Faculty of Sciences and Technology
Department of Informatics Engineering

RISK MANAGEMENT IN BLOCKCHAIN BASED SYSTEMS

Rui Jorge Fonseca Casaleiro

Final Internship Report in the scope of the
Master's in Informatics Engineering (Specialization in Software Engineering),
advised by Professor Paulo Simões and by Professor Fernando Boavida Fernandes,
and presented to the Faculty of Sciences and Technology (Dep. of Informatics Engineering).

January 2020

UNIVERSIDADE D
COIMBRA

Acknowledgements

A Master thesis is the culmination of work developed over an extended period of time, during which we experience an array of situations and emotions that range from feeling on the top of the world with every new puzzle piece that fits together, to facing hardships that makes us question all our work and decisions ever made during the whole process.

In all these situations, regardless of the nature of these moments, there are people that cross our path and share the experience with us, providing valuable guidance, either by keeping our feet on the ground or putting us back to the right track and mindset. For this, I would like to show my appreciation to those who supported me through this journey.

Firstly, I thank my supervisors Professor Paulo Simões and Professor Fernando Boavida, for the knowledge and insight shared that guided me through this journey and without whom this work would not be possible.

Secondly, and also including my supervisors, I give my thanks the PoSeID-on UC team, Professor Marília Curado, Professor Edmundo Monteiro, Professor Nuno Antunes and Dr. Paulo Silva, for providing such an incredible atmosphere to grow in, both academically and as an individual.

To my colleagues at the LCT, for all the time and moments shared, some of which will definitely become engraved in my memory for years to come. And if I do forget, since human memory is volatile, at least we have the pictures to remember.

To my family, my most grateful thanks, for supporting and motivating me through all my life. This extends to my friends, which are a family in itself, by sheer amount of years spent together, in particular to my best friend Maria for sharing the journey with me, despite fighting the same fight in a different city.

Finally, I would like to thank my girlfriend Sofia for all the support, love and patience throughout this process and without whom I would not have finished this work.

Abstract

With the growth of big data analytics more and more services and organizations have started collecting and processing personal information and customer behavior in order to optimize their business strategies and directed marketing. Furthermore, public and private organizations are embracing the information and communication technologies and moving their services to more convenient electronic alternatives, managing and collecting information on a larger scale and much more frequently.

Having sensitive personal information being collected, kept and shared between multiple organizations and services on a daily basis creates a significant risk for individuals as personal information can be used for fraudulent purposes. Increasing the severity of the issue, tracking which organizations are keeping what data about you is nearly impossible.

The General Data Protection Regulation (GDPR) implemented on the 25th of May 2018 aims to protect individuals by imposing stricter regulations regarding collection and processing of personal identifiable information, in order to mitigate the potential malicious use and propagation of such information.

The Protection and control of Secured Information by means of a privacy enhanced Dashboard (PoSeID-on) H2020 project's goal is to address this by creating a platform for individuals to centrally manage their personal data across several services and organizations, while providing businesses with the tools for safeguarding the rights of individuals as declared on the regulation.

The work developed for this thesis, consists on the design and preliminary development of a module for analysis and risk identification in Personal Identifiable Information (PII) exchanges within the PoSeID-on platform. This report presents the architectural description and the interim implementation of this module, which receives and analyses system and PII operations log in real-time, for identification and notification of possible privacy risks to data subjects PII.

Keywords

Anomaly Detection, Machine Learning, Personal Identifiable Information, Privacy, Log Parsing

Resumo

Com o crescimento da análise de dados em grande escala, cada vez mais serviços e organizações armazenam e processam dados pessoais e padrões comportamentais de utilizadores de modo a otimizar os seus processos de negócio e marketing. Para além disso, cada vez mais organizações públicas e privadas adotam serviços de informação e comunicação e movem os seus serviços para meios eletrónicos mais convenientes, gerindo e armazenando informação em grande escala e com mais frequência.

Ter informação sensível e pessoal a ser armazenada, processada e partilhada entre múltiplas organizações e serviços cria um risco significativo para os indivíduos em causa, pois a sua informação pessoal pode ser utilizada para fins fraudulentos. Aumentando a severidade da situação, manter um registo sobre que organizações possuem que dados sobre um individuo é praticamente impossível.

O Regulamento Geral de Proteção de Dados (RGPD) implementado no dia 25 de Maio de 2018, tem como objetivo proteger os indivíduos impondo uma legislação mais restrita sobre a coleta e processamento de informação pessoalmente identificável, de maneira a mitigar o potencial uso malicioso e propagação deste tipo de informação.

O projecto H2020 PoSeID-on tem como objetivo endereçar o problema através do desenvolvimento de uma plataforma através da qual indivíduos podem gerir, de uma maneira centralizada, os dados pessoais partilhados com diversos serviços e organizações. Por outro lado, a criação desta plataforma disponibiliza a estas organizações ferramentas para garantir a segurança dos seus utilizadores, protegendo os mesmos de acordo com o regulamento RGPD.

O trabalho desenvolvido nesta tese, consiste no design e desenvolvimento de um módulo para análise de dados e identificação de riscos em transações e operações envolvendo dados pessoais, dentro da plataforma PoSeID-on. Este relatório apresenta a descrição arquitetural e implementação do protótipo deste módulo, que recebe e analisa registos de sistema e registos de operações sobre dados pessoais em tempo real, para identificação e notificação sobre possíveis riscos para os dados pessoais dos utilizadores.

