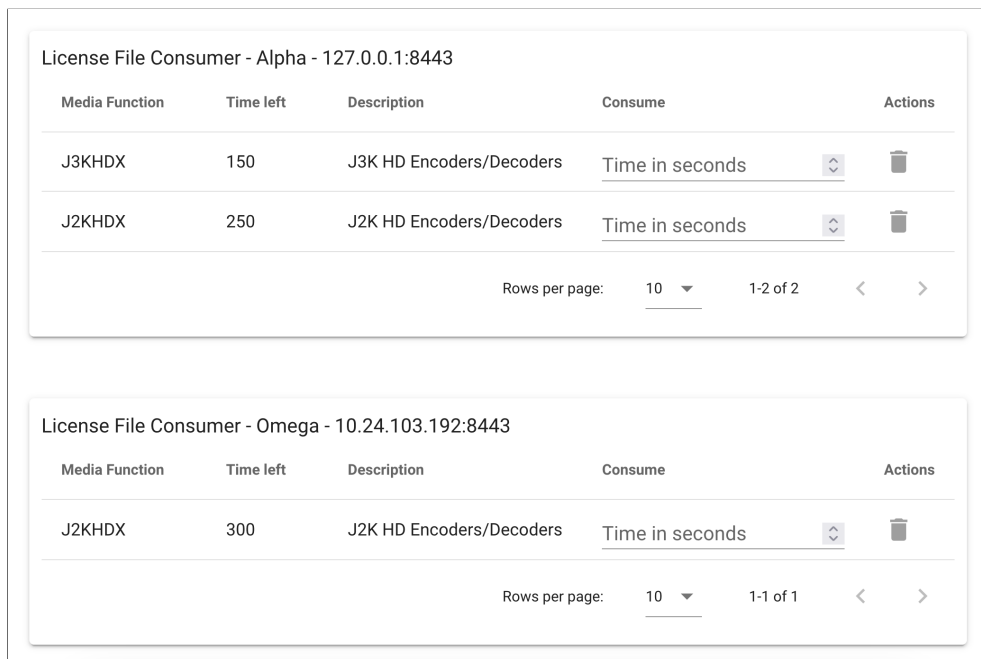


Figure 5.4: Screenshot from front end website (LFA part).

The webpage itself can as stated and shown earlier see all available LFAs and interact with them, but it never makes requests directly to them. The webpage makes requests only to the NMS which then makes requests on behalf of the webpage to the various LFAs. Be it fetching a list of all available LFAs or sending LFA #1 a license for x of n units of time, it all goes through the NMS. This keeps the

networking simple for the webpage and results in a list of six different NMS API endpoints it interacts with.

Code listing 5.22: Calls to NMS endpoints using Axios.

```

1 // Axios functions to call NMS endpoints
2 const fetchLfas      = ()      => {return api.get('/lfa/licenses')}
3 const fetchPools    = ()      => {return api.get('/pool/all')}
4 const fetchLfaNames = ()      => {return api.get('/lfa')}
5 const generateSubLicense = (payload) => {return api.post('/license/lfa', payload)}
6 const consumeLicense = (payload) => {return api.put('/lfa/consume', payload)}
7 const uploadFile    = (payload) => {return api.post('/license', payload,
  { headers: {"Content-Type": "multipart/form-data"} })}

```

5.9 Secrets Handling

For the purpose of the PoC, all secrets have remained embedded within the repos for ease-of-use for anyone who would need to test the system. This is *not* good practice and should not be done in any real-world scenario. Given the fact that this is a proof of the concept and the secrets are not real-world secrets, the group elected to leave them in each respective code-base.

Secrets and their derivatives include: a root private key for the LFS; an intermediate private key and intermediate certificate within the NMS, stored in a keystore; and finally the root certificate within the LFA. Additionally – and less importantly – a private key with an accompanying certificate exist within the LFA for hosting the API on HTTPS. This key & certificate pair is not a part of the main chain-of-trust and should be generated separately.

Some secrets are included inside the compiled versions of the code, such as the passphrase for the keystore containing the intermediate certificate and its private key. While this is fine in the sense of a Proof of Concept, this is rather unpractical in a production environment. This is because a recompilation is necessary in order to swap out the keystore, the pass phrases, and the public key. In the case of this PoC, where the concept is built on a new compilation for each customer relation, the impracticalities are reduced, as it is necessary anyway. This is also supported by the need to keep the secrets hidden from the customer, who has physical access to the device.

5.10 Licenses

As this project handles license management, licenses are necessary. The concept requires the use of two different licenses: Top-Level License (TLL) and Sub-Level License (SLL). These are, respectively, the licenses being redeemed by the NMS and the LFA. This section will discuss some of their key features and their differences.

Code listing 5.23: Top-Level License example.

```
1 {
2   "info":{
3     "date":"2021-08-03 07:49:51",
4     "customer":"TV2",
5     "issuer":"Root",
6     "uid":123
7   },
8   "license":{
9     "keys":[
10      {
11        "name":"J2KHDX",
12        "duration":100,
13        "description":"J2K HD Encoders/Decoders"
14      }
15    ]
16  }
17 }
```

A TLL is the license sent from the licensing company to the customer. An example of a TLL can be seen in Listing 5.23. It contains information about one or more licensed functionality. The functionality is defined by a name, a description, and a duration in seconds, where the functionality is defined as a list of *keys* under the *license* part of the JSON file. For the context of this PoC, the description of the licensed functionality could have been excluded, but was added to improve the graphical user interface of the front-end. The none-license part of the JSON file is the *info* part of the content. This part contains the name of the intended customer, the name of the issuer, a unique identifier² (UID) for the license, and a timestamp. The name of the intended customer and the issuer is included to create the license more human-readable. On the other hand, the timestamp and the UID is included to be able to always have a unique checksum of the license, for securing the redeeming. An SLL is structured in the same way as the TLL except for only one difference; it includes a unique identifier for the intended receiver, which in the Proof of Concept is the unique name of the LFA. Within the general concept of OLM the SLL also supports several licenses within the payload, but this support has not been implemented on the NMS end of the equation in the PoC.

The two different license types are communicated in a package containing several files. For the TLL, it is communicated in a package containing the license file and the signature. In this case, the signature is used for verifying the legitimacy of the license; if a trusted source has created it. The SLL is communicated in a package containing the same files, in addition to the intermediate certificate of the NMS that generated the SLL. The intermediate certificate is used to verify the chain of trust; verify if the NMS is in the correct chain of trust, between the trusted source and the LFA.

²https://en.wikipedia.org/wiki/Unique_identifier, visited April 27th, 2023

Code listing 5.24: Sub-Level License example.

```

1 {
2   "name": "lfa_omega",
3   "info": {
4     "date": "2021-08-03 07:49:51",
5     "customer": "Riot Games",
6     "issuer": "Root"
7   },
8   "license": {
9     "keys": [
10      {
11        "name": "J3KHDX",
12        "duration": 100,
13        "description": "J3K HD Encoders/Decoders"
14      }
15    ]
16  }
17 }

```

5.11 Hosting and Version Control

5.11.1 Distribution

Proof of Concept is implemented as 4 standalone applications, including the frontend. This means that they may be distributed and run isolated from each other, as long as the IPs are correct, the ports are available and firewalls, routers and similar are opened for the required network traffic. In the case of the PoC, please see Table 5.3 for an overview of the ports.

Table 5.3: Proof of Concept Services Ports.

Service	Port
Frontend	8080
NMS REST API	8090
LFA REST API	8443
License Signing	N/A

5.11.2 Deployment

Network Management System

The NMS is, as mentioned in Section 5.3.2, based on Java and Maven. In order to initialize the project, it is necessary to run the *Init* class' main method. This is done through running `mvn run` in the project root. This creates a Jar file, which later can be run with a Java command, more specifically

`java -jar target/nms-software-1.0-SNAPSHOT.jar`. The actions performed during the *Init* class' main method is described in Section 5.6.10.

The Network Management System is compiled with Java and can run platform independently. At the time of writing, it has been tested on macOS, Linux Debian, and Windows 10 & 11.

License File Aggregator

To initialize an instance of the LFA repository, a couple of things need to be done. Firstly, the project requires an installation of CMake [48] and the G++ compiler, as well as the Oatpp web framework. To Install Oatpp, simply run the `utility/install-oatpp-modules.sh` bash script in the project. To compile and build the project, a `build` folder has to be created within the source root folder. Once created and entered, running `cmake ..` initializes the CMake environment. Finally, to build the project, the command `make` has to be run in the same folder.

License Signing

The license signing is written as a script, and not as an intricate application. This makes it necessary to just run the script, preferably through command line, as there are arguments necessary.

The necessary arguments are the license file to sign and the private key file to use when signing.

The license signing script is written as a Bourne Shell Script, thus it can be run on Unix based operating systems only, like macOS and Linux, by default [49]. However, it can also run on a Windows machine with a shell installed, if the first line of the script is altered according to the shell path, as seen in listing 5.1.

Frontend

The frontend is, as mentioned in Section 5.8, a JavaScript application built using Vue, with NPM as a package manager.

For the application to run, the necessary packages will need to be installed. In order to do so, run `npm install` with the project root as the working directory. After this, it is possible to either build a product for deployment or run a local, hot-swappable³, development intended version of the project, on a locally hosted web server. This can be done through either `npm run build` OR `npm run dev`.

The frontend is completely platform independent, and can be run on any operating system supporting NPM and Node.js [50].

³Automatically including changes to source code while running

5.11.3 Git

For the Proof of Concept, GitHub⁴ has been used for version controlling.

The four different components have used their own repositories in the version controlling system, where they are gathered in one GitHub organization, for the purpose of this project.

Each repository has their own README, a file containing information about the repository, how it is used, and similar. The README files of the different repositories can be found as appendix K, L and M.

When working with version controlling, branching has been used. When branching, the source code is copied into parallel tracks of changes before the new branch containing new functionality, is merged into the main track. How this functionality has been used in a quality assurance perspective, is described in Section 5.12. To make sure that no broken functionality has been merged into the functioning code base, or similar, branch protection has been used. This is a measure where it is required for code to pass the quality assurance (i.e., unit tests, CI/CD scripts and pull requests). On GitHub, this functionality is behind a paywall. As a consequence of this, the branch protection has been an internal agreement in the group, rather than a software enforced ruleset. How this has been used in practice has been described in Section 5.12.

The repositories containing the PoC can be found at: <https://github.com/orgs/ntnu-2023-bcs-bidata-bprog-g3/repositories>.

5.12 Quality Assurance

Throughout the project, quality work has been important to the group, and for just that, a couple quality assurance techniques have been utilized.

When working on a PoC, the source code is relevant to put under quality assurance. To do so, several measures have been implemented in the project work. As mentioned in Section 5.11.3, Git versioning has been used through GitHub. When using branches in the version control system, there is always a need for at least one other project member to review the pull requests being submitted after fully implementing the given functionality.

In addition to this, the pull request (both new and old code) will need to pass the CI/CD⁵ implementation. As this CI/CD includes running the unit tests, the quality assurance for new source code also includes unit tests. For a simplified version of the CI/CD implementations, see listing 5.25 and 5.26. This makes sure the newly implemented code is functioning according to intentions before it is merged with

⁴<https://github.com/>, retrieved April 25th 2023

⁵*Continuous integration and continuous delivery*, implemented using GitHub Actions [51]

the existing source code base. This is being done for both of the two main source code repositories, the NMS and the LFA.

Code listing 5.25: NMS source code repository CI/CD implementation (simplified).

```
1 name: Java CI with Maven
2
3 on: [push, pull_request]
4
5 permissions: write-all
6
7 jobs:
8   build:
9     runs-on: ubuntu-latest
10    steps:
11      - name: Checkout
12        uses: actions/checkout@v3
13      - name: Set up JDK 11
14        uses: actions/setup-java@v3
15        with:
16          java-version: '11'
17          distribution: 'corretto'
18          cache: maven
19      - name: Build and run tests
20        run: mvn --batch-mode --fail-at-end package
21      - name: Surefire Report
22        if: ${ always() }
23        uses: ScaCap/action-surefire-report@v1.4.0
24        with:
25          report_paths: 'target/surefire-reports/TEST-*.xml'
26      - name: Upload Report
27        uses: actions/upload-artifact@v3.1.2
28        with:
29          name: report.xml
30          path: target/site/jacoco/jacoco.xml
```

For the CI/CD to be able to test the implemented functionality, unit tests are necessary. These have been implemented in the source code repositories, according to the standards per language. For example, in the NMS repository, unit test classes have been implemented in the same class structure as the ordinary classes, as seen in listing 5.2.

The quality and readability of the source code is important for further development and maintenance. In the NMS repository, the code linter *Checkstyle* has been used with a custom ruleset [52]. The custom ruleset can be seen as appendix O.

Code listing 5.26: LFA source code repository CI/CD implementation (simplified).

```

1 name: build
2
3 run-name: Build project
4
5 on: [push, pull_request]
6
7 jobs:
8   Build-and-run-tests:
9     runs-on: ubuntu-latest
10    steps:
11      - name: Check out repository code
12        uses: actions/checkout@v3
13      - name: Install dependencies
14        run: |
15          cd utility/ && sudo ./install-oatpp-modules.sh && cd ..
16          sudo apt-get install libboost-all-dev libssl-dev -y
17          git submodule update --init --recursive
18      - name: Build project
19        run: mkdir build && cd build && cmake .. && make
20      - name: Run tests
21        run: cd build && sudo ./license_consumer-test

```

5.13 Discussion

5.13.1 Administrative Decisions

The Proof of Concept is not defined as production ready. This is due to several reasons, but the most importantly is the handling of secrets. As mentioned in chapter 5, the PoC has been made with ease of use in mind, specifically regarding the assessment of the project. Key pairs have been left in the online code repositories and secrets have been left in the source code text. It is important to underline the fact that this exact instance is not secure, and the key pairs cannot be re-used for future instances. By handling the secrets this way, the code repositories are prepared for *clone-and-run*, where it is possible to clone the code repositories, compile, and run the application without any setup, like key generation, necessary. If not, it will be necessary to generate two key pairs, one derived from the other, in addition to two certificates derived from the key pairs and a separate HTTPS key pair. The best practices are to handle the secrets in a more secure way, but after discussion within the group and with the thesis supervisor, it was decided to do it this way [53]. However, a guide on how to generate the necessary keys, the location, and similar, relevant to the initial setup of the Proof of Concept, is appended. It can be found in Appendix J or in the license file signing script repository (<https://github.com/ntnu-2023-bcs-bidata-bprog-g3/license-file-signing>).

5.13.2 Proof of Concept VS. Concept

The concept, reviewed and presented in chapter 3, is defining the shell of how a system should be implemented. Many things are still left for decision when the implementation of the Proof of Concept is being performed.

An example of this is the communication between the licensing company and the customer hosting the system, which in this PoC has been deemed irrelevant, according to Section 5.4.2.

Another example of this is the communication between the NMS and the LFA, which in the PoC has been covered by RESTful APIs with JSON request and response bodies. This communication channel could, however, be covered by a variety of other communication methods, like the use of GraphQL, WebSocket or a computer vision script reading images sent through the postal services⁶ [54] [55]. By using RESTful APIs, the communication part of the PoC is deemed to be according to the concept. WebSockets, designed for continuous communication, are not relevant for an implementation where the communication is only needed sporadically, while GraphQL is designed for APIs with a wider range of options for the requests [56] [57]. Further reasoning behind the choice is presented in Section 5.4.2.

The concept does not define which programming language to use in the implementation, and the language choice may have been different from the ones chosen. According to requirements given by Nevion, as seen in Section 5.2, the LFA had to be implemented in C++ and the NMS had to be implemented in either Java or Scala. Java was chosen, out of the two, due to experience from earlier courses in the study program.

The concept requires storage within the LFA, among other things, for storing license pools and the checksums of redeemed licenses. As the source code is written right now, this is not implemented. As mentioned in Section 5.7.2, this does pose a security risk, because it is not implemented according to the concept. This is due to several reasons, mostly because of the lack of time and the fact that this exact functionality is proven in the NMS. This is although one of the few deviations between the PoC and the concept.

5.13.3 Security

As a basis for the concept, security is important. As a result, the Proof of Concept had to be done securely, according to the concept.

One of the languages chosen, however, are not optimal. Java is a language that is hard to compile in a way where the compiled source code is not easily reverse-engineered into readable code⁷. As a result of this, the group was put in a dilemma

⁶This is *obviously* a joke

⁷<https://www.kali.org/tools/ghidra/>, visited April 26th, 2023

between making the PoC secure to the necessary level or following the requirements given by Nevion, as there were no relevant solutions to make the compiled source code hardened enough. After a thorough walkthrough, Java was decided to be the programming language of the NMS despite its flaws.

During the development of the NMS in the PoC, there was no sufficient time to investigate ways to implement HTTPS on the web server as an enhancement over HTTP. HTTPS is by all means the preferred solution for network traffic, and is also preferred for the network traffic accessing the NMS [58]. The content of the network traffic is, however, not deemed sensitive enough to prioritize this improvement. In addition to this, the implementation exists in the web server of the LFA, which supports HTTPS.

The concept requires the PoC to utilize cryptographic techniques for signing and verifying signed TLLs and SLLs. For the NMS, the Security package from the Java standard library is used and is defined as a secure way to do it. In the LFA, however, sub-processes have been used. To do so, method calls to `std::system`, calling the local installation of OpenSSL, have been performed. By doing so, a security risk is created, where, for example, an alias to a different script can make all licenses verifiable [59]. However, the assumptions of the LFA hardware security mentioned earlier prevent this risk.

As a result of these aspects, the PoC do open holes in the security, for example through the choice of programming language, which are not present in the concept. This means that a possible (production-environment) implementation of the concept will need to have its security reviewed separately in order to properly prevent and mitigate new security threats which are not present in the concept.

5.13.4 Is the Concept Proven?

The concept presents a general solution to the issue, where, as mentioned, a lot of decisions are left for the implementation of the concept. A key factor to this is whether the concept is, foremost, implementable and, second of all, a viable solution to the issue. By implementing the PoC, the concept is proven implementable. The question remaining is whether the PoC proves the concept as a viable solution to the issue. When reviewing the PoC, all the necessary functionality from the concept have been implemented and tested to an extent where they are deemed functioning.