Palavras Chave

Deteção de Anomalias, Aprendizagem Computacional, Informação Pessoalmente Identificável, Análise de Registos de Sistema

Contents

Acknowledgements ..... **i**

Abstract ..... **iii**

Resumo ..... **v**

Contents ..... **vii**

Acronyms ..... **ix**

List of Figures ..... **xi**

List of Tables ..... **xiii**

Chapter 1 Introduction **1**

1.1 Motivation 1

1.2 Objectives 2

1.3 The PoSeID-on Project 3

1.3.1 PoSeID-on Goals and motivation 3

1.3.2 PoSeID-on Conceptual Architecture 4

1.3.3 Consortium Partners 5

1.4 Thesis Contributions 6

1.5 Structure 7

Chapter 2 Background **9**

2.1 General Data Protection Regulation 9

2.2 Blockchain 11

2.2.1 Blockchain Categories 13

2.2.2 Overview of Blockchain Components 14

2.3 Anomaly Detection 16

2.4 Log Anomaly Detection 24

2.5 Summary 25

Chapter 3 PoSeID-on Architecture **27**

3.1 Envisioned solution 27

3.2 System actors 28

3.3 System components 29

3.4 Use Cases 31

3.4.1 Initiated by the Data Processor 31

3.4.2 Initiated by the Data Subject 32

3.5 Summary 32

Chapter 4 RMM Design and Specification **33**

4.1.1 Methodology 33

4.1.2 Objectives 33

4.1.3 High-level Functional Requirements 33

4.1.4 Quality Attributes 34

4.1.5 Technical Constraints 35

4.1.6 GDPR concerns ... 36
4.1.7 Data Sources ... 37
4.1.8 Interaction Flow ... 38
4.2 Anomaly Detection Approach ... 39
4.2.1 Log Collection ... 39
4.2.2 Log Parsing ... 40
4.2.3 Feature extraction ... 42
4.2.4 Anomaly Detection ... 43
4.2.5 Stream and Batch Approach ... 44
4.3 Proposed Architecture ... 44
4.3.1 Components ... 46
4.3.2 Interfaces ... 47
4.3.3 Parallelization ... 48
4.4 Summary ... 50
Chapter 5 Implementation and Integration ... 51
5.1 Technological Stack Choices ... 51
5.1.1 Adopted Frameworks ... 51
5.1.2 Adopted Libraries and Packages ... 52
5.1.3 Adopted Tools ... 52
5.1.4 Adopted Storage Database ... 53
5.2 Implemented components ... 53
5.2.1 RMM Message Handler ... 55
5.2.2 Stream Processor ... 55
5.2.3 Risk Manager ... 56
5.2.4 Batch Processor ... 56
5.2.5 External Storage ... 56
5.2.6 Orchestrator ... 57
5.3 Integration Work ... 57
5.3.1 Development Environment ... 57
5.3.2 Module Configuration ... 57
5.3.3 Message Protocol ... 58
5.3.4 Integration Testing ... 59
5.4 Challenges ... 60
5.5 Summary ... 60
Chapter 6 Evaluation ... 61
6.1 Evaluation Strategy and Challenges ... 61
6.1.1 Experimental Setup and Evaluation ... 61
6.1.2 Fulfilled module requirements and quality attributes ... 69
6.1.3 Fulfilled PoSeID-on milestones ... 71
6.1.4 Publications ... 72
6.1.5 Fulfilled thesis objectives ... 73
6.2 Summary ... 73
Chapter 7 Conclusions and Future Work ... 75
7.1 Conclusions ... 75
7.2 Future Work ... 76
Bibliography ... 77

Acronyms

API	Application Programming Interface
ASR	Architecturally Significant Requirement
BDAF	Big Data Analytics Framework
CA	Certificate Authority
D2.2	System Requirements and Architecture Deliverable
DPD	Data Protection Directive
DoW	Description of Work
EU	European Union
GDPR	General Data Protection Regulation
GMM	Gaussian Mixture Model
IoT	Internet of Things
JAR	Java Archive
JVM	Java Virtual Machine
ML	Machine Learning
NIDS	Network Intrusion Detection System
OCSVM	One Class Support Vector Machine
PDA	Personal Data Analyzer
PII	Personal Identifiable Information
PoSeID-on	Protection and control of Secured Information by means of a privacy enhanced Dashboard
QoE	Quality of Experience
RMM	Risk Management Module
RPC	Remote Procedure Call
UC	University of Coimbra
eID	Electronic IDentification
eIDAS	Electronic IDentification, Authentication and trust Services

List of Figures

Figure 1.1 PoSeID-on original architecture proposal 5
Figure 2.1 Example of Transaction (from [16]) 15
Figure 2.2 Generic framework for anomaly detection [22] 17
Figure 2.3 Graphical example of log anomaly detection [12] 25
Figure 3.1 PoSeID-on architecture proposal 28
Figure 4.1 RMM System Interaction (General Overview) 38
Figure 4.2 Summary of Automated Log Parsing Tools (from [53]) 41
Figure 4.3 Example of Log Templates 42
Figure 4.4 RMM Architecture Proposal 45
Figure 4.5 Orchestrator Data Flow Using Message Queue 49
Figure 4.6 RMM Orchestrator Architecture 50
Figure 5.1 Implemented Data Flow 54
Figure 5.2 Graphical Representation of the Message Packet 59
Figure 6.1 Graphical Comparison of Experimental Tests 69
Figure 6.2 Requirement Fulfilment Status 70
Figure A.1 First Semester Workplan 84
Figure A.2 First Semester Effective Work 85
Figure A.3 Second Semester Workplan 86
Figure A.4 Second Semester Effective Work 87
Figure A.5 Extra Semester Effective Work 88

List of Tables

Table 1.1 Project Partners 6
Table 2.1 Compilation of several algorithms used in anomaly detection 23
Table 4.1 Risk Management Module Requirements 34
Table 4.2 Quality Attributes associated with the RMM 35
Table 4.3 Technical Constraints 36
Table 4.4 - Message Parameters 40
Table 5.1 RMM Component Implementation Status (January 2020) 53
Table 6.1 HDFS Dataset Characteristics 62
Table 6.2 Machine Characteristics 63
Table 6.3 Java Virtual Machine Parameters 63
Table 6.4 Spark Local Cluster Configuration 63
Table 6.5 Drain Configuration 64
Table 6.6 Drain Java Implementation Results 65
Table 6.7 K-Means Initialization 65
Table 6.8 Parameters Tested 66
Table 6.9 Test Results 68
Table 6.10 Quality Attributes Fulfilment 71
Table 6.11 PoSeID-on Milestones 72
Table A.1 PoSeID-on Milestones (Reminder) 83

Chapter 1 Introduction

This document is the final thesis report, in the scope of the master's in informatics engineering. This thesis was conducted in the Laboratory of Communications and Telematics of the Center for Informatics and Systems of University of Coimbra (LCT/CISUC), under the supervision of Professors Paulo Simões and Fernando Boavida Fernandes.

The internship was conducted in the scope of the H2020 PoSeID-on research project[1], of which LCT/CISUC is one of the main partners. This research project aims at devising, building and evaluating a platform for supervising the exchanges of Personal Identifiable Information (PII) between different data processors, in order to ensure compliance with recent regulations such as the EU's General Data Protection Regulation (GDPR)[2] and to enable data subjects (e.g. citizens) with tools for properly managing and enforcing their rights to control their PII.

The initial objective of the internship was the design, implementation and integration of the PoSeID-on platform's Risk Management Module (RMM), one of the two platform components to be developed by LCT/CISUC. Nevertheless, since the internship started in the early months of the project, when the PoSeID-on platform itself was not sufficiently defined, significant contributions were also provided to the overall design of PoSeID-on platform and, more recently, to the platform integration process.

This report details the design and development of the preliminary version of the aforementioned RMM – a module for detecting anomalous behaviors within the PoSeID-on platform. It also provides an overview of the PoSeID-on initial architecture design, which was conducted in parallel with the RMM and in which the LCT/CISUC (and the candidate) took a key role, leading, documenting and moderating the whole process.

This chapter describes the motivation and goals for the work developed for this thesis by contextualizing it in the PoSeID-on project. An overview of the PoSeID-on project is provided, followed by the primary goals of this work. A summary of the major contributions and the structure of the document are also presented.

1.1 Motivation

Personal information has never been more valuable[3]. Services and organizations have been collecting and processing personal information and customer patterns in order to optimize their businesses for years and, with the widespread adoption of information and communication technologies, even in services which regularly operate outside this medium, the scale and speed at which this gathering of information occurs has increased significantly. As personal information collection and usage grew, so did awareness of the value and risk of exposing this type of information.

On the 25th of May 2018, the European Union's (EU) General Data Protection Regulation (GDPR) was officially approved, forcing all public and private organizations selling goods or services or in any way collecting and processing personal data of individuals residing in the EU to become compliant. This set of regulations, giving new rights and more control to individuals and in consequence making these organizations accountable for the data they

own about each individual, creates a mandatory step to take in order to achieve compliance: the implementation of tools or systems to provide these rights but most importantly, to guarantee the unhindered functioning of previously provided services. Even today, most organizations have no streamlined tools for allowing data subjects to check which PII they keep and share with others – data subjects are relegated to means such as phone calls, emails or even mail to contact data processors, and data processors also process those requests in a manual fashion which is not cost-effective, efficient or even satisfactory to data subjects.

The PoSeID-on H2020 research project[1] aims to devise, build and demonstrate an innovative and intrinsically scalable platform that data subjects may use to interact with a wide set of data processors (e.g. all data processors from a specific country), in order to monitor and control their PII – checking which PII is kept, processed or exchanged by each data processor, and defining their consent (or lack of) per operation, per PII and per processor, in a fine grained manner. Such a centralized platform is necessary for actually enabling the vision of GDPR, registering exchanges and storage of PII and making them visible to data subjects and regulators.

The motivation behind the work developed and reported in this thesis is the necessity of monitoring the operations that take place within the PoSeID-on platform in order to detect possible anomalies and privacy risks associated with the exchange of PII.

This will allow data subjects to not only be aware of possible risks affecting their personal data but also to make informed decisions over the processing of their PII.

1.2 Objectives

The main goal of the work reported in this thesis is to contribute to the PoSeID-on platform by researching, designing and developing a module for monitoring the platform and detecting risks and anomalies associated with the exchange and processing of PII. In addition, there is a goal the of contributing to the overall architecture of the PoSeID-on platform and to the integration activities.

In a clear and concise way, the goals were the following:

1. To study:
    - the implications of GDPR in data processing and software development.
    - the basis of blockchain technologies.
    - the state-of-the-art in anomaly detection, in order to devise the RMM.
2. To contribute to the definition of the PoSeID-on architecture, based on preliminary study and by promoting and participating in the architecture design and documentation activities.
3. To design and develop a prototype module for privacy risk identification (the RRM), based on available machine learning techniques for anomaly detection and feed by data provided by other PoSeID-on platform components.
4. To integrate this module into the PoSeID-on platform and to test it, in order to validate the adopted anomaly detection approaches.