From the functional requirements, given in Section 5.2, all four requirements are being met: the NMS is able to store TLLs; the NMS is able to generate and sign SLLs; the NMS can send and the LFA can receive SLLs; as well as the fact that the LFA can redeem and consume SLLs. Two out of four of the non-functional requirements are also met, as the requirements to the implementation language has been met. The rest of the non-functional requirements, in addition to some operational requirements, however, are of less of a concrete nature. Whether or

not the implementation is safe, with regard to integrity of the source code and the serialization, is abstract. As mentioned, there is some security functionality missing, for example storage of the checksums of the redeemed licenses in the LFA, although this is reasoned for and seen as proven in the NMS. Similarly, the group believes the PoC is reliable and user-friendly, although the crux of security is the balance between user-friendliness and security; a PoC with fewer threats, and more preventative measures, may be at the expense of the usability requirements. Logging is also implemented. Given this achievement of the requirements, there is little reason to say that the PoC does not prove the concept, in regard to the requirements themselves.

As mentioned in Section 5.13.3, the security aspect of the implementation is not perfect. This is due to the choices made and requirements presented in the course of the implementation phase of the PoC. The group do still believe it is possible to create an implementation of the system concept which do not open any security holes that are not discussed in the security review of the concept (see Chapter 4 – Security Review).

All in all, the PoC does prove the concept – both in feasibility and functionality; the concept is possible to implement, and it is possible to do in a functioning way. The fact that the concept is general makes it possible to implement versions of the concept that suits many different needs; some could need a higher degree of security, while some could need a higher degree of usability.

Chapter 6

Discussion

6.1 Project Retrospective

The whole project started with a kick-off at Nevion headquarters at Lysaker, outside Oslo. Here, the group got a detailed presentation of the project task, what Nevion does, their product portfolio, and some of their customers. This made it easier to understand exactly what they were looking for when they made the task description. After the kick-off, the work on the concept began: different technologies, cryptographic techniques, and integrity techniques were researched. A first draft of the concept, close to how it looks today, saw the light of day in the beginning of February. Quickly after this, the implementation of the PoC began, where the group split up into working on the NMS and the LFA. The following month and a half, the implementation process continued, including new iterations of the concept adapted after experiences with the PoC. In the middle of March, the work on the security review began, and lasted approximately a month. When the security review was finished, in the beginning of April, the work on the first draft of the report began; it was finished no later than the early beginning of May. This gave the group close to three weeks of iterations containing rewrites, add-ons, report reviews, and proofreading.

Early in the project period, the group discovered that the task at hand was more difficult to solve securely than first anticipated. The reason for this, is the requirements of an offline network and on-premises equipment. The problem is, as mentioned in the report, no general truly safe solution for an entirely offline license management system exists on the market, which would have made a potential general solution from this project groundbreaking for the industry.

For these reasons, the group spent the first couple of weeks coming to the conclusion that no all-purpose general solution is likely to be invented during this thesis, and that the focus should rather be on developing an as-close-as-possible rendition of a safe OLM system and then comprehensively showcase the faults of

such a system. The Security Review is in essence this showcase.

Further challenges met during the project period was another course running in parallel during the first half of the project period. This course often took several days of the week from two out of three members, which often left the third to work alone in shorter segments. The course finally culminated in an exam which caused a week-long hiatus from the project.

As outlined, the PoC was written between the concept development and the security review of the concept. There are several advantages and disadvantages to this, where the disadvantages speak in the direction of writing the PoC after the security review. One of the greatest advantages of maintaining the order that has been used, is the fact that the PoC development process makes it easier to discover the security threats that are present in the concept, that may be useful for the security review. By implementing the PoC before the security review is performed, however, makes it hard to include the mitigation and prevention measures that the work on the security review introduces. Although, a better balance can be to work on the development of the PoC parallel to the security review. The disadvantage of this is the context switching introduced by this approach, which make it harder to focus on one thing at a time.

Other than this, the group started on the thesis itself right after completing a working prototype of the Proof of Concept (PoC). This allowed the group to write from recent memory about core functionality, and the progress experienced an acceleration. The progress took a considerable hit once the group started on the Security Review, as nearly all aspects of each paragraph required research and discussion amongst the group. The thorough nature of the review resulted in approximately a month of work, and the first draft was finished during the end of April.

In the beginning of the project, the group agreed to use the Campus premises as working environment. By doing so, the cooperation, discussion, and dialogue within the group in the course of the project work was made easier. Although, when the project work started to move from concept & PoC development, and towards report writing, some group members tended towards using a more hybrid solution. This meant spending some days, or parts of days, working from home, and was approved within the group. Towards the end of the project, as report reviews and discussions got more relevant, the Campus premises once again became the norm for all group members.

6.2 Project Plan

As a part of the project, a project plan was written before the project commenced. The project plan has been mentioned several times during the report, and can be found as appendix C. It contains a Gantt diagram, describing the planned progression of the project; a ruleset and guidelines for the use of version control systems; a definition of the project goals; and a description of the group member respons-

ibilities, to mention some things. The project plan also included a set of group rules signed by the group members.

The Gantt diagram, attached as appendix D, has been used as a guide to the preferred progression of the project. Overall, the progression has been followed. The progression plan gives the group a couple of weeks margin at the end of the project; the plan was to finish the thesis in the beginning of May. However, this margin was used up, and a longer thesis writing period became necessary. Because of this margin, the group extended the thesis writing period without reaching the final deadline. The diagram also contains a structure of the SCRUM sprints, where the sprint number is set per week. After the decision to transition to bi-weekly sprints in the end of April, this part of the diagram was no longer accurate. Overall, the diagram was followed, and there are no immediate points of improvement discussed within the group.

The project plan set clear descriptions for the responsibilities of the different group members, in addition to defining how the group should work with this division of responsibilities. All group members have however worked on all domains of the thesis; the group cooperated on part one before moving to part two, etc., where the member responsible for the given domain lead the group. This way, everyone got experience with leading the group and taking responsibility.

The group leader during the project has been Sander Osvik Brekke, where the only real use of a group leader is in the case of disagreements, arguments or similar, where the group leader would then try to mediate within the group. There have, although, not been any need for this, because the group have discussed decisions internally and professionally, in order to avoid arguments and disagreements that could affect the thesis work. In addition to this, the group have focused on making the group environment open and available, where anyone can present their ideas and ask questions, and receive a professional response.

Through the use of the project plan, the thesis work was well organized, where everyone was aware of, among other things, how things were going to be done, how the decision-making processes should go and how the version control system should be used.

6.3 Version Control & QA

The group has utilized GitHub for version control throughout the project, primarily for the implementation process of the Proof of Concept (PoC). Additionally, version control has been employed during the report writing phase using Overleaf's built-in solution.

As part of the project plan, a set of internal rules for version control system usage was established, covering aspects such as commit messages, pull requests, branching, and similar practices. Some rules, like the mandatory passing of tests before

merging a pull request, are automatically enforced by GitHub, triggering an error if not satisfied. Other rules have been upheld by the group members themselves. In hindsight, the group collectively agrees that the rules were followed to the best of their abilities. However, there were instances where bug fixes, patches, and similar changes were directly pushed to the main branch, which is considered poor practice. Fortunately, the issue did not escalate to a significant extent. One rule stipulated in the project plan was that pull requests must be reviewed by another group member. This posed challenges in cases where pair programming was employed, since it was unclear who should review the code that all members contributed to. To address this, the group believed that everyone participating in the pair programming sessions constituted sufficient reviewing. Although code reviewing proved difficult when substantial changes were made to the codebase, the group always made an effort to conduct reviews, and no exceptions were made to this rule.

The main branch was designated as a protected branch, forbidding direct commits and requiring all changes to be made through pull requests from different branches. This safeguard ensured that the code in the main branch was functional, runnable, tested, and documented. Consequently, a significant amount of code was merged into the development branch instead of being frequently merged with the main branch. This was primarily due to incomplete functionality. In hindsight, the group acknowledges that they should have invested more effort into avoiding the main branch being practically empty until the later stages of the development process. While it is nearly impossible to completely eliminate merge errors when working with branches, the group addressed such incidents by fixing the errors locally on one member's computer after manually pushing fixes to protected branches. Only a few such incidents occurred. Furthermore, the extensive use of pair programming should be noted. The preferred software for pair programming was JetBrains' *Code with me* functionality, allowing multiple programmers to collaborate on the same source code with live updates of each other's edits. In these cases, the commits following the session were attributed to the group member who hosted the *Code with me* session. Consequently, the distribution of commits, particularly in the NMS repository, does not accurately reflect individual contributions.

Regarding the report writing process in Overleaf, version control also involved manually downloading copies of the \LaTeX source files after completing major work as a precaution against errors or server failures.

The quality assurance of the PoC has also consisted of a continuous implementation and continuous development pipeline, implemented and integrated in GitHub through the use of their *Actions* system. The use of GitHub Actions have been described in the report. The pipeline was written and implemented at an early stage, where it builds the source code on pushes and pull requests. Later, the running of unit tests was added to the pipeline. However, the number of tests were low, and the test coverage was not at a satisfying level. The low number of tests is mostly

due to the lack of time, and is something that would be improved given more project time.

6.4 SCRUM

SCRUM has been an important tool during the project's course, as it has helped organize the project into smaller parts, and has generally been functioning well. Through the use of SCRUM, it was easy to find new tasks to work on after completing a previous task.

A series of columns with an associated set of rules, first defined in the Project Plan, was used for each sprint. The 'Review' column was throughout the sprints an obligatory stop for all issues claimed finished by an individual in the group. When an issue had reached the review column, only another group-member could move it, either back to 'working on it' or into the 'done' column. This ensured that all pieces of work that had been done got two sets of eyes on them before being accepted as done. Effectively working as a *two-thirds vote* for the work to be regarded as done, before the group accepted it and moved on. As always, there were discovered exceptions to the rules during the project period, which were granted amnesty from further scrutiny. As an example: who reviews the issue if everyone participated in the work?

It was not always possible to estimate an issue work-load accurately, and there were some instances where the team missed the mark. In these cases, the SCRUM board was not empty when the sprint ended. To not break the flow of work, these tasks were kept and added to the next sprint. The issue that the group faced, was that some tasks were too large and never got finished within the given sprint. However, this emphasizes the importance of well-defined and broken down tasks, as they are more manageable for the group to handle effectively.

When the project progressed to the thesis writing period, one-week long sprints turned out to be too short, and two-week long sprints were introduced. During the period of implementing functionality in, for example, source code, the issues were smaller, which was not the case during the thesis writing. The tasks, which were often defined by thesis sections, got larger, and it became more difficult to finish a set of tasks within the sprints with a duration of one week. The doubling in the sprint duration happened from sprint number 12, as outlined in the Scrum Sprints Summary. The group was happy with the decision, and the flow of the sprints got improved with the longer durations.

6.5 Meetings

The meetings have been beneficial in helping the group work together and achieve its goals. Weekly meetings with the thesis supervisor helped us stay on track and focus on the work at hand. These meetings allowed the group to discuss any issues or concerns that arose. Similarly, bi-weekly meetings with Anders Dale, the representative from Nevion, provided valuable guidance and feedback on our progress during the development phase. However, there were times when meetings with Nevion were not necessary or productive, such as when the group progressed from development of the PoC to writing of the thesis. On these occasions, the meetings were cancelled as the Nevion representative had nothing to add to the current work at hand.

In addition to this, there were often held unofficial meetings within the group, where decisions were made, logic was discussed, and solutions were discovered. These meetings were not planned, thus, no meeting notes were created. Similarly, there were no meeting minutes produced either, as the outcome of the meeting often was put into effect immediately.

6.6 Thesis

The work on the report started early and slowly, alongside the end of the work with the Proof of Concept. This was a result of the parallel course that two of the three group members attended, which periodically took up a lot of time. During this time, the third group member started writing some parts of the report.

As mentioned, in the beginning of May, the first draft was completed. This first draft was, however, an already reworked version, which the group had put more effort into than what would normally be expected from a “*normal*” first draft. Thus, the remaining three weeks were spent reworking a draft that had already been through excessive work. This is not a bad thing in the groups’ opinion, as it allows for a deeper dive to into, and polish for, several important aspects of the thesis.

While working on the concept, the Proof of Concept, and the Security Review security review, there was little feedback given by our supervisor, except making sure the group was on track, according to the project plan timeline, and the calendar in general. When the work on the remaining project report began, the feedback saw an increase, and became continuous. The continuous feedback allowed the group to always have something to rework and improve, such that there were no downtime due to any lack of feedback.

6.7 Engineering Values and Perspectives

Within engineering, topics range from ethics and economy to health and environment that needs to be acknowledged, weighed, and taken into account. As with every new system developed, computer software implemented, bridge built, and tool designed, there are both negative and positive consequences and repercussions. The Offline License Management system is no exception, and there are aspects that need to be discussed and taken into account.

Economically speaking, the OLM system helps customers, the companies consuming licenses, to easier afford the license functionality necessary. The concept introduces an alternative to lifetime licensing, which can be hard to afford and has a higher up-front investment requirement. With the new concept, licenses can be sold as they are being used, and they become less of an investment and more of a running cost. Although, this may be more expensive over time, the initial investment is no longer present, and the threshold to invest is removed. Through the distribution of licenses with a lower threshold of purchase, it is more likely that a greater amount of companies survives after their startup period, creating a more diverse market, with more jobs and more local companies. The distribution of affordable licenses may also make it easier for worldwide trades, where companies in developing countries, which need the licensed functionality, can afford it. This also has benefits for the licensing company, where they are able to grow a larger customer base, with smaller customers which would normally not have the possibility to invest in the lifetime licenses.

In an environmental perspective, the solution is working towards decreasing the environmental footprint of the software business. The data centers, containing both computational power and data storage, are responsible for 2% of the global carbon emissions [60]. When using an offline licensing solution, the network traffic is decreased drastically across the board. The traffic is decreased to TLLs only; the amount of central servers necessary is decreased to almost none, except the storage of root private keys; and the amount of stored data is decreased to only the root private keys, in comparison to storing the DRM software and necessary associated data. Through this decrease, the amount of electrical power necessary can be decreased, thus decreasing the environmental footprint of the software, as most sources of electrical power are none-green.

The concept of OLM also plays a role in the ethical aspects of engineering. By developing a system to safely handle licensed offline functionality, thus handling offline Digital Rights Management, the system may help companies secure and protect their intellectual property rights and proprietary software. By doing so, the act of cracking and distributing pirated copies of software and licensed functionality, which cost the software business more than 46 billion USD between 2015 and 2017 [61], becomes harder.

Chapter 7

Conclusion

The purpose of the project was to develop a concept for a system, deployed within a customer's internal network, that could generate licenses on a licensing company's behalf. Moreover, a proof of concept was to be developed given certain requirements from Nevion and a security analysis was to be conducted.

The solution provides a concept which describes an Offline License Management system where the licensing company can send out digitally signed Top-Level Licenses to the Network Management System (NMS) deployed on the customer's internal network. The Network Management System can use the allotted time from the Top-Level Licenses to generate Sub-Level Licenses which are sent to the connected hardware-deployed LFAs to unlock functionality, while maintaining the integrity of the whole system.

Following this, the proof of concept was successfully developed and tested in a controlled environment, proving the feasibility of the proposed solution. However, a thorough security review revealed several vulnerabilities that need to be addressed before the system can be implemented and deployed in a production environment. The review assessed the identified threats and implemented prevention and mitigation measures to reduce the risk. Despite these measures, the review determined that the residual risk is still higher than the desired level of risk, defined through the risk appetite. Therefore, the concept cannot be considered viable from a strict security perspective, and further measures are required to reduce the risk to an acceptable level without affecting the usability of the system.

Despite the security challenges, the project achieved its main objective of developing a concept for a system that can generate licenses on a licensing company's behalf. Further development and refinement of the concept, along with addressing several key security concerns raised in Chapter 4 – Security Review, could lead to a more secure solution for an Offline License Management system.

7.1 Further Work

The development of the concept has created a basis for further development, although there are several weaknesses that can benefit from improvements. One of the main aspects is the secrets handling, which needs to be combined with the distribution model and the customer relationship management. According to the security review, the concept is not a fully secure solution to offline Digital Rights Management. Because of this, the security aspect will need further work before an instance of the concept can be developed for production environments.

Bibliography

- [1] Nevion Europe AS. 'About nevion.' (2023), [Online]. Available: <https://nevion.com/about/> (visited on 09/03/2023).
- [2] 'Soft patching,' TheProductionAcademy. (17th Sep. 2021), [Online]. Available: <https://www.theproductionacademy.com/blog/soft-patching> (visited on 06/05/2023).
- [3] M. E. Whitman and H. J. Mattord, *Principles of information security*, 4th ed. Mason, OH: CENGAGE Learning Custom Publishing, Jan. 2011.
- [4] C. Roach. 'Cryptography,' Data Insider. (4th Jan. 2023), [Online]. Available: <https://www.digitalguardian.com/blog/what-digital-rights-management> (visited on 28/04/2023).
- [5] '1219- farming simulator 22: Platinum edition (v1.9.0.0 + all dlcs + multi23) – [dodi repack],' DODI Repacks. (26th Mar. 2023), [Online]. Available: <https://dodi-repacks.site/1219-farming-simulator-22-v1-1-1-0-dlcs-online-multiplayer-multi23-dodi-repack/> (visited on 28/04/2023).
- [6] 'Software cracking,' Wikipedia. (31st Mar. 2023), [Online]. Available: https://en.wikipedia.org/wiki/Software_cracking (visited on 28/04/2023).
- [7] M. Taylor. 'A great day for drm as denuvo lapse renders tons of games temporarily unplayable,' PC Gamer. (8th Nov. 2021), [Online]. Available: <https://www.pcgamer.com/a-great-day-for-drm-as-denuvo-lapse-renders-tons-of-games-temporarily-unplayable/> (visited on 28/04/2023).
- [8] 'Listen offline,' Spotify. (), [Online]. Available: <https://support.spotify.com/no-en/article/listen-offline/> (visited on 28/04/2023).
- [9] 'Downloaded title says 'expired',' Netflix. (), [Online]. Available: <https://help.netflix.com/en/node/54865> (visited on 28/04/2023).
- [10] 'Network time protocol,' Wikipedia. (27th Apr. 2023), [Online]. Available: https://en.wikipedia.org/wiki/Network_Time_Protocol (visited on 28/04/2023).
- [11] 'Cryptography,' Wikipedia. (2nd May 2023), [Online]. Available: <https://en.wikipedia.org/wiki/Cryptography> (visited on 04/05/2023).