1.3 The PoSeID-on Project

As already mentioned, the work reported in this thesis was performed in the scope of the PoSeID-on project [1]. The University of Coimbra (UC) team contributes to this project with several activities, of which the more relevant, for the purpose of this thesis, were leading the definition of the overall platform architecture (Task 2.2) and the design and development of the components not directly related with blockchain technology (Work Package 4), of which the already mentioned RMM is part (Task 4.2).

The RMM implementation is planned in two phases. A first version has been released in August 2019, for platform integration and usage in a first round of Pilot Trials, and a second (and final) RMM release will be provided by May 2020.

This section aims to contextualize the reader with the project, by giving a general overview of PoSeID-on's motivation, goal, envisioned solution and conceptual architecture. A large part of the content provided here is directly adapted from the project's initial documentation (i.e. the project proposal).

1.3.1 PoSeID-on Goals and motivation

With the enforcement of the GDPR starting from the 25th of May 2018, all public and private organizations selling goods or services or in any way collecting and processing personal data of individuals residing in the EU are forced to be compliant to the regulation or else become subject to fines that can reach 20 million euro or 4 percent of a company's worldwide annual turnover. This set of regulations, giving new rights and more control to individuals and in consequence making these organizations accountable for the data they own about each individual creates a mandatory step to take in order to achieve compliance: the implementation of tools or systems to provide these rights and most importantly, to guarantee the unhindered functioning of previously provided services.

The PoSeID-on project aims to develop a transparent and intrinsically scalable ecosystem for personal data protection, providing the necessary tools for businesses and public organizations to be able to be compliant to the GDPR while providing data subjects with a way to manage access and authorizations to personal data in an easy, secure and independent way, while safeguarding their rights. The use of the PoSeID-on platform for personal data management facilitates the transition to compliance for these services and encourages new business opportunities by creating an ecosystem based on transparency, trust and security for personal data sharing, while leveraging innovative technologies such as blockchain, smart contracts and the cloud. In summary, the PoSeID-on platform aims at managing all the PII transactions between data subjects and data processors/controllers in order to:

1. Define the treatment purpose of all personal data and exactly who (external entities) can access it.
2. Define the set of information that can be processed for each specific purpose, by a particular entity.
3. Identify all the entities involved in the processing of the personal data of a data subject.
4. Secure and track transactions through which personal data is shared between data subjects and the data controllers (who define the purpose of data processing).

5. Secure and track transactions through which personal data is shared between data controllers and data processors.
6. Make informed decisions on who will be allowed to process personal data, based on data controller/provider trustworthiness.
7. Alert of risks and threats in case of privacy exposure due to anomalies on the transactions or the detection of Personally Identifiable Information.

Evaluation of the platform and achievement of the aforementioned goals will be done through four pilot studies across four countries, each testing its functionalities in public, private and mixed contexts. An Italian pilot will enhance e-services for management of public sector employees. A Spanish pilot will integrate and improve the e-government services available for the citizens of Santander. An Austrian pilot was planned to focus on integrating PoSeID-on with a e-government platform for business and organizations management (meanwhile this Pilot has been substantially changed), and a French pilot will simplify e-services for French citizens. These pilots will be run in controlled environments, with a select set of users, in order to simulate real life services and conditions.

1.3.2 PoSeID-on Conceptual Architecture

The PoSeID-on project consists in designing, implementing and validating a Privacy Enhancing Dashboard for personal data protection, a platform which will manage all PII exchanges between a data subject or controller of a data subjects' personal data and private or public entities requesting that data, also taking a role of a data controller or data processor within the PoSeID-on system. Data subjects will be able to manage and monitor information regarding the state of their personal data, through a user friendly and usability focused web dashboard, which should allow them to track exchanges of their PII, manage permissions of access to their data and view risk levels stemming from their privacy exposure and system operations. This dashboard will only be available through secure authentication via electronic Identification (eID) accounts in line with the electronic IDentification, Authentication and trust Services (eIDAS) Regulation in order to reduce identity frauds and protect the privacy of users.

The PoSeID-on proposal envisages a solution leveraging permissioned blockchain technologies and smart contracts. Through the use of smart contracts for PII management data confidentiality, access control and transparency of operations will be guaranteed for data subjects. Data controllers and data processors will only be able to access PII when properly authorized through a request using this technology. Permissioned blockchain technologies will provide a tamper resistant way to manage and record PII accesses and exchanges in a secure and distributed way while removing the possibility of unauthorized third parties monitoring or eavesdropping on data exchanges. No PII will be stored in blockchain transactions or smart contracts, only anonymized references to it, respecting GDPR data minimization principles and the right to be forgotten.

Figure 1.1 shows the original PoSeID-on contextual architecture, as described and presented in the projects description of work (DoW), identifying the major components and actors (data subjects and data processors) of the platform.

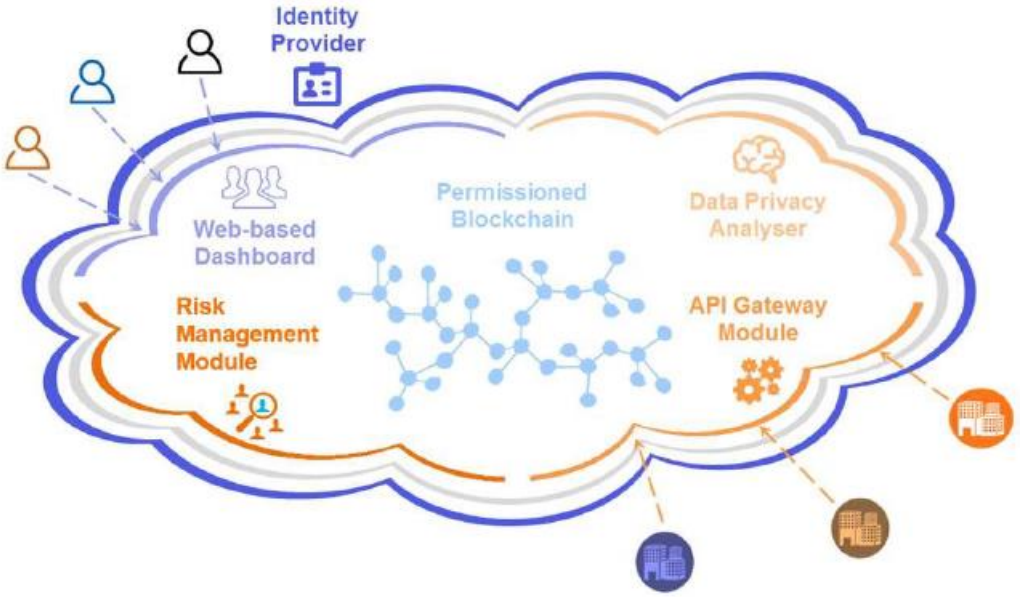

Figure 1.1 PoSeID-on original architecture proposal

While this original architecture is clearly not sufficiently well-defined to fully understand the PoSeID-on concept, it is what was available at the start of the project and at the start of this MSc Thesis. For the RMM module, the initial vision was that, somehow, a specialized module would be able to receive information from the other platform components (e.g. logs from data subjects accessing the dashboard, logs from data processors exchanging PII) and, based on those logs, would be able to detect anomalies reflecting abusive usage of the platform (e.g. illegitimate syphoning of user data, abnormal activity of data processors).

The results of the subsequent work detailing of the PoSeID-on architecture and components will be provided in Chapter 3 (PoSeID-on Architecture).

1.3.3 Consortium Partners

The PoSeID-on project combines the expertise of ten partners from seven European countries, which can be seen in Table 1.1. This subsection aims to present them and give an overview of each partner's role in the project.

The Consortium consists of two public administrations (MEF, SAN), three large industry IT companies (ACN, PNO and SOFT), one public and private IT provider (BRZ), one legal small-medium enterprise (ELEX) and two research and university institutes (TECN, UC). Reflecting the different profiles of each partner, most of the technical ICT work was performed by three partners: Tecnalia (blockchain tools), JIBE (dashboard and leadership of overall integration), and UC (architecture design, RMM and Data Privacy Analyzer).

Partner No	Organization name	Short name	Country
1 (Coordinator)	Ministero dell'Economia e delle Finanze	MEF	IT
2	Accenture S.p.A.	ACN	IT

3	PNO Innovation	PNO	BE
4	e-Lex Studio Legale	ELEX	IT
5	Fundacion Tecnalia Research & Innovation	TECN	SP
6	Ayuntamiento de Santander	SAN	SP
7	Softeam	SOFT	FR
8	Universidade de Coimbra	UC	PT
9	Bundesrechenzentrum GmbH	BRZ	AU
10	SMARTFEEDZ B.V.	JIBE	NL

Table 1.1 Project Partners

1.4 Thesis Contributions

This work developed for this thesis contributed to the anomaly detection state-of-the-art by providing a design and proof-of-concept implementation for a scalable, on-line anomaly detection module using PoSeID-on's logs (from the system and from PII transactions) for data subject risk assessment. Moreover, it contributed to the PoSeID-on project not only with the RMM module but also with additional efforts in the definition of the architecture of the whole platform.

The main contributions resulting from this work are:

1. A review of GDPR, blockchain technology and anomaly detection concepts and approaches.
2. The architectural design for a horizontally scalable module for online anomaly detection based on logs and system and PII transactions (the RMM Module).
3. Implementation of a proof-of-concept module based on the proposed architectural design.
4. A collection of scripts for building, containerizing and deploying the RMM module in Kubernetes, as well as the module dependencies, such as storage containers.
5. Implementation of the Drain Log Parser algorithm in Java, based on the original python implementation. Some quality of life adaptations to the algorithm were also developed to accommodate the use of Apache Spark Streaming [4] and log auditing.
6. A preliminary evaluation of the RMM module, based on synthetic datasets, to be later complemented with a more complete evaluation in the scope of the Pilots.

In addition, the following secondary outcomes of this work are noteworthy:

1. Documentation of PoSeID-on's System Requirements and Architecture, a joint work from several project partners that was led by the University of Coimbra and resulted in Deliverable 2.2 of the PoSeID-on project, led by the author of this Thesis [Annex 1].
2. The documentation of the Risk Management Module Interim implementation, that is a large part of Deliverable 4.3 of the PoseID-on project[5] .

3. Co-authorship of a journal paper describing the PoSeID-on's concept and initial architecture design, accepted for publication in the Journal of Data Protection & Privacy[6].
4. Co-authorship of a dissemination paper describing the PoSeID-on concept and focused on work conducted by University (the RMM and the Personal Data Analyzer), accepted for publication in the 9th "Congresso Luso-Moçambicano de Engenharia" (CLME2020) [7].
5. Co-authorship of a technical paper describing the PoSeID-on concept and focused on work conducted by University (the RMM and the Personal Data Analyzer), submitted to the Annual Privacy Forum 2020 (under review)[8].
6. Invited presentation in the public "Workshop on Privacy, Data Protection and Digital Identity", organized by the PoSeID-on Project, Coimbra, July 11, 2019.

Furthermore, the work developed during this thesis (wh[9]ich will continue in the following months) is expected to result in another research paper, focused specifically on the RMM solution.