- [12] ‘Caesar cipher,’ Wikipedia. (22nd Apr. 2023), [Online]. Available: https://en.wikipedia.org/wiki/Caesar_cipher (visited on 03/05/2023).
- [13] C. Paar and J. Pelzl, *Understanding cryptography: a textbook for students and practitioners*. Springer-Verlag, 2009, ISBN: 9783642041013.
- [14] ‘Xor cipher,’ Wikipedia. (20th Mar. 2023), [Online]. Available: https://en.wikipedia.org/wiki/XOR_cipher (visited on 03/05/2023).
- [15] ‘Rijndael s-box,’ Wikipedia. (14th Sep. 2022), [Online]. Available: https://en.wikipedia.org/wiki/Rijndael_S-box (visited on 27/04/2023).
- [16] M. Dworkin, E. Barker, J. Nechvatal, J. Foti, L. Bassham, E. Roback and J. Dray, *Advanced encryption standard (aes)*, en, 26th Nov. 2001. DOI: <https://doi.org/10.6028/NIST.FIPS.197>. [Online]. Available: <https://www.nist.gov/publications/advanced-encryption-standard-aes> (visited on 01/05/2023).
- [17] ‘X.509,’ Wikipedia. (3rd May 2023), [Online]. Available: <https://en.wikipedia.org/wiki/X.509> (visited on 05/05/2023).
- [18] ‘Circular dependency,’ Wikipedia. (2nd Nov. 2022), [Online]. Available: https://en.wikipedia.org/wiki/Circular_dependency (visited on 01/05/2023).
- [19] ‘Embedded software,’ Wikipedia. (20th Apr. 2023), [Online]. Available: https://en.wikipedia.org/wiki/Embedded_software (visited on 16/05/2023).
- [20] C. P. Pfleeger, S. L. Pfleeger and J. Margulies, *Security in Computing*, 5th ed. Philadelphia, PA: Prentice Hall, Jan. 2015.
- [21] H. Bergsjø, R. Windvik and L. Øverlier, *Digital sikkerhet: en innføring*, Norwegian. 2020.
- [22] EC-Council Cybersecurity Exchange. ‘Dread threat modeling: An introduction to qualitative risk analysis.’ (2022), [Online]. Available: <https://eccouncil.org/cybersecurity-exchange/threat-intelligence/dread-threat-modeling-intro/> (visited on 24/03/2023).
- [23] A. de Ruijter and F. Guldenmund, ‘The bowtie method: A review,’ *Safety Science*, vol. 88, pp. 211–218, 2016, ISSN: 0925-7535. DOI: 10.1016/j.ssci.2016.03.001.
- [24] ‘Stride (security),’ Wikipedia. (20th Feb. 2023), [Online]. Available: [https://en.wikipedia.org/wiki/STRIDE_\(security\)](https://en.wikipedia.org/wiki/STRIDE_(security)) (visited on 06/04/2023).
- [25] M. Paul, *Official (ISC)² Guide to the CSSLP CBK* (ISC2 Press), 2nd ed. Philadelphia, PA: Auerbach, Aug. 2013.
- [26] NTNU. ‘Conducting risk assessments.’ Norwegian. (), [Online]. Available: <https://i.ntnu.no/wiki/-/wiki/English/Conducting+risk+assessments> (visited on 12/04/2023).

- [27] *Information security, cybersecurity and privacy protection – Guidance on managing information security risks*, ISO/IEC 27005:2022. Vernier, Geneva, Switzerland: International Organization for Standardization, Oct. 2021. [Online]. Available: <https://www.iso.org/standard/80585.html>.
- [28] NTNU. 'Informasjonsklassifisering - informasjonssikkerhet.' Norwegian. (), [Online]. Available: <https://i.ntnu.no/wiki/-/wiki/Norsk/Informasjonsklassifisering+-informasjonssikkerhet> (visited on 28/03/2023).
- [29] 'Zip bomb,' Wikipedia. (2nd May 2023), [Online]. Available: https://en.wikipedia.org/wiki/Zip_bomb (visited on 16/05/2023).
- [30] 'Openssl,' Wikipedia. (15th Mar. 2023), [Online]. Available: <https://en.wikipedia.org/wiki/OpenSSL> (visited on 15/03/2023).
- [31] 'Getting started with java,' Oracle Java. (), [Online]. Available: <https://dev.java/learn/getting-started/> (visited on 24/04/2023).
- [32] 'Apache maven features,' Apache. (), [Online]. Available: <https://maven.apache.org/maven-features.html> (visited on 14/04/2023).
- [33] 'Building rest services with spring,' VMware Tanzu. (), [Online]. Available: <https://spring.io/guides/tutorials/rest/> (visited on 14/04/2023).
- [34] 'Vue.js,' Vuejs. (), [Online]. Available: <https://vuejs.org/> (visited on 24/04/2023).
- [35] 'Vue component framework,' Vuetify. (), [Online]. Available: <https://v2.vuetifyjs.com/en/>.
- [36] 'Axios,' Axios. (), [Online]. Available: <https://axios-http.com/> (visited on 24/04/2023).
- [37] 'Npm.' (), [Online]. Available: <https://www.npmjs.com/> (visited on 26/04/2023).
- [38] 'Nodejs documentation.' (), [Online]. Available: <https://nodejs.org/en/docs> (visited on 26/04/2023).
- [39] 'Security concerns with using third-party dependencies.' (), [Online]. Available: <https://ottofeller.com/blog/security-concerns-with-using-third-party-dependencies> (visited on 25/04/2023).
- [40] 'What is a rest api?' Red Hat. (8th Jul. 2020), [Online]. Available: <https://www.redhat.com/en/topics/api/what-is-a-rest-api> (visited on 14/04/2023).
- [41] 'What is a restful api?' MuleSoft. (), [Online]. Available: <https://www.mulesoft.com/resources/api/restful-api> (visited on 24/04/2023).
- [42] 'Http response status codes,' Mozilla Developer. (10th Apr. 2023), [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Status> (visited on 18/04/2023).
- [43] R. T. Fielding, M. Nottingham and J. Reschke, *HTTP Semantics*, RFC 9110, Jun. 2022. DOI: 10.17487/RFC9110. [Online]. Available: <https://www.rfc-editor.org/info/rfc9110>.

- [44] ‘Apache httpcomponents httpclient overview,’ The Apache Software Foundation. (), [Online]. Available: <https://hc.apache.org/httpcomponents-client-5.2.x/> (visited on 21/04/2023).
- [45] ‘Java jdk 11: Log level,’ Oracle. (), [Online]. Available: <https://docs.oracle.com/en/java/javase/11/docs/api/java.logging/java/util/logging/Level.html> (visited on 03/05/2023).
- [46] ‘Man-in-the-middle attacks,’ Wikipedia. (31st Mar. 2023), [Online]. Available: https://en.wikipedia.org/wiki/Man-in-the-middle_attack (visited on 27/04/2023).
- [47] ‘Too many cooks may worsen the openssl mess.’ (), [Online]. Available: <https://www.infoworld.com/article/2608410/too-many-cooks-may-worsen-the-openssl-mess.html> (visited on 27/04/2023).
- [48] ‘Get the software.’ (), [Online]. Available: <https://cmake.org/download/> (visited on 25/04/2023).
- [49] ‘Bourne shell,’ Wikipedia. (17th Feb. 2023), [Online]. Available: https://en.wikipedia.org/wiki/Bourne_shell (visited on 26/04/2023).
- [50] ‘Downloading and installing node.js and npm,’ NPMJS. (), [Online]. Available: <https://docs.npmjs.com/downloading-and-installing-node-js-and-npm> (visited on 26/04/2023).
- [51] ‘Github actions documentation,’ GitHub. (), [Online]. Available: <https://docs.github.com/en/actions> (visited on 25/04/2023).
- [52] ‘About checkstyle,’ Checkstyle. (), [Online]. Available: <https://checkstyle.sourceforge.io/index.html> (visited on 04/05/2023).
- [53] ‘5 best practices for secrets management,’ HashiCorp. (14th Mar. 2023), [Online]. Available: <https://www.hashicorp.com/resources/5-best-practices-for-secrets-management> (visited on 10/05/2023).
- [54] ‘GraphQL,’ GraphQL Foundation. (), [Online]. Available: <https://graphql.org/> (visited on 10/05/2023).
- [55] ‘Websocket api,’ Mozilla Developer. (), [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets%5C_API (visited on 10/05/2023).
- [56] ‘GraphQL,’ Wikipedia. (9th May 2023), [Online]. Available: <https://en.wikipedia.org/wiki/GraphQL> (visited on 10/05/2023).
- [57] ‘Websocket,’ Wikipedia. (8th Apr. 2021), [Online]. Available: <https://en.wikipedia.org/wiki/WebSocket> (visited on 10/05/2023).
- [58] K. Annoh. ‘Http vs https – what’s the difference?’ FreeCodeCamp. (16th Aug. 2022), [Online]. Available: <https://www.freecodecamp.org/news/http-vs-https/> (visited on 26/04/2023).

- [59] L. Mui. 'Wyntk: Unix system administrator by linda mui,' O'Reilly Media, Inc. (), [Online]. Available: <https://www.oreilly.com/library/view/wyntk-unix-system/1565921046/ch03s12.html> (visited on 26/04/2023).
- [60] 'The impact of data centers on global carbon emissions & how removing rot data can help reduce it,' NowVertical. (11th Sep. 2022), [Online]. Available: <https://www.nowvertical.com/insights/impact-of-data-centers-on-global-emissions/> (visited on 05/05/2023).
- [61] B. Vuleta. '23 corrupting piracy statistics you must know in 2023,' LegalJobs. (30th Mar. 2023), [Online]. Available: <https://legaljobs.io/blog/piracy-statistics/> (visited on 05/05/2023).

Appendix A

Project Agreement

Fastsatt av prorektor for utdanning 10.12.2020

STANDARDAVTALE

om utføring av studentoppgave i samarbeid med ekstern virksomhet

Avtalen er ufravikelig for studentoppgaver (heretter oppgave) ved NTNU som utføres i samarbeid med ekstern virksomhet.

Forklaring av begrep

Opphavsrett

Er den rett som den som skaper et åndsverk har til å fremstille eksemplarer av åndsverket og gjøre det tilgjengelig for allmennheten. Et åndsverk kan være et litterært, vitenskapelig eller kunstnerisk verk. En studentoppgave vil være et åndsverk.

Eiendomsrett til resultater

Betyr at den som eier resultatene bestemmer over disse. Utgangspunktet er at studenten eier resultatene fra sitt studentarbeid. Studenten kan også overføre eiendomsretten til den eksterne virksomheten.

Bruksrett til resultater

Den som eier resultatene kan gi andre en rett til å bruke resultatene, f.eks. at studenten gir NTNU og den eksterne virksomheten rett til å bruke resultatene fra studentoppgaven i deres virksomhet.

Prosjektbakgrunn

Det partene i avtalen har med seg inn i prosjektet, dvs. som vedkommende eier eller har rettigheter til fra før og som brukes i det videre arbeidet med studentoppgaven. Dette kan også være materiale som tredjepersoner (som ikke er part i avtalen) har rettigheter til.

Utsatt offentliggjøring

Betyr at oppgaven ikke blir tilgjengelig for allmennheten før etter en viss tid, f.eks. før etter tre år. Da vil det kun være veileder ved NTNU, sensorene og den eksterne virksomheten som har tilgang til studentarbeidet de tre første årene etter at studentarbeidet er innlevert.

1. Avtaleparter

Norges teknisk-naturvitenskapelige universitet (NTNU) Institutt:
Veileder ved NTNU: Steven Yves Le Moan e-post: steven.lemoan@ntnu.no
Ekstern virksomhet: Nevion AS Ekstern virksomhet sin kontaktperson og e-post: Anders Dale, adale@nevion.com
Student: Ivan Norderhaug Fødselsdato: 04/02/2000
Student: Sander Osvik Brekke Fødselsdato: 03/04/2001
Student: Kristian Røren Svanholm Fødselsdato: 23/05/2000

Partene har ansvar for å klarere eventuelle immaterielle rettigheter som studenten, NTNU, den eksterne eller tredjeperson (som ikke er part i avtalen) har til prosjektbakgrunn før bruk i forbindelse med utførelse av oppgaven. Eierskap til prosjektbakgrunn skal fremgå av eget vedlegg til avtalen der dette kan ha betydning for utførelse av oppgaven.

2. Utførelse av oppgave

Studenten skal utføre: (sett kryss)

Masteroppgave	
Bacheloroppgave	X
Prosjektoppgave	
Annen oppgave	

Startdato: 09/01/2022
Sluttdato: 08/06/2022

Oppgavens arbeidstittel er: External generation of software licenses using modern cryptographic solutions
--

Ansvarlig veileder ved NTNU har det overordnede faglige ansvaret for utforming og godkjenning av prosjektbeskrivelse og studentens læring.

3. Ekstern virksomhet sine plikter

Ekstern virksomhet skal stille med en kontaktperson som har nødvendig faglig kompetanse til å gi studenten tilstrekkelig veiledning i samarbeid med veileder ved NTNU. Ekstern kontaktperson fremgår i punkt 1.

Formålet med oppgaven er studentarbeid. Oppgaven utføres som ledd i studiet. Studenten skal ikke motta lønn eller lignende godtgjørelse fra den eksterne for studentarbeidet. Utgifter knyttet til gjennomføring av oppgaven skal dekkes av den eksterne. Aktuelle utgifter kan for eksempel være reiser, materialer for bygging av prototyp, innkjøp av prøver, tester på lab, kjemikalier. Studenten skal klarere dekning av utgifter med ekstern virksomhet på forhånd.

Ekstern virksomhet skal dekke følgende utgifter til utførelse av oppgaven:

- Nødvendige reiseutgifter tilknyttet arbeidet med oppgaven, deriblant fysiske møter ved kontorene til Nevion AS.
- Utgifter til kost, deriblant ved reise, der dette viser seg nødvendig.
- Nødvendige utgifter til programvare, lisenser og liknende verktøy, etter godkjenning.

Dekning av utgifter til annet enn det som er oppført her avgjøres av den eksterne underveis i arbeidet.

4. Studentens rettigheter

Studenten har opphavsrett til oppgaven¹. Alle resultater av oppgaven, skapt av studenten alene gjennom arbeidet med oppgaven, eies av studenten med de begrensninger som følger av punkt 5, 6 og 7 nedenfor. Eiendomsretten til resultatene overføres til ekstern virksomhet hvis punkt 5 b er avkrysset eller for tilfelle som i punkt 6 (overføring ved patenterbare oppfinnelser).

I henhold til lov om opphavsrett til åndsverk beholder alltid studenten de ideelle rettigheter til eget åndsverk, dvs. retten til navngivelse og vern mot krenkende bruk.

Studenten har rett til å inngå egen avtale med NTNU om publisering av sin oppgave i NTNUs institusjonelle arkiv på Internett (NTNU Open). Studenten har også rett til å publisere oppgaven eller deler av den i andre sammenhenger dersom det ikke i denne avtalen er avtalt begrensninger i adgangen til å publisere, jf. punkt 8.

¹ Jf. Lov om opphavsrett til åndsverk mv. av 15.06.2018 § 1

5. Den eksterne virksomheten sine rettigheter

Der oppgaven bygger på, eller videreutvikler materiale og/eller metoder (prosjektbakgrunn) som eies av den eksterne, eies prosjektbakgrunnen fortsatt av den eksterne. Hvis studenten skal utnytte resultater som inkluderer den eksterne sin prosjektbakgrunn, forutsetter dette at det er inngått egen avtale om dette mellom studenten og den eksterne virksomheten.

Alternativ a) (sett kryss) Hovedregel

<input type="checkbox"/>	Ekstern virksomhet skal ha bruksrett til resultatene av oppgaven
--------------------------	--

Dette innebærer at ekstern virksomhet skal ha rett til å benytte resultatene av oppgaven i egen virksomhet. Retten er ikke-eksklusiv.

Alternativ b) (sett kryss) Unntak

<input checked="" type="checkbox"/>	Ekstern virksomhet skal ha eiendomsretten til resultatene av oppgaven og studentens bidrag i ekstern virksomhet sitt prosjekt
-------------------------------------	---

Begrunnelse for at ekstern virksomhet har behov for å få overført eiendomsrett til resultatene:

Ønsker bruk av kode og rapport i kommersiell sammenheng og videreutvikling av denne.

6. Godtgjøring ved patenterbare oppfinnelser

Dersom studenten i forbindelse med utførelsen av oppgaven har nådd frem til en patenterbar oppfinnelse, enten alene eller sammen med andre, kan den eksterne kreve retten til oppfinnelsen overført til seg. Dette forutsetter at utnyttelsen av oppfinnelsen faller inn under den eksterne sitt virksomhetsområde. I så fall har studenten krav på rimelig godtgjøring. Godtgjøringen skal fastsettes i samsvar med arbeidstakeroppfinnelsesloven § 7. Fristbestemmelsene i § 7 gis tilsvarende anvendelse.

7. NTNU sine rettigheter

De innleverte filer av oppgaven med vedlegg, som er nødvendig for sensur og arkivering ved NTNU, tilhører NTNU. NTNU får en vederlagsfri bruksrett til resultatene av oppgaven, inkludert vedlegg til denne, og kan benytte dette til undervisnings- og forskningsformål med de eventuelle begrensninger som fremgår i punkt 8.

8. Utsatt offentliggjøring

Hovedregelen er at studentoppgaver skal være offentlige.