1.5 Structure

The rest of this document is organized as follows:

- Section 1 is this introduction.
- Section 2 presents the background for the work presented in this thesis. It gives an overview of the General Data Protection Regulation, followed by a overview of blockchain technologies basic concepts and anomaly detection approaches.
- Section 3 presents the PoSeID-on project architecture and use cases.
- Section 4 details the Risk Management Module proposed design and specification, starting with the methodology that led to the initial proposal, followed by the anomaly detection approach taken and architectural design decisions.
- Section 5 describes the implementation and integration efforts performed during the development of this work, describing the technological stack and implemented components and deployment scripts. Finally, challenges faced during implementation are discussed.
- Section 6 evaluates the module and thesis results in face of the initial requirements and objectives and presents the experimental methodology and results achieved for the prototype implementation of the Risk Management Module. Publications which resulted from the work developed are also presented followed by the fulfilled thesis objectives.
- Section 7 presents the conclusions taken from the developed work and presents the future work for the RMM module.

Chapter 2 Background

The GDPR [7] is a regulation with the goal of harmonizing data privacy laws across Europe. It builds upon the European Data Protection Directive (DPD)'s concepts and principles, introduced in 1995, and aims at increasing the protection legislation provides to individuals regarding their personal data. To better understand the GDPR, without diving into the extensive details present in the regulation, it is helpful to identify what it applies to, who it affects, and what protections it aims to provide [9].

2.1 General Data Protection Regulation

The GDPR [7] is a regulation with the goal of harmonizing data privacy laws across Europe. It builds upon the European Data Protection Directive (DPD)'s concepts and principles, introduced in 1995, and aims at increasing the protection legislation provides to individuals regarding their personal data. To better understand the GDPR, without diving into the extensive details present in the regulation, it is helpful to identify what it applies to, who it affects, and what protections it aims to provide [7].

What does GDPR apply to?

GDPR has an over-arching scope over all material involving personal data of data subjects in the EU. Personal data is defined in Article 4 of GDPR as:

> *"any information relating to an identified or identifiable natural person"* [7]

This definition implies that the GDPR applies to not only data that is directly and obviously related to a natural person, such as a name or address, but also to any other data that can potentially be used to directly or indirectly identify someone. An example is logged metadata regarding someone's behavior on an online platform. For example: if it is possible to trace the operational logs back to a person, by connecting the log to an account number and that account number to a natural person, this is also considered personal data.

Who does it concern?

GDPR can be understood as set of rules defining the interactions between three main actors:

- **Data subject** is whom the data relates to, and the actor GDPR aims to protect.
- **Data controller** is defined in as the "natural or legal person, public authority, agency or other body which, alone or jointly with others, determines the purpose and means of the processing of personal data" [7]. It is the actor who is ultimately accountable for compliance and liable if the regulation is breached. It is also responsible of exercising the data subject's rights. It must always be possible to identify the data controller.
- **Data processor** is a party who "processes personal data on the behalf of the controller"[7]. The data processors have a set of obligations under the GDPR, but accountability falls onto the data controller governing them.

Principles, rights and obligations

In order to pave the way for the protection of personal data rules, GDPR sets the foundation of six data processing principles:

- **Lawfulness, fairness and transparency** - personal data should be "processed lawfully, fairly and in a transparent manner"[7]. Data controllers must clearly communicate the intended use of the data and ensure collection and processing of this data is done according to the legislation.
- **Purpose limitation** - personal data shall be "collected for specified, explicit and legitimate purposes"[7] meaning that any use of data for other purposes than those clearly identified upon data collection is against the law. Exceptions to this are archiving purposes for public interest or scientific and historical research.
- **Data minimization** - personal data collected should be "adequate, relevant and limited to what is necessary"[7] for the stated purposes.
- **Storage limitation** - personal data must be "kept in a form which permits identification of data subjects for no longer than is necessary"[7] for the stated purposes. Exceptions to this are archiving purposes for public interest or scientific and historical research.
- **Accuracy** - personal data kept by a data controller must be "accurate and, where necessary, kept up to date"[10]. This translates into the data subject's right of asking for correction of their personal data, correction which must be met by the data controller, by either correcting or deleting the no longer valid data.
- **Integrity and confidentiality** - personal data must be "processed in a manner that ensures appropriate security"[11], being the data controller's responsibility of ensuring that the data collected is secure against attacks, loss, destruction or damage.

GDPR also defines processing as lawful when it is done with the data subjects' consent, freely given, specific, informed and unambiguous. This consent can be later revoked as the data subject has the right to withdraw his or her consent at any time. Data controllers do not always need to comply to the revocation and subsequent deletion of the personal data if it is necessary for fulfilment of a contract, protection of vital interests of the data subject or other party, or for the performance of a task carried out in the public interest or legitimate interest pursued by the controller (if these do not go against the interests and fundamental rights and freedoms of the data subject).

From the principles above the GDPR derives a set of rights for data subjects and obligations for data controllers. In the perspective of general system design and development, and in particular from the viewpoint of the PoSeID-on project and architecture design, the following data subject rights are particularly relevant:

- **Right to rectification** - data subjects have the right to correct and keep their data updated.
- **Right to be forgotten** - data subjects have the right of having their personal data deleted if it is no longer necessary to fulfil the purpose it was requested for.
- **Right of access** - data subjects have the right to request information regarding if their data is being stored or processed, how and where.

- **Rights related to automated processing** - data subjects have the right to not be subject to a decision based solely on automated processing, including profiling, which produces legal or other significant effects.

Data controllers have the general obligations of exercising the rights of the data subjects, specially the obligation of processing personal data in a lawful way. Besides this, in the context of PoSeID-on's platform, there are two relevant obligations:

- **Data protection by design** - data controllers should design and implement their systems taking in consideration the compliance with GDPR.
- **Security of personal data** - data controllers should guarantee the security of the personal data they hold, to reasonable extents.

Data controllers must be also responsible for (and able to) demonstrate compliance with the GDPR principles we above described. The non-compliance results in fines that can reach 20 million euro and/or 4 percent of a company's worldwide annual turnover.

While GDPR received a lot of attention from regulators, data processors and the public in general, up to now the way GDPR is enforced is in general very rudimentary: most data processors have no automated channels to interface with data subjects or regulators, there is no common control of PII exchange between data processors, and data subjects have to rely in a multitude of ad-hoc interfaces for contacting all the data processors handling their PII. Even in the cases where data processor's information systems are already able to handle GDPR requirements, there is no harmonized mechanism offered to data subjects or regulators. PoSeID-on focuses precisely on this gap, by providing a general framework for the relation between data subjects and data processors, that offers a common dashboard to data subjects (which they can use for all data processors holding their PII) and transaction registration mechanisms able to register all PII transactions between different data processors. Among other technical solutions, PoSeID-on explicitly assumes blockchain as the core technology for providing a distributed and secure mechanism for registering data subject's consent and specific request, data processors response to those requests and exchange of PII between data subjects and/or data providers – in a safe, privacy-enabled and repudiation-proof manner.

2.2 Blockchain

Blockchain is the common denomination for fully distributed digital ledgers that enforce a tamper-resistant and tamper-evident implementation, usually without having a central authority (i.e. bank, company or government) governing over them. The blockchain allows for a community of users sharing the same ledger to record transactions in a distributed manner which prevents, in theory and under normal operation, transactions from being changed once committed to the chain. It became widely known when in 2008 it was combined with several other technologies and computing concepts to propose the first cryptocurrency, Bitcoin[11]. The novelty of this technology was the fact that it was a form of electronic cash protected through cryptographic mechanisms instead of a central repository or authority.

In 2009 the Bitcoin network launched with success and became widely known and the staple in cryptocurrency as it is possible to still see today. Bitcoin users were provided with a way to transfer digital information representing electronic cash through a distributed system.

Common to all cryptocurrencies available today, users are able to digitally sign and transfer rights to that information to other users and this is recorded publicly, allowing all participants of the network to verify the validity of the transactions without the need of central authority. Through the use of cryptographic mechanisms which guarantee that consensus in the network a distributed group of participants maintain and manage the blockchain making it resilient to attempts to tamper with the ledgers state, be it by altering previous transactions or forging new ones. Although blockchain is strongly associated with Bitcoin and cryptocurrencies in general due to the popularity it gained over the years the blockchain technology has a variety of applications and new applications are being researched in a variety of sectors, such is the case of the PoSeID-on project in which this thesis is inserted.

Blockchain technologies rely on several components, cryptographic primitives and distributed systems mechanisms which detailed exploration and description is beyond the scope of this thesis. It is possible however to describe each component in a simple way in order to help understand the blockchain system as a whole. The following is an informal definition of blockchains which conveys the focal points of the technology:

"Blockchains are distributed digital ledgers of cryptographically signed transactions that are grouped into blocks. Each block is cryptographically linked to the previous one (making it tamper evident) after validation and undergoing a consensus decision. As new blocks are added, older blocks become more difficult to modify (creating tamper resistance). New blocks are replicated across copies of the ledger within the network, and any conflicts are resolved automatically using established rules"[12].

Many electronic currency schemes were implemented before Bitcoin (e.g. ecash and NetCash, but they did not achieve widespread use. The main appeal of the Bitcoin which fostered its growth and gain in popularity was the use of the fully distributed blockchain which allowed no single user to control the electronic currency and prevented a single point of failure of the network. This, in addition to allowing direct transactions between users without the need of a trusted third party promoted its widespread use. Bitcoin's popularity is also associated with the fact that it provided users who contributed to main the network the possibility of earning more cryptocurrency if they were able to publish new blocks and maintain copies of the ledger. These users, commonly known as *miners* were able to, in a distributed manner, manage the network without the need for organization as the automated rewards for doing so is enough to build a solid network of *miners*. The cryptographic primitives and self-policing mechanisms on which the blockchain builds upon are enough to guarantee consensus in the network's administration and endurance that only valid transaction blocks are added to the blockchain.

In Bitcoin, the blockchain allowed users to be pseudonymous, effectively anonymizing users but not their account identifiers. This paired with the transparency of transactions has effectively allowed the creation of accounts without any identification or authorization process and provided pseudo-anonymity to users. As such the bitcoin network is an environment where users do not know others identity. In order to create trust between users without the use of a trusted third party, the blockchain provides the following four characteristics:

- **Ledger** - append only ledger which provides full transactional history. Transactions and values in a blockchain are never overridden unlike traditional databases.

- **Secure** - security is achieved through proven cryptographically secure methods, which ensure both the integrity and verifiability of the entries in the ledger.
- **Shared** - the ledger is publicly shared amongst multiple participants, providing transparency across the network participants.
- **Distributed** - being a fully distributed technology, scaling of the number of nodes participating in the network management bolsters its resilience to attacks by malicious nodes.

These characteristics effectively allow the anonymous creation and participation of accounts in the blockchain network while still delivering trust between parties with no prior knowledge of one another. Without the need of a third party to validate transactions blockchain networks may achieve faster and cheaper transactions between participants. This is the case for the Bitcoin and following cryptocurrencies following the same model of anonymous account creation. Blockchains which leverage this model are called permissionless blockchain networks, however it is also possible to more tightly control access to the network, provided that some degree of trust is shared among users. These are called permissioned blockchain networks.