Sett kryss

<input checked="" type="checkbox"/>	Oppgaven skal være offentlig
-------------------------------------	------------------------------

I særlige tilfeller kan partene bli enige om at hele eller deler av oppgaven skal være undergitt utsatt offentliggjøring i maksimalt tre år. Hvis oppgaven unntas fra offentliggjøring, vil den kun være tilgjengelig for student, ekstern virksomhet og veileder i denne perioden. Sensurkomiteen vil ha tilgang til oppgaven i forbindelse med sensur. Student, veileder og sensorer har taushetsplikt om innhold som er unntatt offentliggjøring.

Oppgaven skal være underlagt utsatt offentliggjøring i (sett kryss hvis dette er aktuelt):

Sett kryss

Sett dato

Sett kryss	Sett dato
<input type="checkbox"/>	ett år
<input type="checkbox"/>	to år
<input type="checkbox"/>	tre år

Behovet for utsatt offentliggjøring er begrunnet ut fra følgende:

Dersom partene, etter at oppgaven er ferdig, blir enig om at det ikke er behov for utsatt offentliggjøring, kan dette endres. I så fall skal dette avtales skriftlig.

Vedlegg til oppgaven kan unntas ut over tre år etter forespørsel fra ekstern virksomhet. NTNU (ved instituttet) og student skal godta dette hvis den eksterne har saklig grunn for å be om at et eller flere vedlegg unntas. Ekstern virksomhet må sende forespørsel før oppgaven leveres.

De delene av oppgaven som ikke er undergitt utsatt offentliggjøring, kan publiseres i NTNUs institusjonelle arkiv, jf. punkt 4, siste avsnitt. Selv om oppgaven er undergitt utsatt offentliggjøring, skal ekstern virksomhet legge til rette for at studenten kan benytte hele eller deler av oppgaven i forbindelse med jobbsøknader samt videreføring i et master- eller doktorgradsarbeid.

9. Generelt



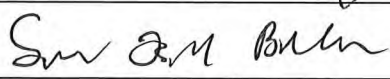
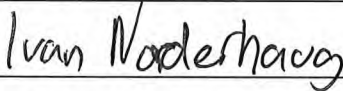
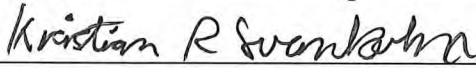
Denne avtalen skal ha gyldighet foran andre avtaler som er eller blir opprettet mellom to av partene som er nevnt ovenfor. Dersom student og ekstern virksomhet skal inngå avtale om konfidensialitet om det som studenten får kjennskap til i eller gjennom den eksterne virksomheten, kan NTNUs standardmal for konfidensialitetsavtale benyttes.

Den eksterne sin egen konfidensialitetsavtale, eventuell konfidensialitetsavtale den eksterne har inngått i samarbeidprosjekter, kan også brukes forutsatt at den ikke inneholder punkter i motstrid med denne avtalen (om rettigheter, offentliggjøring mm). Dersom det likevel viser seg at det er motstrid, skal NTNUs standardavtale om utføring av studentoppgave gå foran. Eventuell avtale om konfidensialitet skal vedlegges denne avtalen.

Eventuell uenighet som følge av denne avtalen skal søkes løst ved forhandlinger. Hvis dette ikke fører frem, er partene enige om at tvisten avgjøres ved voldgift i henhold til norsk lov. Tvisten avgjøres av sorenskriveren ved Sør-Trøndelag tingrett eller den han/hun oppnevner.

Denne avtale er signert i fire eksemplarer hvor partene skal ha hvert sitt eksemplar. Avtalen er gyldig når den er underskrevet av NTNU v/instituttleder.

Signaturer:

Instituttleder: Dato:	
Veileder ved NTNU: Steven Yves Le Moan Dato: 17.01.23	
Ekstern virksomhet: Nevion AS v/ Anders Dale Dato: 13/1-23	
Student: Sander Osvik Brekke Dato: 13/1-23	
Student: Ivan Norderhaug Dato: 13/1-23	
Student: Kristian Røren Svanholm Dato: 13/1-23	

Appendix B

Task Description

External generation of software licenses using modern cryptographic solutions

Nevion is a leader in the media industry providing award-winning media transport solutions to broadcasters, telecom service providers, governmental agencies and other industries.

One of Nevions solutions is a hardware platform with several features that may be unlocked by buying a software license file. These licenses may last for a limited period, but typically has infinite duration.

Although this suits many customers, some customers aren't using the equipment 24/7, and would like to pay only when they are using a specific feature. One example of this is equipment located at sport venues, which is only used during matches.

Nevion is also a provider of a software network management system (NMS) that has access to all the units in the customers' network.

We would like some students to find a solution for how our NMS, which is deployed in the customers internal network, safely could generate licenses on Nevion's behalf.

One approach to solving this could be to use the same functionality the HTTPS protocol uses to validate the authenticity of a host. This is done using a Chain of Trust using X.509 certificates and modern cryptographic solutions. It should be possible to reuse the same concept, thus allowing a piece of software in the NMS deployed at customer site to create and sign software licenses on Nevions behalf using an intermediate certificate signed by Nevions root certificate. The public part of the root certificate will be embedded in the hardware platform, allowing both the NMS generated license file, and a license file issued directly from Nevion to be approved in the device.

Nevion will issue software licenses to the customers NMS for a given feature for X number of minutes. This goes into a common pool, and the NMS can freely use these minutes among devices

Nevion would like a report on the security of such a system; Is it secure? Any weak points to consider? How to handle weak points? How to handle certificates? Revoking certificates? Revoking licenses? How to keep track of license usage?

Nevion would also like a PoC system implemented. This includes the software to generate licenses (preferably c++), software to run within the NMS (preferably Java or Scala code), and software to authenticate license files (preferably c++).

In addition to getting experience in programming, this assignment will give the students a good understanding of encryption using asymmetric encryption algorithms, and modern authentication using digital certificates. This is increasingly relevant as the digital world is adopting secure practices.

Assignment reserved for: Sander Osvik Brekke, Ivan Norderhaug and Kristian Røren Svanholm

Contact info:

Anders Dale, Senior Software Architect

adale@nevision.com, +47 4819925

Nevion AS, Nordre Kullerød 1, 3241 Sandefjord

Appendix C

Project Plan

Client Side Generation of Software Licenses Using Modern Cryptographic Solutions

—

Project Plan

Sander Osvik Brekke

Ivan Norderhaug

Kristian Røren Svanholm

January 2023

Contents

List of Tables	iii
Acronyms	iv
1 Goals and frameworks	1
1.1 Background	1
1.2 Project goals	1
1.3 Constraints	2
2 Scope	3
2.1 Knowledge Domain	3
2.2 Delimitation	3
2.3 Task description	4
3 Project Organization	5
3.1 Roles and Responsibilities	5
3.1.1 Group Structure and Hierarchy	5
3.1.2 Development and Research Roles and Responsibilities	5
3.1.3 Documentation Roles and Responsibilities	6
3.1.4 Process Framework Roles and Responsibilities	6
3.1.5 Summary	7
3.2 Routines and Group Rules	7
4 Planning, follow-up and reporting	8
4.1 Process Framework	8
4.2 Meeting Plan	9
5 Quality Assurance	10
5.1 Documentation, tools and standards	10

5.1.1	Documentation	10
5.1.2	Tools	10
5.1.3	Standards	10
5.1.4	Git Workflow	11
5.2	Test Plan	12
5.3	Risk analysis	13
5.4	Inspection	15
6	Progress Plan	16
6.1	Gantt	16
6.2	Activities	16
6.3	Milestones	16
A	Gantt Diagram	18

List of Tables

1	Responsibilities and roles summary	7
2	Risk Analysis Table	13
3	Risk Matrix	14
4	Remedy Table	14
5	Project milestones	16

Acronyms

PoC Proof of Concept.

NMS Network management system.

PKI Public key infrastructure.

MVP Minimum viable product

1 Goals and frameworks

1.1 Background

Nevion is a leader in the media industry providing award-winning media transport solutions to broadcasters, telecom service providers, governmental agencies and other industries.

One of their solutions is a hardware platform with several features that may be unlocked by buying a software license file. These licenses may last for a limited period, but typically has infinite duration. Although this suits many customers, some customers aren't using the equipment 24/7, and would like to pay only when they are using a specific feature. One example of this is equipment located at sport venues, which is only used during matches.

They are also a provider of a software NMS that has access to all the units in the customers' network and would like us to find a solution for how their NMS, which is deployed in their customers internal network off the internet, safely could generate licenses on their behalf. Our delivery will be in two parts: Firstly a security review regarding the safety of such a system in its environment and secondly a proof of concept Nevion can re-implement into their own product line.

1.2 Project goals

Goal	Description
Effect-oriented	Nevion will receive a concept on how to implement a solution for providing time limited licenses for their customers
	Nevion will receive a security review enabling them to comprehend and act on the risks related to implementing the concept in their own systems.
Results-oriented	A security review of the system and the implications of its environment.
	A proof of concept cryptography solution to the problem consisting of three standalone pieces of software. <ul style="list-style-type: none">• Piece of Software to generate license files• Piece of (preferably C++) Software to authenticate license files• Piece of (preferably Java) Software to operate within the NMS
Learning outcomes	Gain a deeper understanding of cryptography and chain of trust concepts
	Obtain real life experience from a real life industry project.

1.3 Constraints

The project constraints are as follows:

Time constraints

- The project will be delivered the 22nd of May, 2023.
- The project plan and thesis-contract will be delivered the 31st of January, 2023

Economic constraints

- NTNU will not cover any expenses for the project.
- Nevia will cover necessary expenses as outlined in the thesis-contract.

Code constraints

- Authentication of license files will be coded in C++
- The management software will be coded in Java/Scala

Other constraints

- All documentation in the final delivery shall be written in English.
- The project description is locked and is not flexible for expansion during the project period.
- Our knowledge domain is limited in the sense that we cannot recruit anyone to help during the project period. However, given our experience in learning concepts in short periods of time from nearly three years of attending NTNU, we will be able to expand our total knowledge domain somewhat during the project period

2 Scope

The project aims to provide a secure solution for the client's NMS to generate licenses while being deployed in customers' internal networks, which are not connected to the internet.

2.1 Knowledge Domain

The project falls under the domain of network security and licensing management. The proposed solution involves generating licenses for the client's NMS securely while being deployed in customers' internal networks, which are not connected to the internet. To achieve this, the project will need to consider the following key concepts and technologies:

- Network security** This includes the various security measures and protocols that are used to protect networks and the data that is transmitted over them.
- PKI** This is a security infrastructure that is used to manage digital certificates and public-private key pairs. PKI is commonly used to secure communications over the internet and to authenticate users and devices.
- License file management** This involves managing licenses for software and hardware that is used on a network. This may include tracking the number of licenses that are in use, enforcing license compliance, and generating new licenses.

2.2 Delimitation

The following delimitations will act as a boundary for what is relevant and not for this project. As mentioned in the constraints (section 1.3), this project statement is locked to these delimitations and will not be altered or expanded at any time.

- The project will not cover integration of PoC into Nevions codebase.
- The implementation of PoC will not be designed for direct insertion into Nevions codebase beyond using the corresponding programming languages. The PoC will function as inspiration and a demonstration of the concept in practice.
- The project does not include developing a user interface for PoC outside internal tools made for interfacing between the three different pieces of software.
- The PoC will only utilize conventional security measures for communication between different aspects of the system such as TLS.
- The security review will only cover our own system in a vacuum and will not consider other aspects of Nevions system.
- The security review will not be an isolated product of its own, but a part of the thesis.

2.3 Task description

Our task is to deliver a security review of a "chain of trust"-solution for generating genuine software license files on an external closed network, disconnected from Nevia HQ. We are also tasked with creating our own proof of concept for this solution that Nevia may integrate into their own product line or take inspiration from if they so wish.

The security review will require the following:

- Identifying the main assets of the system.
- Identifying the main actors, roles and use-cases within the system.
- Identifying potential threat actors and abuse-cases.
- Performing a risk assessment using the DREAD model [1].
- Create a threat model using the STRIDE model [2].
- Document different remedies for potential threats.

The proof of concept will require the following:

- An entirely local solution. (Standard within the field of broadcasting)
- One piece of software for generating new license files for different functionality.
- One piece of software for authenticating license files. Written in C++.
- One piece of software to orchestrate and manage all the license files in the system with different time-pools for different functionality.

3 Project Organization

3.1 Roles and Responsibilities

For a team working on a bachelor thesis, some internal distribution of roles and responsibilities is necessary, as a part of making sure workload is equally distributed and that the responsibilities are clear from the beginning.

3.1.1 Group Structure and Hierarchy

The group is aiming for the internal group structure to be as flat as possible. This means that there will be no-one in the group who has a higher degree of decision making power than the others, except when the group rules and routines states it. This is because a fully flat structure is impractical, for example, in the event of an internal conflict. For this reason, a group leader is necessary.

The group leader is Sander Osvik Brekke.

The group leader role and the related group rules are further explained under section 3.2.

3.1.2 Development and Research Roles and Responsibilities

Cryptography Concept Responsible

The largest part of the project is developing a concept using cryptography. The development of this concept is the main responsibility of the cryptography concept responsible. This responsibility includes, but is not restricted to, making sure the necessary progress is present, making sure the concept follows the given requirements, making sure the necessary research is performed and making sure the tasks this development requires are delegated within the team.

The concept responsible is Ivan Norderhaug.

Proof of Concept

A part of the project description, thus also the thesis, is an implemented proof of concept, abbreviated to PoC. The PoC responsible has the responsibilities of making sure the PoC is developed according to requirements and making sure the tasks necessary to make the PoC complete are being delegated within the team.

The PoC responsible is Kristian Røren Svanholm.

Underneath the PoC responsibility role, there are developers. Different parts of the PoC is written in two different languages, Java and C++, where the team members have different knowledge in the two programming languages. As a result of this, team members will have different development roles.

Development roles

<i>Kristian</i>	C++ developer
<i>Sander</i>	Java developer
<i>Ivan</i>	Java developer

3.1.3 Documentation Roles and Responsibilities

Meeting Minutes

During the course of the project, there will be a number of meetings taking place. To make sure all these meetings have their corresponding meeting minutes, the group will have a meeting minutes responsible. The responsibilities consist of writing the meeting minutes, or delegating this responsibility when necessary, and making sure the meeting minutes are stored in a compliant way.

The meeting minutes responsible is Ivan Norderhaug.

Other Documentation

While working with the bachelor thesis, there is a large amount of documentation required. The documentation responsible is responsible of making sure the correct documentation is being created according to the requirements, within the related deadlines, and that the created documentation is being stored in a compliant way. The documentation responsible is also responsible of making sure the required hand-ins are being handed in within the set deadlines, even though it is the group's common responsibility to make sure the products and material needed to hand in is done within the set deadlines.

The documentation responsible is Sander Osvik Brekke.

3.1.4 Process Framework Roles and Responsibilities

The process framework chosen for this bachelor project is Scrum. For the process framework Scrum to work optimal, a Scrum master is necessary. This is a responsibility that, amongst other things, consist of leading the scrum meetings, both the sprint planning, sprint review and retrospective.

The Scrum master is Kristian Røren Svanholm.

3.1.5 Summary

A summary of the roles and responsibilities that has been delegated can be seen in table 1.

Student	Responsibilities
Sander Osvik Brekke	Group leader Java developer Main contact person for Nevion Documentation responsible
Ivan Norderhaug	Java developer Concept development responsible Meeting minutes responsible
Kristian Røren Svanholm	C++ developer PoC responsible Scrum master

Table 1: Responsibilities and roles summary

3.2 Routines and Group Rules

For the cooperation and group dynamic to work, it is necessary to have clear routines for when incidents occur and rules that team members need to follow.

A complete list of the group rules is attached to the final bachelor thesis, signed by the group members. The group rules does also contain an ordered list of actions followed at the incident of a broken rule.

4 Planning, follow-up and reporting

4.1 Process Framework

We've chosen to utilize Scrum as our Process Framework for the project duration to organize our work. This allows us to create smaller objectives in the form of sprints over one-week periods. During a sprint, we stay on track and prioritize important tasks. If an important issue arises, it is added to the backlog for future sprint planning. We will not be holding "official" daily standup meetings, but will keep a flowing conversation daily during our work hours to keep everyone up to date.

Backlog prioritization

We will be prioritizing different issues in the backlog for new sprints based on a few of their qualities and the state of the project. We prioritize our backlog items by assessing their impact on the project's objectives and the dependencies they have with other items. We will also consider the urgency of the item and its complexity and effort required to complete it. As an example, new functionalities have higher priority than issues related to minor bugs. Given a rather large bug this changes and the bug takes priority. A very generalized list of priorities can be found below. The order of priority may be looked away from given imminent deadlines

1. Project-hindering bugs
2. Mission critical rework of old works
3. New functionality / New documentation
4. Minor bug
5. Refinement of old works

Kanban Columns

To Do	Issue planned for this sprint, not started yet.
Working on it	Someone has started work on the issue.
Help needed	Assignee is stuck and either requires help or someone to take over. Should be comment explaining problem on the issue.
Review	Issue is ready for another team member to look over the work and OK/Deny
Done	Issue is done

More columns might be added in the future if we discover a need for a more granular approach.

General rules for working with the Kanban board

- Always assign yourself to your issues once you move them into "Working on it".
- It is allowed to assign yourself to an issue preemptively, but if the issue stagnates in "To Do" anyone else with time can take it.
- Do not delay moving an issue to its next logical column.
- An item can be moved back to "Working on it" if needed after review.
- The only way to "Done" is through "Review".
- If you get stuck and require assistance move your issue to "Help needed" and write a comment about what you are stuck on. If the next person entirely takes over the issue, they can assign themselves to it to keep clarity in the "Working on it" column.

These rules are in place to avoid double work, misunderstandings and low quality work.

4.2 Meeting Plan

Communication about change of plans for meetings with people outside the main group will take place over eMail with a corresponding calendar event being cancelled/updated in the group calendar. Internal changes will be communicated either verbally or over our Discord server.

Sprint meetings

For the sprints we will hold three different kinds of meetings at slightly different intervals. They will be held mainly physically, but potentially digitally on the Discord server.

- Sprint review is each Friday 09.00 in the morning.
- Sprint planning meeting is each Friday after the previous sprints' review meeting.
- Sprint retrospective is every third Friday after the first two meetings.

Nevion meetings

We will meet with our contact at Nevion every other week at Thursdays 11.00 to 12.00. This will likely take less than an hour, but we've decided to keep it this way to ensure an open calendar in case a longer meeting is required. These meetings will take place mostly digitally over Teams but may take place physically as we hit important milestones throughout the project.

Thesis-counselor meetings

We will meet with our thesis-counselor every Wednesday at 10.30 to 11.30. Again this is likely larger than needed, but is kept at an hour for the same reason as above. These meetings will take place digitally over Teams.

5 Quality Assurance

5.1 Documentation, tools and standards

5.1.1 Documentation

We are going to create and maintain a comprehensive set of documentation throughout the project. As the client wishes to integrate our implementation into their existing products, this part will be of utmost importance. All documentation will be kept up-to-date and will reflect the state of the project. We will store our documentation in a central location so that it can be easily accessible to all team members. This location will be Notion.

5.1.2 Tools

GitHub Tool for version control of the codebase with integrated issue tracking system and built-in CI/CD pipeline that can be used to build, test, and deploy our code.

Notion Tool for note-taking and organizing project-related information.

Jira Tool for Scrum management.

Toggl A time tracking tool that will be used to track the time spent on different tasks and activities, to help us manage our project schedule.

Overleaf A collaborative \LaTeX editor that will be used to write and edit our project's documentation, and also to prepare the final report and thesis.

5.1.3 Standards

As the source code will consist of multiple programming languages such as C++ and Java, it is essential that the code base is well structured and follows the best practices of each respective language to ensure efficient and optimized code. To achieve this we will be following the Java and C++ standards and use code linting tools to highlight potential problems with styling and syntax.

As the project is of considerable size and complexity, it is essential to follow a consistent and organized approach to version control. To ensure the quality and maintainability of the code base, we will adhere to the following guidelines for committing code:

1. The commit message should briefly describe the changes made in the commit. It should be clear and easy to understand.
2. If the commit is related to a specific issue or bug, include the issue number in the commit message.

5.1.4 Git Workflow

To manage the codebase effectively and efficiently, we will follow a Git workflow. This will ensure that the codebase is well organized and that changes can be tracked and reviewed easily. The following is an outline of the Git workflow we will follow:

- All development will take place on branches, with the main development branch being 'dev'.
- The 'main' branch will represent the current production version of the codebase.
- The codebase on the 'main' will be stable, well documented and tested.
- Each feature or bug fix will have its own branch, which will be created from the 'dev' branch and will be named after the feature or bug fix.
- When a feature or bug fix is complete, a merge request will be created to merge the feature or bug fix branch into the 'dev' branch.
- The merge request will be reviewed by at least one other member of the team before being merged.
- The 'dev' branch needs to be stable.
- The branch naming convention will be meaningful and in line with the task, bug or feature retrieved from Jira.

By following these guidelines, we can ensure that the codebase is well-organized and that changes can be easily tracked and reviewed. This will help ensure the quality and maintainability of the codebase.

5.2 Test Plan

We will use JUnit for Java and CppUnit for C++ to perform unit testing on individual code modules. This will help to ensure that each module is functioning as expected and identify any bugs throughout the development process.

These tests will be integrated with GitHub Actions (CI/CD). This will ensure that unit tests are automatically run with every code change, and any failing tests will prevent the code from being merged to the main branch.

In addition to unit testing, we will also be performing a thorough code review process, which is further explained under section 5.4. This will involve reviewing the code for any bugs, security vulnerabilities, or other issues. By identifying and addressing any issues early on, we can ensure that the final product is of high quality and meets the project's objectives.

We will not be conducting user testing or usability testing as the project does not involve end-user interactions. As for security testing, we will not be performing any penetration testing, but we will be performing a thorough code review process and security analysis to identify and address any potential vulnerabilities.

Appendix D

Gantt Diagram

Appendix E

Group Rules

Group rules

- A. Everyone is expected to attend all meetings. If someone is unable to attend, advance notice must be given within the day before the agreed meeting.
- B. Responsibilities must be distributed so that no one works completely alone on one aspect to ensure that illness/absence does not delay parts of the project more than necessary.
- C. Everyone is expected to complete a minimum of 30 hours of work per week per student. If this cannot be done, an average of around 30 hours per week is expected over the duration of the project.
- D. Everyone is expected to contribute with their share, approximately 1/3 of the total workload. If a group member's contribution is drastically lower than expected over a longer period, the group member risks being expelled from the group.
- E. Everyone is expected to follow common customs, be polite to other group members and act professionally both internally and externally.
- F. Everyone is expected to follow up and maintain their own obligations, roles, and responsibilities, as defined in the project plan.
- G. If unavoidable expenses arise, and they are not covered by the client, these must be shared equally between the group members.

In the event of a breach of the rules, the following measures are taken, in the given order:

1. The group tries to resolve the conflict internally through group meetings. In the event of disagreements, the group leader will try to mediate.
2. A written warning is given to the relevant group member. The warning must contain the reason for the warning and attempts at mediation. A copy of the written warning is sent to the supervisor. After 2 written warnings, further mediation is initiated.
3. The supervisor, Steven Yves Le Moan, is contacted, the situation is clarified, and the supervisor tries to mediate.
4. The course coordinator, Tom Røise, is contacted, the situation is clarified, and the course coordinator tries to mediate.
5. If disagreements and conflicts are not resolved, the group will be split up according to current procedures at NTNU.

Date and signatures:

Sander Emil Bohr

Ivan Nordstrem

Kristin R. Soreide

Appendix F

Status Report

Status Report

Group 2 -
Kristian Røren Svanhold
Sander Osvik Brekke
Ivan Norderhaug

March 2, 2023

Contents

1	Shorthands	1
2	Status rapport	2
2.1	Establishment	2
2.2	Technologies	2
2.2.1	Front-end	2
2.2.2	Back-end	2
2.2.3	Network Management System	2
2.2.4	License File Aggregator	2
2.2.5	Hosting	3
2.3	Core Functionality	3
2.4	Scrum	3
3	Architecture diagram	4

1 Shorthands

NMS - Network Management System
LFA - License File Aggregator

2 Status rapport

2.1 Establishment

During the last two months we've worked hard on developing our proof of concept and are nearing a stage in development where we feel we can finish within a few weeks from now (2.march 2023). We've performed all the development using scrum and following the group rules we defined at an earlier stage of the project. Our code is also up and running at NTNU's SkyHigh to simulate the real world environment it could be used in.

We've also logged all our working hours in Toggl where we per today (2. march 2023) have logged an average total of ~ 168 hours per person which averages out to around 24 hours of work per person per week. This is below our target but is explained by the group consisting of two individuals which have the INGG2300 course which have consumed an absurd amount of time out of their work week to no fault of their own.

2.2 Technologies

The different parts of this complex machinery uses a range of different technologies, including different languages. This section will go through the different technologies used in the different parts of the system.

2.2.1 Front-end

The front-end is a web site written with demonstration purposes, where JavaScript and Vue2 has been used as the technologies. Vuetify has also been used. This web site will be used to send API requests to the different APIs being served by the LFAs and the NMSs, displaying the current statuses of different aspects and to upload license files.

2.2.2 Back-end

The back-end consists of both the Network management systems and the license file aggregators. These serve their own web servers. This is done to ensure the ability to communicate between the different parts of the system.

2.2.3 Network Management System

The NMS is written in Java 11, using Maven as the package manager. In addition to this, Apache is being used as an HTTP Client. The RESTFUL API and the web server is made through Spring Web, and more specifically Spring Boot Start.

2.2.4 License File Aggregator

The LFA software is written in C++. In addition, OatPP is being used as a web server and a RESTFUL API.

2.2.5 Hosting

The proof of concept is hosted in the NTNU Internal infrastructure solution named "SkyHigh". There is being hosted one NMS and two LFAs, communicating on the local network. The hosting happens on 3 different Linux Debian servers on a closed local network.

2.3 Core Functionality

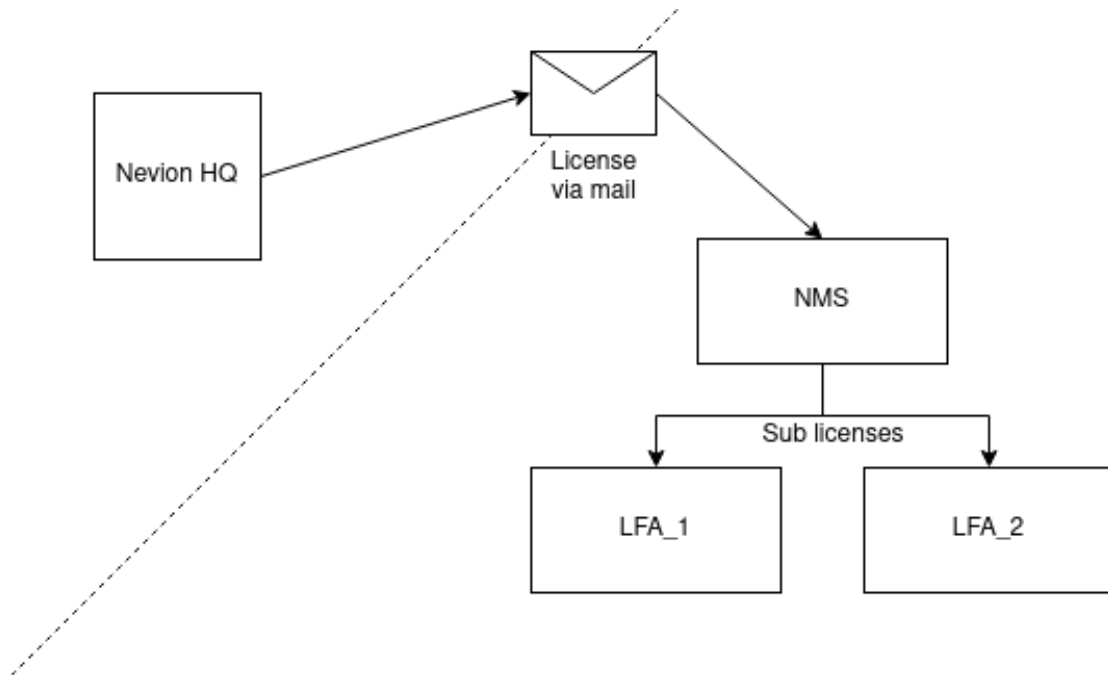
- The NMS can receive and parse licenses
- Licenses and license usage persist between instances
- The LFA can receive a sub license, verify it and then use it if all is ok.
- The NMS can GET all licenses available in the LFA and consume them if they have expired.
- As all data from the NMS is stored locally, which is a huge security concern. Everything of value is encrypted using a mathematically derived key.
- "Doomsday" mechanism is active on the NMS and if a vital file is modified or deleted, the program will not work. This is because integrity has been breached.
- We have implemented simple frontend to showcase the system.

2.4 Scrum

We decided early on to utilize the SCRUM framework and we feel we have been successful during the last two months in implementing the system. We perform weekly sprints with a corresponding weekly sprint review and sprint planning meeting for the next sprint. Asynchronously we hold tri-weekly sprint-retrospective meetings to take a wider look at how the last three sprints have gone.

Our chosen tool for managing SCRUM is Jira where we continuously update the backlog and work through our sprints within a Kanban board. A more thorough explanation can be found in the initial project plan.

3 Architecture diagram



Our architecture is quite simple consisting of two main parts. Our LFAs and our NMS. These two components exist together on different devices on its own closed network with no access to the wider internet which means that all Top-level Licenses are sent by email to an actor at the network location. This individual who will then proceed to manually upload the license to the NMS as seen in the above diagram.

Appendix G

Meeting Minutes

Meeting Minutes 20230109

👤 Attendees	① Ivan ③ Sander Osvik Brekke ④ Kristian
☰ External attendees	Anders Dale
📅 Event time	@January 9, 2023 10:00 AM → 12:00 PM
☰ Place	NTNU (T114)
⌵ Type	Client meeting
☰ Tag	

Topics

- Contract Work
- Development Tools
- Project strategy

Details

At this meeting, we initiated the planning process for our bachelor thesis by discussing and selecting effective tools for our project. We decided to utilize Joplin for taking notes and maintaining meeting minutes and ShareLatex, similar to Overleaf, for creating our main report. Additionally, we progressed with outlining the details of the agreement between our group and Nevion. We also discussed and selected scrum as our project strategy.

Actions:



@sanderob: Set up a Joplin server and a ShareLatex server.

Meeting Minutes 20230118

Attendees	
External attendees	
Event time	@January 18, 2023 10:30 AM
Place	Microsoft Teams
Type	Thesis supervisor meeting
Tag	

This meeting was cancelled by Steven Yves Le Moan with no description.

Meeting Minutes 20230120

👤 Attendees	① Ivan ② Kristian ③ Sander Osvik Brekke
☰ External attendees	Anders Dale
📅 Event time	@January 20, 2023 10:00 AM → 12:00 PM
☰ Place	Nevion
⌵ Type	Client meeting
☰ Tag	

Topics:

- Contract Writing
- Program Development
- Security Report
- Meeting Plan

Details:

Contract Writing:

- Nevion will own the rights to the project
- No concerns regarding confidentiality

Program Development:

- Separate from VIP/Virtuoso
- Preferably writing in C++ for authenticator & generator, and Java/Scala for VIP functionality
- TPM (Trusted Platform Module) to be discussed
- OpenSSL to be considered as an option

Security Report:

- To be incorporated into the project, not as a separate document
- Review whether or not the concept/concept implementation is secure or not. What holes would we need to cover?

Meeting Plan:

- Bi-weekly meetings with Nevion, weekly meetings with the supervisor
- Specific dates TBD.

Meeting Minutes 20230120

Attendees	① Ivan ② Kristian ③ Sander Osvik Brekke
External attendees	
Event time	@January 20, 2023 9:00 AM → 9:30 AM
Place	Discord
Type	Sprint meeting
Tag	

Topics:

- Sprint Review
- Sprint Planning

Details:

The initial sprint was successful, with all issues being completed before the next sprint. The team had a positive experience using this framework for the project.

The planning for the next sprint is on schedule and we will continue as outlined in the Project Plan's Gantt chart. Kristian suggests filling up the backlog as soon as possible.

During the meeting, we also talked about configuring development environments and determining the meaning of various repositories, this was added to the backlog and will be done during Sprint 2.

Meeting Minutes 20230125

Attendees	① Ivan ③ Sander Osvik Brekke ④ Kristian
External attendees	Steven Yves Le Moan
Event time	@January 25, 2023 10:30 AM → 11:00 AM
Place	Microsoft Teams
Type	Thesis supervisor meeting
Tag	

Topics:

- Project plan
- MVP
- Collaboration agreement

Details:

Steven approved of the project plan.

We have decided to create a secure as possible solution as it is not possible to make something 100% when it is on the premise of the threat actor. Steven agrees. (This might be updated in the future once we get more information from experts in the field)

A MVP of the proposed solution would be practical, and so does Steven.

We have not created a collaboration agreement. However, we have a group rules document which is signed and Steven approved it as a replacement for the collaboration agreement.

Meeting Minutes 20230126

Attendees	Ivan Sander Osvik Brekke Kristian
External attendees	Anders Dale
Event time	@January 26, 2023 11:00 AM → 12:00 PM
Place	Microsoft Teams
Type	Client meeting
Tag	

Topics

- Discussion regarding the current proposal
- Concept/Proof concept

Details

Unless there is "secure boot", then it is not possible to be 100% secure.

Virtuso holds the public certificate/public key of Nevion.

Garbage in, garbage out. Intermediate CA must be generated by Nevions private key

Intermediate CA will time out, will need to somehow renew the CA.

Pool function wont work if source code is altered.

Check if the pool is signed by the Intermediate CA on the Virtuso.

Maybe ignore the pool aspect

Is Hardware locked software the solution?

License generator does not need to be in C++. However, the NMS software and Virtuso software would be preferable to be in C++ and Java.

Licenses must be same format. Create out JSON file → Sign → Give to the target software

Prerequisites

- We can assume that NMS is safe, therefore this should be possible
- We can assume that the source code is safe

Meeting Minutes 20230127

Attendees	① Ivan ③ Sander Osvik Brekke ④ Kristian
External attendees	
Event time	@January 27, 2023 9:00 AM → 10:00 AM
Place	NTNU (A155)
Type	Sprint meeting
Tag	

Topics:

- Sprint Review
- Sprint Planning

Details:

The sprint was successful, with all issues being completed before the next sprint. The team still had a positive experience using this framework for the project.

The plan for the upcoming sprint is on track and we will follow the schedule outlined in the Project Plan's Gantt chart. However, this progress is only possible because we are not focusing on group work in another course. This approach is likely not sustainable and our progress may slow down temporarily while the other course is ongoing.

Meeting Minutes 20230130

Attendees	Ⓢ Sander Osvik Brekke Ⓛ Ivan Ⓚ Kristian
External attendees	Tjerand Silde
Event time	@January 30, 2023 4:15 PM → 5:00 PM
Place	Microsoft Teams
Type	Domain Expert Meeting
Tag	

Topics

- Ideas to license distribution inside network
- Ideas to securing the integrity of licenses

Details

- Tjerand suggests anonymous tagging of licenses
 - This was researched, but appears to be of a different use than what is relevant for the project

Meeting Minutes 20230130

Attendees	Ⓢ Sander Osvik Brekke Ⓛ Ivan Ⓚ Kristian
External attendees	Laszlo Erdodi
Event time	@January 30, 2023 3:15 PM → 4:00 PM
Place	Microsoft Teams
Type	Domain Expert Meeting
Tag	

Topics

- Java source code hardening
- Secrets inside and integrity of source code
- Source code integrity self check?

Details

- Source code hardening of Java is possible, but Java is not a good language for integrity.
- Integrity self checks are hard to perform, easy to bypass.
- Secrets in compiled languages, such as C++, should be fine. Secure HW is a plus

Meeting Minutes 20230201

Attendees	① Ivan ② Kristian ③ Sander Osvik Brekke
External attendees	Steven Yves Le Moan
Event time	@February 1, 2023 10:30 AM → 10:33 AM
Place	Microsoft Teams
Type	Thesis supervisor meeting
Tag	

Topics:

- Status meeting

Details

We had a short meeting with the thesis coordinator, Steven, where we informed him about our progress, and our plans onward.

Steven was content with our progress.

Meeting Minutes 20230203

Attendees	① Ivan ③ Sander Osvik Brekke ④ Kristian
External attendees	
Event time	@February 3, 2023 9:00 AM → 10:00 AM
Place	Discord
Type	Sprint meeting
Tag	

Topics

- Sprint Review
- Sprint Planning

Details

We ended up completing the sprint earlier than expected. This led us to do way more work, work which was completely unplanned. Luckily this didn't cause any problems this time around, but we are aware that we can't do this anymore and we should rather better plan our sprints.

During development we figured out that there are a lot of questions we didn't answer, so this sprint we will focus on answering these questions so that we can continue with development.

Meeting Minutes 20230209

Attendees	I Ivan S Sander Osvik Brekke K Kristian
External attendees	Anders Dale
Event time	@February 9, 2023 11:00 AM → 12:00 PM
Place	Microsoft Teams
Type	Client meeting
Tag	

Topics

- Forslag til løsning
- Sikre NMS er umulig

Details

Foreslår ide til Anders, hvor Virtuosoene på samme nettverk kan kommunisere med hverandre. Svar: "Nei".

Virtuoso er bare en server, og skal bare motta.

Da må vi holde track på data i VIP, og hva den har sendt ut.

Kunden ønsket eksplisitt å kunne deploye VIP hvor som helst, når som helst. Vi kan derfor ikke hardware låse VIP.

Tillitsbasert kundeforhold, derfor må vi anta at kundene ikke vil misbruke systemet.

Kan vi lage en stor kryptert fil som inneholder lisens og signatur?

VIP holder styr på alle virtuoso boksene, og vet derfor når de er ferdig med å recorder.

En UID kommer kun en gang, Nevion → VIP

Meeting Minutes 20230209

Attendees	① Ivan ③ Sander Osvik Brekke ④ Kristian
External attendees	Steven Yves Le Moan
Event time	@February 9, 2023 10:30 AM → 11:00 AM
Place	Microsoft Teams
Type	Thesis supervisor meeting
Tag	

Topics:

- Status meeting

Details

We had a short meeting with the thesis coordinator, Steven, where we informed him about our progress, and our plans onward.

Steven had nothing to comment on.

Meeting Minutes 20230215

Attendees	
External attendees	
Event time	@February 15, 2023 10:30 AM
Place	Microsoft Teams
Type	Thesis supervisor meeting
Tag	

This meeting was cancelled by Steven Yves Le Moan without description

Meeting Minutes 20230217

👤 Attendees	① Ivan ② Kristian ③ Sander Osvik Brekke
☰ External attendees	
📅 Event time	@February 17, 2023 9:00 AM → 10:00 AM
☰ Place	NTNU (A155)
⌵ Type	Sprint meeting
☰ Tag	

Topics

- Sprint Review
- Sprint Planning

Details

This is the first time we were unable to complete the sprint. This was caused by Sander and Ivan not being able to allocate more time to the bachelors project due to us having another course with a really heavy workload. We also failed on estimating the amount of work needing to be done on one of the issues, which led to the other issues standing still.

For the next sprint, we'll plan more accordingly and have smaller issues until Sander and I are done with the secondary course which would be in a month from now-

Meeting Minutes 20230222

Attendees	Ivan Sander Osvik Brekke Kristian
External attendees	Steven Yves Le Moan
Event time	@February 22, 2023 11:00 AM → 11:30 AM
Place	Microsoft Teams
Type	Thesis supervisor meeting
Tag	

Topics:

- Status meeting

Details

We had a short meeting with the thesis coordinator, Steven, where we informed him about our progress, and our plans onward.

Nothing else to be added.

Meeting Minutes 20230223

Attendees	Ⓚ Kristian Ⓢ Sander Osvik Brekke Ⓛ Ivan
External attendees	Anders Dale
Event time	@February 23, 2023 11:00 AM → 12:00 PM
Place	Microsoft Teams
Type	Client meeting
Tag	

Topics

- Proof of concept done soon
- Extension of PoC
- Walkthrough of current implementation

Details

ING course is eating up Ivan and Sanders time. Two weeks until finish maybe

Virtuoso hello request is realistic.

Virtuoso code not entirely functional yet.

We do system calls in Java, Anders does not mind.

We show both Java code and simple frontend code

Rework Java verification code. Don't send pubkey with request, should be imbedded in NMS

Retract unused license from virtuoso not in scope.

Would be nice to retract time from virtuoso, but on the very edge of scope.

All licenses need unique ID (delete request with ID)

Talked about how we save our license pool and hash contra their real solution with some ID number at pc and generate customer relationship keys from that

Meeting Minutes 20230224

👤 Attendees	① Ivan ③ Sander Osvik Brekke ④ Kristian
☰ External attendees	
📅 Event time	@February 24, 2023 9:00 AM → 10:00 AM
☰ Place	NTNU (A155)
⌵ Type	Sprint meeting
☰ Tag	

Topics

- Sprint Review
- Sprint Planning

Details

We were unable to complete the sprint again, caused by the workload in ING2300. We project that once the course is done, our swift progress will resume and we'll be done in no time. It is unfortunate, that one course would have this effect but we'll plan accordingly and split the workload more efficiently.

Meeting Minutes 20230301

Attendees	① Ivan ③ Sander Osvik Brekke ④ Kristian
External attendees	Steven Yves Le Moan
Event time	@March 1, 2023 11:00 AM → 11:30 AM
Place	Microsoft Teams
Type	Thesis supervisor meeting
Tag	

Topics

- Info about status report
- Status update

Details

Status report to be delivered 8.3.2023 to Steven. Should contain 2-3 pages regarding our current project status.

As per usual, we informed our coordinator that we are on progress to be done soon with the PoC. No further remarks.

Meeting Minutes 20230303

Attendees	① Ivan ③ Sander Osvik Brekke ④ Kristian
External attendees	Steven Yves Le Moan
Event time	@March 3, 2023 11:00 AM → 11:30 AM
Place	Microsoft Teams
Type	Thesis supervisor meeting
Tag	

Topics:

- Status meeting
- Presentation of Proof of Concept

Details

During this meeting, we demonstrated our proof of concept to Steven. Steven was very positive to our result and our progress.

Meeting Minutes 20230308

Attendees	Ⓢ Sander Osvik Brekke Ⓡ Ivan Ⓚ Kristian
External attendees	Steven Yves Le Moan
Event time	@March 8, 2023 11:00 AM → 11:15 AM
Place	Microsoft Teams
Type	Thesis supervisor meeting
Tag	

Topics:

- Status meeting

Details

We had a short meeting with Steven, where we informed him about our progress, and our plans onward.

Steven was content with our progress and had a positive outlook on our work.

Meeting Minutes 20230309

Attendees	Ⓢ Sander Osvik Brekke Ⓚ Kristian
External attendees	Anders Dale
Event time	@March 9, 2023 11:00 AM → 11:35 AM
Place	Microsoft Teams
Type	Client meeting
Tag	

Topics

- Demo showcase
- Security details
 - Separate Keypairs between NMS and each Virtuoso → see

[Security Hole](#)

Details

We showed Anders the demo successfully and had a long talk about how it all worked. Anders liked the implementation.

We also discussed certain specifics about the security and how using system Calls to OpenSSL might not be the best approach in a production environment due to the absence of control of what exactly “OpenSSL” calls on any given system. (A simple alias could drastically compromise our security 🚩🚩)

We also talked a bit about the source code of the virtuoso and the general algorithm for uploading sublicenses.

Meeting minutes 20230310

👤 Attendees	
☰ External attendees	
📅 Event time	@March 10, 2023 9:00 AM
☰ Place	Discord
⌵ Type	Sprint meeting
☰ Tag	

Sprint meeting 10.03.2023 is canceled due to illness.

Meeting Minutes 20230315

Attendees	
External attendees	
Event time	@March 15, 2023 11:00 AM
Place	Microsoft Teams
Type	Thesis supervisor meeting
Tag	

This meeting was cancelled by Steven Yves Le Moan without description

Meeting Minutes 20230322

Attendees	
External attendees	
Event time	@March 22, 2023 11:00 AM
Place	Microsoft Teams
Type	Thesis supervisor meeting
Tag	

This meeting was cancelled by Steven Yves Le Moan due to sick child.

Meeting Minutes 20230323

Attendees	Ⓢ Sander Osvik Brekke Ⓚ Kristian Ⓛ Ivan
External attendees	Anders Dale
Event time	@March 23, 2023 11:00 AM → 11:30 AM
Place	Microsoft Teams
Type	Client meeting
Tag	

Topics

- Status update

Details

After discussing with the client, we came to the conclusion that we need to add an ID field to all SLLs to ensure that each license can only be used with the corresponding LFA.

We will send the security review to the client when we need the them to proofread.

Brew up an email regarding the presentations.

Meeting Minutes 20230324

Attendees	① Ivan ③ Sander Osvik Brekke ④ Kristian
External attendees	
Event time	@March 24, 2023 9:00 AM → 9:30 AM
Place	Discord
Type	Sprint meeting
Tag	

Topics

- Sprint Review
- Sprint Planning

Details

Due to some member inactivity related to other courses, we are slowly beginning to fully work on the thesis again. This week's sprint was somewhat completed, but we could not complete it due to some member absence. We plan on doing some code refactoring for the next sprint and further work on the security review.

Meeting Minutes 20230329

Attendees	
External attendees	
Event time	@March 29, 2023 11:00 AM
Place	Microsoft Teams
Type	Thesis supervisor meeting
Tag	

This meeting was cancelled by Steven Yves Le Moan without description.

Meeting Minutes 20230330

Attendees	Ⓢ Sander Osvik Brekke Ⓛ Ivan Ⓚ Kristian
External attendees	Anders Dale
Event time	@March 30, 2023 3:00 PM → 3:40 PM
Place	Microsoft Teams
Type	Client meeting
Tag	

This meeting was moved from april 6th to this date

Meeting Minutes 20230331

Attendees	① Ivan ③ Sander Osvik Brekke ④ Kristian
External attendees	
Event time	@March 31, 2023 9:00 AM → 9:30 AM
Place	Discord
Type	Sprint meeting
Tag	

Topics

- Sprint Review
- Sprint Planning

Details

This sprint went a lot better and we are all back on track to work full time with the thesis. This sprints focus has been the security review and we have had major progress with it this week. We plan on speaking with a professional regarding our security review 31.03.2023 and will proceed accordingly after the feedback.

Next week will be Easter break, which will cause some minor unavailability some days but the goal is to complete the first draft of the security review of our concept.

Meeting Minutes 20230331

Attendees	Ⓚ Kristian Ⓢ Sander Osvik Brekke
External attendees	Erjon Zoto
Event time	@March 31, 2023 11:00 AM → 12:00 PM
Place	NTNU, Topas
Type	Domain Expert Meeting
Tag	

Topics

- How to write a security review?
- Revision of first draft

Details

- Erjon suggests to read more theory, in order to learn about the structure and content of a security review
- Erjon will gladly read more drafts, and give feedback.

Note: A second review was sent, but no feedback was received.

Meeting Minutes 20230405

Attendees	Ivan Sander Osvik Brekke
External attendees	Steven Yves Le Moan
Event time	@April 5, 2023 11:00 AM → 11:30 AM
Place	Microsoft Teams
Type	Thesis supervisor meeting
Tag	

Topics:

- Status meeting

Details

“motivasjonen” betyr konteksten/bakgrunnen for oppgaven

“konklusjon/oppsummering” bør legges til i kapittel 7.

alle andre kapitler bør ha innledning og konklusjon.

Meeting Minutes 20230407

👤 Attendees	
☰ External attendees	
📅 Event time	@April 7, 2023 9:00 AM
☰ Place	Discord
⌵ Type	Sprint meeting
☰ Tag	

This meeting was cancelled due to Easter break.

Meeting Minutes 20230412

Attendees	Ⓢ Sander Osvik Brekke Ⓚ Kristian
External attendees	Steven Yves LeMoan
Event time	@April 12, 2023 11:00 AM → 11:30 AM
Place	Microsoft Teams
Type	Thesis supervisor meeting
Tag	

Finish first draft of security review by 14th of April. Send to Steven for reviewing; feedback on the 19th of April.

No further questions.

Meeting Minutes 20230414

👤 Attendees	🗨️ Sander Osvik Brekke🗨️ Ivan🗨️ Kristian
☰ External attendees	
📅 Event time	@April 14, 2023 9:00 AM → 10:00 AM
☰ Place	Discord
📄 Type	Sprint meeting
☰ Tag	

Topics

- Sprint Review
- Sprint Planning

Details

Significant progress was made on several parts of the thesis, resulting in substantial restructuring. All issues related to the PoC were resolved except for one, which was deferred to the next sprint for finalization.

Meeting Minutes 20230420

Attendees	Ivan Kristian
External attendees	Anders Dale
Event time	@April 20, 2023 11:00 AM → 11:30 AM
Place	Microsoft Teams
Type	Client meeting
Tag	

This meeting it was agreed to no longer have weekly meetings, but invite to meetings when necessary.

Meeting Minutes 20230421

Attendees	Ⓢ Sander Osvik Brekke Ⓛ Ivan Ⓚ Kristian
External attendees	Steven Yves Le Moan
Event time	@April 21, 2023 11:00 AM → 11:30 AM
Place	Microsoft Teams
Type	Thesis supervisor meeting
Tag	

Topics:

- Status meeting

Details

We had a short meeting with Steven, where we informed him about our progress, and our plans onward. He had no comments to add, and was content with our progress.

Meeting Minutes 20230426

Attendees	Ⓚ Kristian Ⓢ Sander Osvik Brekke Ⓛ Ivan
External attendees	Steven Yves Le Moan
Event time	@April 26, 2023 11:00 AM → 11:15 AM
Place	Microsoft Teams
Type	Thesis supervisor meeting
Tag	

Topics

- Status update

Details

After giving Steven an update on our progress, we predicted to be done with the first draft by 5th of May. He agreed, and set aside some time in his calendar for reviewing the first draft.

Meeting Minutes 20230428

👤 Attendees	① Ivan ③ Sander Osvik Brekke ④ Kristian
☰ External attendees	
📅 Event time	@April 28, 2023 9:00 AM → 9:30 AM
☰ Place	Discord
⌵ Type	Sprint meeting
☰ Tag	

Topics

- Sprint Review
- Sprint Planning
- Sprint Retrospective

Details

The sprint "Process, Theory and PoC" lasted for 2 weeks and consisted of writing the final report. The team felt that this sprint went well and all issues were either done or very-very close to done. We also created a ToggleTrack Parser which was out of scope, but saves us valuable time when extracting data regarding the time of work we've tracked.

Meeting Minutes 20230503

Attendees	Ⓢ Sander Osvik Brekke Ⓛ Ivan Ⓚ Kristian
External attendees	Steven Yves Le Moan
Event time	@May 3, 2023 11:00 AM → 11:20 AM
Place	Microsoft Teams
Type	Thesis supervisor meeting
Tag	

Topics

- Thesis discussion
- Secrets in GitHub repositories

Details

After updating Steven on the groups progress, which he was happy with, we asked for him to slowly begin reading through out LaTeX document in order to give us some feedback. We were still on track to having a first draft of the report by 5th May.

The group asked questions about secrets and keys in the GitHub repository. Steven answered that, for the intention of easier grading, it is tolerated to leave the keys and secrets in the repositories.

Meeting Minutes 20230509

Attendees	Ⓢ Sander Osvik Brekke
External attendees	Steven Yves Le Moan
Event time	@May 9, 2023 11:30 AM → 12:00 PM
Place	Microsoft Teams
Type	Thesis supervisor meeting
Tag	

Topics

- Feedback after first draft delivery

Details

Steven thinks it is a good report that is written in a clear way.

He is happy that we are integrating his comments and suggestions to improvement.

We are working on further implementing his suggestions to improvement.

Easily read report, more than enough covering. Good flow and a clear “red line”

Through the report, many things that are not well enough reasoned for/against. We need to remember to reason any choice, decision and similar.

Somewhat missing reflection; we need to reflect more over our own product, work, etc. Perhaps relevant to the different discussions.

Please use the assessment criteria that are made available.

Check for syntax errors, spelling errors, etc. This is a big part of the impact of the report.

New read-through, will come back with new suggestions to improvement

Next meeting

Ordinary meeting tomorrow, if we do not get any other message. Else, the meeting will be postponed until Steven has finished his new read-through.

Meeting Minutes 20230511

Attendees	① Sander Osvik Brekke ① Ivan ① Kristian
External attendees	Steven Yves Le Moan
Event time	@May 11, 2023 2:00 PM → 2:30 PM
Place	Microsoft Teams
Type	Thesis supervisor meeting
Tag	

Topics

Security details

Thesis flow

Order of chapters

Rewritten chapter 1

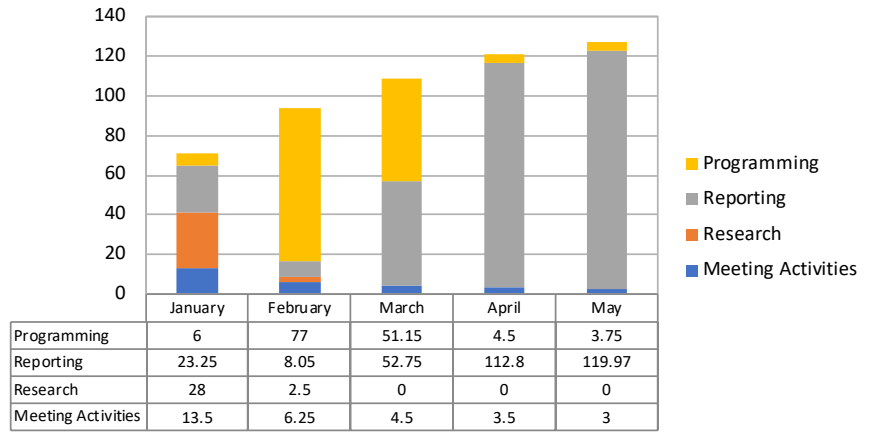
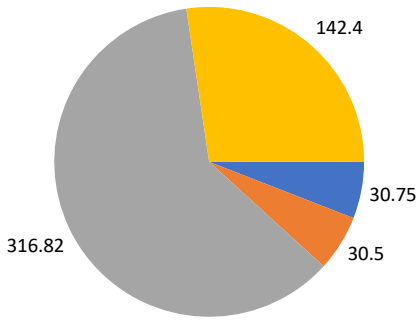
Details

- Steven read the report and says it looks great. He would give it a high grade.
 - Easily read
 - Very in-depth
- Chapter 1 looks good now
- Development process could be moved. Steven is unsure.
 - Move it if we want
 - Chicken and egg situation
 - Think it looks good, not necessary to move
- Report structure is good for readability.
- Steven likes the tables and figures
- Weird to read “protect revenue streams” in security review
 - We will look into adding further explanations
 - Steven says it makes sense, but was unsure.
- The reviews in Overleaf are all his last remarks.
- Steven can help practice for the presentation

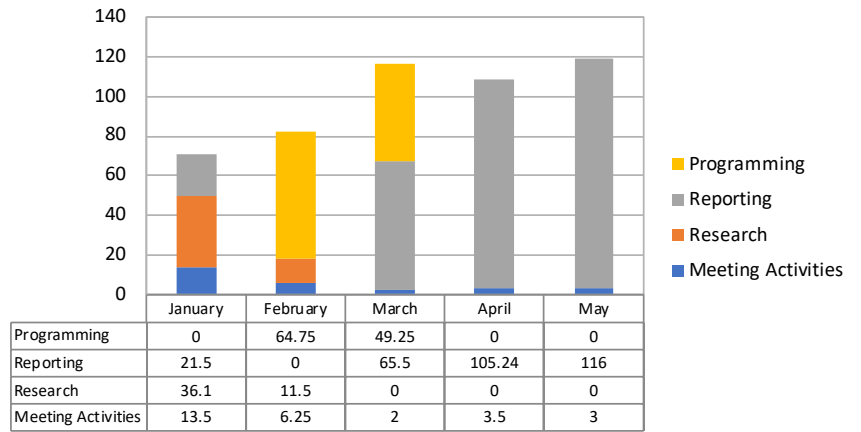
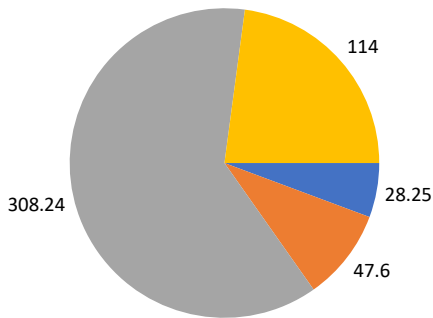
Appendix H

Time Sheets

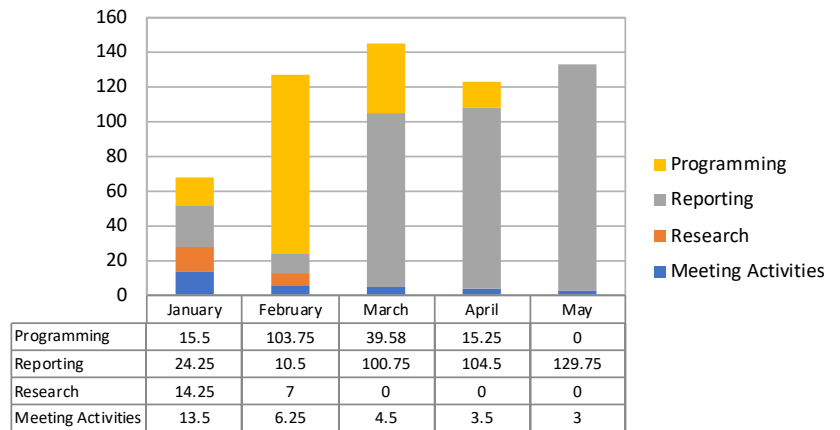
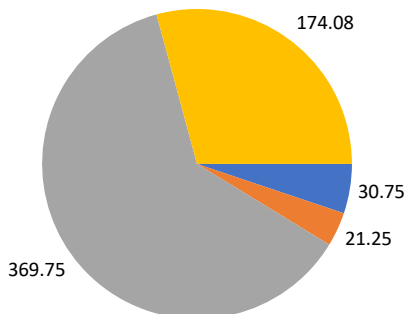
Sander



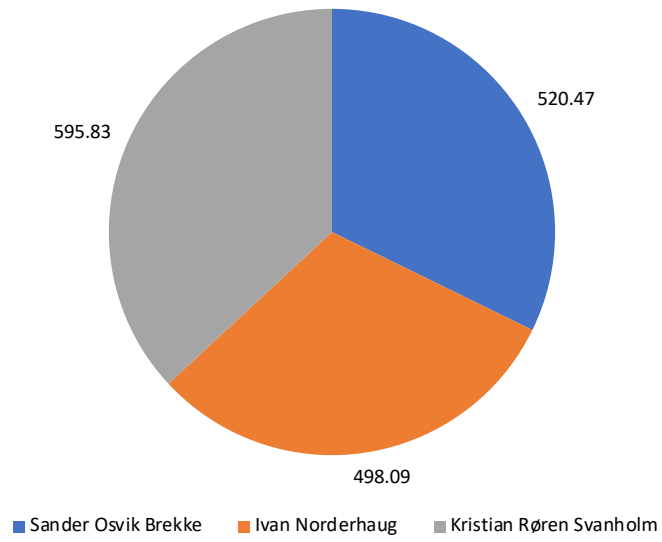
Ivan



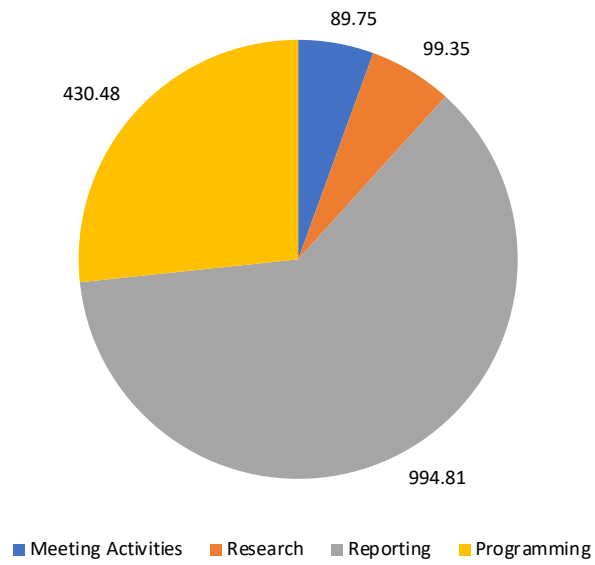
Kristian



Total Hours



Total Hours per Category



Appendix I

Frontend Graphical User Interface

NMS pool

 License file

SEND

Media Function	Time left	Description
J2KHDX	5700	J2K HD Encoders/Decoders

Rows per page: 10 1-1 of 1 < >

Generate sublicense

Send to



Media function

0

SEND

Aggregates

License File Consumer - Alpha - 127.0.0.1:8443

Media Function	Time left	Description	Consume	Actions
J2KHDX	300	J2K HD Encoders/Decoders	Time in seconds	 

Rows per page: 10 1-1 of 1 < >

Appendix J

Proof of Concept Initial Setup Guide

Initial Setup Guide

Background

This guide is written as a part of the proof of concept, written as a part of a bachelor project in Computer Science at NTNU Gjøvik, during the spring of 2023.

In order to use the demonstration of an OLM system, a set of keys and certificates are necessary. This guide will go through the steps necessary, in order to start, run and use the applications.

License Company Root Key Pair

The first step is to generate a key pair for the License Company, as the root.

This is done through:

1. Run the following command to create a private key and a certificate:

```
openssl req -x509 -newkey rsa:4096 \  
            -keyout rootkey.pem \  
            -out rootcert.crt -sha256 \  
            -days 3650
```

Then follow the interactive guide presented. This creates a private key and a certificate. The certificate is self-signed, and is valid for 10 years.

2. Run the following command to generate a public key from the certificate:

```
openssl x509 -in rootcert.crt \  
            -pubkey -noout > rootpub.pem
```

3. The private key (rootkey.pem) is kept in a secure place, the certificate (rootcert.crt) is distributed to the license file aggregator (follow steps below), and the public key (rootpub.pem) is distributed to the network management system (follow steps below). These are being used to certify the origin of license signatures at a later time.
4. Update the NMS source code file named "CustomerConstants.java" located in `src/main/java/no/ntnu/nms/CustomerConstants.java`, and replace the current root public key with the updated one, in the exact same format.

Network Management System Intermediate Key Pair

The next step is to generate an intermediate key pair for the Network Management System.

This is done through:

1. Run the following command to create a private key:

```
openssl genrsa -aes256 -out nmskey.pem 4096
```

2. Run the following command in order to create a certificate request:

```
openssl req -new -sha256 \  
            -key nmskey.pem \  
            -out nmscert.csr
```

Then follow the interactive guide presented. This creates a private key and a certificate request.

3. Run the following command to sign the intermediate key with the root key:

```
openssl x509 -req \  
            -in nmcert.csr \  
            -CA rootcert.crt \  
            -CAkey rootkey.pem -CAcreateserial \  
            -out nmcert.crt -days 3600
```

The intermediate key and intermediate certificate is to be distributed to the network management system.

4. Verify the intermediate certificate against the root certificate by running:

```
openssl verify -CAfile rootcert.crt nmcert.crt
```

Where the output should be `nmcert.crt: OK`.

5. Create a keystore object for the NMS software:

1. Create a pkcs12 keystore with the intermediate cert and intermediate private key by running:

```
openssl pkcs12 -export \  
            -in nmcert.crt \  
            -inkey nmskey.pem \  
            -out keystore.p12 \  
            -name keystore \  
            -password pass:secret
```

Please note that the password is `secret`, as stated in the source code. Do not use this in a production environment. 2. Place the keystore in `src/main/resources/`.

License File Aggregator Setup

1. Replace the root certificate, located in `cert/external/` with the newly created one, renamed to `root.cert`.

Please note: Do NOT alter the keys placed under `cert/internal/`. These are used for the web server HTTPS connection, and are not to be altered, unless new, signed, certificates are to be used.

Application Startup

To start the application, the different parts are dependant on being started in the right order, in order to connect properly.

1. Start the Network Management System by following its guide.
 1. Initialize the files
 2. Start the jar
2. Start the License File Aggregator by following its guide.
 1. Enter correct name and port for the LFA, and the correct IP and port of the NMS.
 2. Build the project
 3. Run the application

4. Enter the secret for the key
3. (Optional) Start the frontend
 1. Enter the correct IP and port of the NMS
 2. Build the project
 3. Run the application

Appendix K

Network Management System ReadMe

Network Management System (NMS)

Background

The network management system is developed and written as a part of a bachelor's thesis at the University of Science and Technology in Gjøvik, Norway.

Description

NMS, short for network management system, is one of the several components of an offline license management system.

Requirements

The application requires Java 11 to run, in addition to Maven.

In order to redeem licenses, a root public key is necessary, and to generate and distribute licenses, an intermediate private key and certificate is necessary. The root public key is included in the source code, inside `no/ntnu/nms/Constants.java`, the intermediate private key and the intermediate certificate is included in the keystore.

A guide on how to generate these keys and certificates can be found [HERE](#).

Launch

To launch the application, you need to use an IDE bundled with Maven or install Maven on your local machine.

Install Maven

For Linux Debian, Ubuntu, etc.:

```
sudo apt-get install maven
```

For MacOS: *Please note, Brew is required.*

```
brew install maven
```

Package the application

Package the application by running:

```
mvn clean compile package exec:java
```

Run the application

Then run the created Jar file:

```
java -jar target/nms-software-1.0-SNAPSHOT.jar
```

Appendix L

License File Aggregator ReadMe

License File Aggregator

Overview

Purpose

The License File Aggregator (LFA) is made to function as a functionality consumer. I.E the place where a license for a certain functionality is actually ran and used up.

It has been designed for operating within an Offline License Management System (OLM) and is assumed to be ran on proprietary hardware as to ensure its integrity.

This repo was made to be used together with the NMS and functions as a Proof of Concept together with it. This means certain functionality that would be required within a real OLM system is either missing / changed for time or due to it having been proved in the NMS already.

Project layout

```
.
|-- CMakeLists.txt
|-- LICENSE
|-- README.md
|-- src
|   |-- App.cpp
|   |-- AppComponent.hpp
|   |-- client
|   |   |-- client.hpp
|   |-- controller
|   |   |-- Controller.cpp
|   |   |-- Controller.hpp
|   |-- dto
|   |   |-- DTOs.hpp
|   |-- error
|   |   |-- error.cpp
|   |   |-- error.hpp
|   |-- file
|   |   |-- fileHandler.cpp
|   |   |-- fileHandler.hpp
|   |-- shared.hpp
|   |-- ssl
|   |   |-- certificates.cpp
|   |   |-- certificates.hpp
|-- test
|   |-- ControllerTest.cpp
|   |-- ControllerTest.hpp
|   |-- app
|   |   |-- MyApiTestClient.hpp
|   |   |-- TestComponent.hpp
|   |-- tests.cpp
|-- utility
|   |-- install-oatpp-modules.sh
```

Build and Run

Requiments

- Git submodule oatpp-openssl. Run following command to install.

```
$ git submodule update --init --recursive
```
- oatpp module installed. You may run `utility/install-oatpp-modules.sh` script to install required oatpp modules.

Build

For Linux:

```
$ mkdir build && cd build
$ cmake ..
$ make
```

For MacOS:

```
$ mkdir build && cd build
$ cmake -DOPENSSL_ROOT_DIR="/usr/local/opt/openssl@1.1" ..
$ make
```

Run

Inside the build folder.

```
$ ./lfa-exe # - run application.
```

the secret is 'secret' if using the default chain of trust.

When running the LFA it is important to enter the secret before sending NMS requests. If the secret has not been given, the server is not ready and the NMS will remove it from its list of active LFAs. Should this happen, the LFA would need a reboot to be recognized by the NMS.

Swapping certificates

If you are changing the chain of trust, all that needs to be swapped is the file `./cert/external/root.cert` with the new root certificate of the same name.

The contents of the `./cert/local/` folder is for HTTPS and can be swapped with new files of the same names if wanted.

Oatpp

This repo was based on a starter project of oat++. See more:

- [Oat++ Website](#)
- [Oat++ Github Repository](#)

Appendix M

Frontend ReadMe

OLM Frontend

Demo tool for interacting with Offline License Management system.

All requests to LFAs are first sent to the NMS and then relayed to the correct LFA in order to support the system topology.

Note This codebase is out of scope for the bachelor thesis, meaning less time has been spent on QA.

Project setup

Installs all dependencies:

```
npm install
```

Compiles and hot-reloads for development:

```
npm run dev
```

Compiles and minifies for production:

```
npm run build
```

The code should automatically detect an online **NMS** on the same device / network. If not, ensure that the **NMS** is running on port 8090.

Appendix N

License File Signing ReadMe

License File Signer (LFS)

Important

There is currently a private key located in this repository. By no means, whatsoever, should this be the case for anyone else. This is only here for simplicity sake so that people can clone and run the different projects without any other necessary steps.

Background

The license file signer is developed and written as a part of a bachelor's thesis at the University of Science and Technology in Gjøvik, Norway.

Description

LFS, short for license file signer, is the component acting as the licensing company in this proof of concept.

Requirements

- OpenSSL ([windows](#))

Usage

To use the LFS correctly, clone the repository and set it as the working directory. Then simply run the following command in the terminal:

For linux/macOS

```
./sign.sh <file_to_sign> <private_key_file>
```

where the `<file_to_sign>` is the *example_license.json* and the `<private_key_file>` is the *root-pk.key*

Finally, select the *example_license.json* and *example_license.json.signature* and compress them to a zip file. The payload is now ready to be used.

The structure should be looking like this:

```
payload.zip/  
├── license.json  
└── license.json.signature
```

Alternatively, there is a already signed and ready payload located in the repository, named *example_payload.zip*.

Appendix O

Checkstyle Ruleset

Code listing O.1: NMS Checkstyle Rulesheet.

```
<?xml version="1.0"?>
<!DOCTYPE module PUBLIC
    "-//Checkstyle//DTD Checkstyle Configuration 1.3//EN"
    "https://checkstyle.org/dtds/configuration_1_3.dtd">

<module name="Checker">
  <module name="SuppressWarningsFilter"/>

  <property name="charset" value="UTF-8"/>

  <property name="severity" value="warning"/>

  <property name="fileExtensions" value="java, properties, xml"/>
  <!-- Excludes all 'module-info.java' files -->
  <!-- See https://checkstyle.org/config_filefilters.html -->
  <module name="BeforeExecutionExclusionFileFilter">
    <property name="fileNamePattern" value="module\-info\.java$"/>
  </module>
  <!-- https://checkstyle.org/config_filters.html#SuppressionFilter -->
  <module name="SuppressionFilter">
    <property name="file" value="${org.checkstyle.google.suppressionfilter.config}"
      default="checkstyle-suppressions.xml" />
    <property name="optional" value="true"/>
  </module>

  <!-- Checks for whitespace -->
  <!-- See http://checkstyle.org/config_whitespace.html -->
  <module name="FileTabCharacter">
    <property name="eachLine" value="true"/>
  </module>

  <module name="LineLength">
    <property name="fileExtensions" value="java"/>
    <property name="max" value="100"/>
    <property name="ignorePattern" value="^package.*|^import.*|a
      href|href|http://|https://|ftp://" />
  </module>
```

```

<module name="TreeWalker">
  <module name="OuterTypeFilename"/>
  <module name="IllegalTokenText">
    <property name="tokens" value="STRING_LITERAL, CHAR_LITERAL"/>
    <property name="format"
      value="\\u00(09|0(a|A)|0(c|C)|0(d|D)|22|27|5(C|c))|\\(0(10|11|12|14|15|42|47)|134)"/>
    <property name="message"
      value="Consider using special escape sequence instead of octal
      value or Unicode escaped value."/>
  </module>
  <module name="AvoidEscapedUnicodeCharacters">
    <property name="allowEscapesForControlCharacters" value="true"/>
    <property name="allowByTailComment" value="true"/>
    <property name="allowNonPrintableEscapes" value="true"/>
  </module>
  <!--
  <module name="AvoidStarImport"/>
  -->
  <module name="OneTopLevelClass"/>
  <module name="NoLineWrap">
    <property name="tokens" value="PACKAGE_DEF, IMPORT, STATIC_IMPORT"/>
  </module>
  <module name="EmptyBlock">
    <property name="option" value="TEXT"/>
    <property name="tokens"
      value="LITERAL_TRY, LITERAL_FINALLY, LITERAL_IF, LITERAL_ELSE,
      LITERAL_SWITCH"/>
  </module>
  <module name="NeedBraces">
    <property name="tokens"
      value="LITERAL_DO, LITERAL_ELSE, LITERAL_FOR, LITERAL_IF,
      LITERAL_WHILE"/>
  </module>

  <module name="LeftCurly">
    <property name="tokens"
      value="CLASS_DEF, ANNOTATION_DEF, ENUM_CONSTANT_DEF, ENUM_DEF,
      CTOR_DEF, INTERFACE_DEF, METHOD_DEF, LAMBDA, LITERAL_CASE,
      LITERAL_CATCH, LITERAL_DEFAULT,
      LITERAL_DO, LITERAL_ELSE, LITERAL_FINALLY, LITERAL_FOR,
      LITERAL_IF,
      LITERAL_SWITCH, LITERAL_SYNCHRONIZED, LITERAL_TRY,
      LITERAL_WHILE,
      OBJBLOCK, STATIC_INIT, RECORD_DEF, COMPACT_CTOR_DEF"/>
  </module>
  <module name="RightCurly">
    <property name="id" value="RightCurlySame"/>
    <property name="tokens"
      value="LITERAL_TRY, LITERAL_CATCH, LITERAL_FINALLY, LITERAL_IF,
      LITERAL_ELSE,
      LITERAL_DO"/>
  </module>
  <module name="RightCurly">
    <property name="id" value="RightCurlyAlone"/>
    <property name="option" value="alone"/>
    <property name="tokens"
      value="CLASS_DEF, METHOD_DEF, CTOR_DEF, LITERAL_FOR, LITERAL_WHILE,
      STATIC_INIT,
      INSTANCE_INIT, ANNOTATION_DEF, ENUM_DEF, INTERFACE_DEF,
      RECORD_DEF,

```



```

        COMPACTCTOR_DEF"/>
</module>
<module name="SuppressionXpathSingleFilter">
  <!-- suppression is required till
    https://github.com/checkstyle/checkstyle/issues/7541 -->
  <property name="id" value="RightCurlyAlone"/>
  <property name="query" value="//RCURLY[parent::SLIST[count(./*)=1]
    or
    preceding-sibling::*[last()][self::LCURLY]]"/>
</module>
<module name="WhitespaceAfter">
  <property name="tokens"
    value="COMMA, SEMI, TYPECAST, LITERAL_IF, LITERAL_ELSE,
    LITERAL_RETURN,
    LITERAL_WHILE, LITERAL_DO, LITERAL_FOR, LITERAL_FINALLY,
    DO_WHILE, ELLIPSIS,
    LITERAL_SWITCH, LITERAL_SYNCHRONIZED, LITERAL_TRY,
    LITERAL_CATCH, LAMBDA,
    LITERAL_YIELD, LITERAL_CASE"/>
</module>
<module name="WhitespaceAround">
  <property name="allowEmptyConstructors" value="true"/>
  <property name="allowEmptyLambdas" value="true"/>
  <property name="allowEmptyMethods" value="true"/>
  <property name="allowEmptyTypes" value="true"/>
  <property name="allowEmptyLoops" value="true"/>
  <property name="ignoreEnhancedForColon" value="false"/>
  <property name="tokens"
    value="ASSIGN, BAND, BAND_ASSIGN, BOR, BOR_ASSIGN, BSR, BSR_ASSIGN,
    BXOR,
    BXOR_ASSIGN, COLON, DIV, DIV_ASSIGN, DO_WHILE, EQUAL, GE, GT,
    LAMBDA, LAND,
    LCURLY, LE, LITERAL_CATCH, LITERAL_DO, LITERAL_ELSE,
    LITERAL_FINALLY,
    LITERAL_FOR, LITERAL_IF, LITERAL_RETURN, LITERAL_SWITCH,
    LITERAL_SYNCHRONIZED,
    LITERAL_TRY, LITERAL_WHILE, LOR, LT, MINUS, MINUS_ASSIGN, MOD,
    MOD_ASSIGN,
    NOT_EQUAL, PLUS, PLUS_ASSIGN, QUESTION, RCURLY, SL, SLIST,
    SL_ASSIGN, SR,
    SR_ASSIGN, STAR, STAR_ASSIGN, LITERAL_ASSERT,
    TYPE_EXTENSION_AND"/>
  <message key="ws.notFollowed"
    value="WhitespaceAround: '{0}' is not followed by whitespace.
    Empty blocks
    may only be represented as '{0}' when not part of a multi-block
    statement (4.1.3)"/>
  <message key="ws.notPreceded"
    value="WhitespaceAround: '{0}' is not preceded with whitespace."/>
</module>
<module name="OneStatementPerLine"/>
<module name="MultipleVariableDeclarations"/>
<module name="ArrayTypeStyle"/>
<module name="MissingSwitchDefault"/>
<module name="FallThrough"/>
<module name="UpperEll"/>
<module name="ModifierOrder"/>
<module name="EmptyLineSeparator">
  <property name="tokens"

```

```

        value="PACKAGE_DEF, IMPORT, STATIC_IMPORT, CLASS_DEF,
        INTERFACE_DEF, ENUM_DEF,
        STATIC_INIT, INSTANCE_INIT, METHOD_DEF, CTOR_DEF,
        VARIABLE_DEF, RECORD_DEF,
        COMPACT_CTOR_DEF"/>
    <property name="allowNoEmptyLineBetweenFields" value="true"/>
</module>
<module name="SeparatorWrap">
    <property name="id" value="SeparatorWrapDot"/>
    <property name="tokens" value="DOT"/>
    <property name="option" value="nl"/>
</module>
<module name="SeparatorWrap">
    <property name="id" value="SeparatorWrapComma"/>
    <property name="tokens" value="COMMA"/>
    <property name="option" value="EOL"/>
</module>
<module name="SeparatorWrap">
    <!-- ELLIPSIS is EOL until https://github.com/google/styleguide/issues/259
    -->
    <property name="id" value="SeparatorWrapEllipsis"/>
    <property name="tokens" value="ELLIPSIS"/>
    <property name="option" value="EOL"/>
</module>
<module name="SeparatorWrap">
    <!-- ARRAY_DECLARATOR is EOL until
    https://github.com/google/styleguide/issues/258 -->
    <property name="id" value="SeparatorWrapArrayDeclarator"/>
    <property name="tokens" value="ARRAY_DECLARATOR"/>
    <property name="option" value="EOL"/>
</module>
<module name="SeparatorWrap">
    <property name="id" value="SeparatorWrapMethodRef"/>
    <property name="tokens" value="METHOD_REF"/>
    <property name="option" value="nl"/>
</module>
<module name="PackageName">
    <property name="format" value="^[a-z]+(\.[a-z][a-z0-9]*)*$"/>
    <message key="name.invalidPattern"
        value="Package name ''{0}'' must match pattern ''{1}''."/>
</module>
<module name="TypeName">
    <property name="tokens" value="CLASS_DEF, INTERFACE_DEF, ENUM_DEF,
        ANNOTATION_DEF, RECORD_DEF"/>
    <message key="name.invalidPattern"
        value="Type name ''{0}'' must match pattern ''{1}''."/>
</module>
<module name="MemberName">
    <property name="format" value="^[a-z][a-z0-9][a-zA-Z0-9]*$"/>
    <message key="name.invalidPattern"
        value="Member name ''{0}'' must match pattern ''{1}''."/>
</module>
<module name="ParameterName">
    <property name="format" value="^[a-z]([a-z0-9][a-zA-Z0-9]*)?$"/>
    <message key="name.invalidPattern"
        value="Parameter name ''{0}'' must match pattern ''{1}''."/>
</module>
<module name="LambdaParameterName">
    <property name="format" value="^[a-z]([a-z0-9][a-zA-Z0-9]*)?$"/>
    <message key="name.invalidPattern"

```

```

        value="Lambda parameter name ''{0}'' must match pattern ''{1}''."/>
</module>
<module name="CatchParameterName">
  <property name="format" value="^[a-z]([a-z0-9][a-zA-Z0-9]*)?$"/>
  <message key="name.invalidPattern"
    value="Catch parameter name ''{0}'' must match pattern ''{1}''."/>
</module>
<module name="LocalVariableName">
  <property name="format" value="^[a-z]([a-z0-9][a-zA-Z0-9]*)?$"/>
  <message key="name.invalidPattern"
    value="Local variable name ''{0}'' must match pattern ''{1}''."/>
</module>
<module name="PatternVariableName">
  <property name="format" value="^[a-z]([a-z0-9][a-zA-Z0-9]*)?$"/>
  <message key="name.invalidPattern"
    value="Pattern variable name ''{0}'' must match pattern ''{1}''."/>
</module>
<module name="ClassTypeParameterName">
  <property name="format" value="(^[A-Z][0-9]?)([A-Z][a-zA-Z0-9]*[T])$"/>
  <message key="name.invalidPattern"
    value="Class type name ''{0}'' must match pattern ''{1}''."/>
</module>
<module name="RecordComponentName">
  <property name="format" value="^[a-z]([a-z0-9][a-zA-Z0-9]*)?$"/>
  <message key="name.invalidPattern"
    value="Record component name ''{0}'' must match pattern ''{1}''."/>
</module>
<module name="RecordTypeParameterName">
  <property name="format" value="(^[A-Z][0-9]?)([A-Z][a-zA-Z0-9]*[T])$"/>
  <message key="name.invalidPattern"
    value="Record type name ''{0}'' must match pattern ''{1}''."/>
</module>
<module name="MethodTypeParameterName">
  <property name="format" value="(^[A-Z][0-9]?)([A-Z][a-zA-Z0-9]*[T])$"/>
  <message key="name.invalidPattern"
    value="Method type name ''{0}'' must match pattern ''{1}''."/>
</module>
<module name="InterfaceTypeParameterName">
  <property name="format" value="(^[A-Z][0-9]?)([A-Z][a-zA-Z0-9]*[T])$"/>
  <message key="name.invalidPattern"
    value="Interface type name ''{0}'' must match pattern ''{1}''."/>
</module>
<module name="NoFinalizer"/>
<module name="GenericWhitespace">
  <message key="ws.followed"
    value="GenericWhitespace ''{0}'' is followed by whitespace."/>
  <message key="ws.preceded"
    value="GenericWhitespace ''{0}'' is preceded with whitespace."/>
  <message key="ws.illegalFollow"
    value="GenericWhitespace ''{0}'' should followed by whitespace."/>
  <message key="ws.notPreceded"
    value="GenericWhitespace ''{0}'' is not preceded with whitespace."/>
</module>
<module name="Indentation">
  <property name="basicOffset" value="4"/>
  <property name="braceAdjustment" value="2"/>
  <property name="caseIndent" value="4"/>
  <property name="throwsIndent" value="4"/>
  <property name="lineWrappingIndentation" value="4"/>
  <property name="arrayInitIndent" value="4"/>

```

```

</module>
<module name="AbbreviationAsWordInName">
  <property name="ignoreFinal" value="false"/>
  <property name="allowedAbbreviationLength" value="0"/>
  <property name="tokens"
    value="CLASS_DEF, INTERFACE_DEF, ENUM_DEF, ANNOTATION_DEF,
          ANNOTATION_FIELD_DEF,
          PARAMETER_DEF, VARIABLE_DEF, METHOD_DEF, PATTERN_VARIABLE_DEF,
          RECORD_DEF,
          RECORD_COMPONENT_DEF"/>
</module>
<module name="NoWhitespaceBeforeCaseDefaultColon"/>
<module name="OverloadMethodsDeclarationOrder"/>
<module name="VariableDeclarationUsageDistance"/>
<module name="MethodParamPad">
  <property name="tokens"
    value="CTOR_DEF, LITERAL_NEW, METHOD_CALL, METHOD_DEF,
          SUPER_CALL, ENUM_CONSTANT_DEF, RECORD_DEF"/>
</module>
<module name="NoWhitespaceBefore">
  <property name="tokens"
    value="COMMA, SEMI, POST_INC, POST_DEC, DOT,
          LABELED_STAT, METHOD_REF"/>
  <property name="allowLineBreaks" value="true"/>
</module>
<module name="ParenPad">
  <property name="tokens"
    value="ANNOTATION, ANNOTATION_FIELD_DEF, CTOR_CALL, CTOR_DEF, DOT,
          ENUM_CONSTANT_DEF,
          EXPR, LITERAL_CATCH, LITERAL_DO, LITERAL_FOR, LITERAL_IF,
          LITERAL_NEW,
          LITERAL_SWITCH, LITERAL_SYNCHRONIZED, LITERAL_WHILE,
          METHOD_CALL,
          METHOD_DEF, QUESTION, RESOURCE_SPECIFICATION, SUPER_CALL,
          LAMBDA,
          RECORD_DEF"/>
</module>
<module name="OperatorWrap">
  <property name="option" value="NL"/>
  <property name="tokens"
    value="BAND, BOR, BSR, BXOR, DIV, EQUAL, GE, GT, LAND, LE,
          LITERAL_INSTANCEOF, LOR,
          LT, MINUS, MOD, NOT_EQUAL, QUESTION, SL, SR, STAR, METHOD_REF,
          TYPE_EXTENSION_AND "/>
</module>
<module name="AnnotationLocation">
  <property name="id" value="AnnotationLocationMostCases"/>
  <property name="tokens"
    value="CLASS_DEF, INTERFACE_DEF, ENUM_DEF, METHOD_DEF, CTOR_DEF,
          RECORD_DEF, COMPACT_CTOR_DEF"/>
</module>
<module name="AnnotationLocation">
  <property name="id" value="AnnotationLocationVariables"/>
  <property name="tokens" value="VARIABLE_DEF"/>
  <property name="allowSamelineMultipleAnnotations" value="true"/>
</module>
<module name="NonEmptyAtclauseDescription"/>
<module name="InvalidJavadocPosition"/>
<module name="JavadocTagContinuationIndentation"/>
<module name="SummaryJavadoc">

```

```

    <property name="forbiddenSummaryFragments"
        value="^@return the *|^This method returns |^A [{}@code
            [a-zA-Z0-9]+{}]( is a )"/>
</module>
<module name="JavadocParagraph"/>
<module name="AtclauseOrder">
    <property name="tagOrder" value="@param, @return, @throws, @deprecated"/>
    <property name="target"
        value="CLASS_DEF, INTERFACE_DEF, ENUM_DEF, METHOD_DEF, CTOR_DEF,
            VARIABLE_DEF"/>
</module>
<module name="JavadocMethod">
    <property name="accessModifiers" value="public"/>
    <property name="allowMissingParamTags" value="false"/>
    <property name="allowMissingReturnTag" value="false"/>
    <property name="allowedAnnotations" value="Override, Test"/>
    <property name="tokens" value="METHOD_DEF, CTOR_DEF, ANNOTATION_FIELD_DEF,
        COMPACT_CTOR_DEF"/>
</module>

<module name="MissingJavadocMethod">
    <property name="scope" value="public"/>
    <property name="minLineCount" value="0"/>
    <property name="allowedAnnotations" value="Override, Test"/>
    <property name="allowMissingPropertyJavadoc" value="false"/>

    <property name="tokens" value="METHOD_DEF, CTOR_DEF, ANNOTATION_FIELD_DEF,
        COMPACT_CTOR_DEF"/>
</module>
<module name="MissingJavadocType">
    <property name="scope" value="protected"/>
    <property name="tokens"
        value="CLASS_DEF, INTERFACE_DEF, ENUM_DEF,
            RECORD_DEF, ANNOTATION_DEF"/>
    <property name="excludeScope" value="nothing"/>
</module>
<module name="MethodName">
    <property name="format" value="^[a-z][a-z0-9]\w*$"/>
    <message key="name.invalidPattern"
        value="Method name ''{0}'' must match pattern ''{1}''."/>
</module>
<module name="SingleLineJavadoc"/>
<module name="EmptyCatchBlock">
    <property name="exceptionVariableName" value="expected"/>
</module>
<module name="CommentsIndentation">
    <property name="tokens" value="SINGLE_LINE_COMMENT, BLOCK_COMMENT_BEGIN"/>
</module>
<!-- https://checkstyle.org/config_filters.html#SuppressionXpathFilter -->
<module name="SuppressionXpathFilter">
    <property name="file"
        value="{org.checkstyle.google.suppressionxpathfilter.config}"
        default="checkstyle-xpath-suppressions.xml" />
    <property name="optional" value="true"/>
</module>
<module name="SuppressWarningsHolder" />
<module name="SuppressionCommentFilter">
    <property name="offCommentFormat" value="CHECKSTYLE.OFF\: ([\w\|]+)" />
    <property name="onCommentFormat" value="CHECKSTYLE.ON\: ([\w\|]+)" />
    <property name="checkFormat" value="$1" />

```

```
</module>
<module name="SuppressWithNearbyCommentFilter">
  <property name="commentFormat" value="CHECKSTYLE.SUPPRESS\:([\w\|]+)"/>
  <!-- $1 refers to the first match group in the regex defined in
    commentFormat -->
  <property name="checkFormat" value="$1"/>
  <!-- The check is suppressed in the next line of code after the comment -->
  <property name="influenceFormat" value="1"/>
</module>
</module>
</module>
```