2.2.1 Blockchain Categories

It is important to make a distinction between blockchain permission models as they directly impact the interactions between users and the blockchain network. It also impacts blockchain components and the interaction flow between them. In summary, if the blockchain is open for anyone to publish new blocks it is *permissionless*. If only particular users can publish blocks, it is *permissioned*. Permissionless blockchain networks can be compared to a controlled corporate intranet, while a permissioned blockchain network is like the public internet, where anyone can participate [13].

Permissionless – blockchain networks are decentralized ledger platforms where anyone can publish blocks without need of approval from a trusted authority. Similarly, reading the blockchain and issuing transactions is also open to anyone. Since anyone can read and write to the ledger in order to prevent malicious users from tampering with the existing records by introducing malicious transactions, mechanisms are implemented in order to guarantee multiparty agreement. This is generally called "consensus" mechanisms in the blockchain. These mechanisms usually require users running blockchain nodes to spend computational resources when publishing blocks or to maintain a certain amount of cryptocurrency in order to prevent attacks [14]. This is the general permission model used by well-known cryptocurrencies and blockchain implementations such as Bitcoin and Ethereum[11].

Permissioned – blockchain networks are controlled by an authority that can be decentralized or not. As such only authorized users are able to maintain and publish to the blockchain. Access can also be controlled in order to allow anyone to read from the chain or just authorized users. It is possible for permissioned blockchains to have the same traceability and transparency of digital assets and transactions in the blockchain network and be equally distributed, with resilient and redundant data storage. Since trust is controlled and established through an authority, users can be held accountable in case of malicious behaviour. This results in less restrictive consensus mechanisms which generally makes permissioned blockchains less computationally expensive to maintain and transactions and operations are in most cases faster than in permissionless counterparts.

Besides providing trust and accountability to parties participating in the blockchain network, some permissioned blockchains also provide the ability of only revealing transactions to users which have been previously identified and given the right credentials. In this manner the content of transactions or even the existence of a transaction can be hidden from parties not involved in them. Some frameworks that currently leverage these capabilities are Hyperledger Fabric [11] and Quorum's Ethereum-based protocol [11].

2.2.2 Overview of Blockchain Components

In order to understand the role of blockchain within the PoSeID-on system, a general understanding of certain blockchain components is necessary. This section will provide a general overview of these components and concepts. Technical detail regarding consensus mechanisms, hashing functions, and cryptographic mechanisms behind blockchain technologies can be extensively found in recent literature (for instance [15]) and as such will not be detailed in this section.

Transactions

A *transaction* represents an interaction between participants in the blockchain. Transactions usually represent records of activities regarding an asset, physical or digital. For example, in cryptocurrency, such as Bitcoin, a transaction is an exchange of currency but blockchain is not limited to that. In other scenarios such as property markets it can represent a transfer of property.

A transaction usually includes the sender's address or other relevant identifier, his public key, a digital signature and transaction inputs and outputs:

Inputs are most commonly the digital assets to be transferred. The transaction will contain a reference to the source of those assets which is either the previous transaction regarding said assets or the origin event (the transaction representing the creation of an asset) if the asset is new and has never been transferred before. Assets, *per se*, do not change, they are only referenced by new transactions as an input, it is impossible to add or remove assets. For example, in cryptocurrency, if you have a transaction representing a quantity of currency you hold, you cannot change the asset itself, but you can instead split or combine it with other assets by referencing them as input to a new transaction. Of course, this implies that the creator of new transactions has ownership and access to the referenced inputs, usually proven by the use of a private key to sign the new transaction.

Outputs are usually the addresses or identifiers of the accounts which are the recipients of the asset and, in case of cryptocurrencies, the amount being transferred. If the amount being transferred is less than the asset used as input, an extra transaction is created with the sender of the asset as recipient.

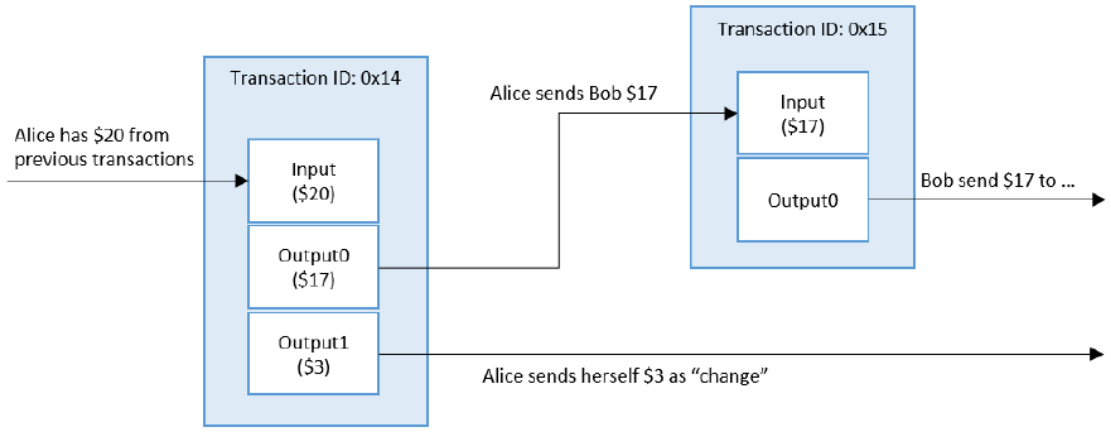

Figure 2.1 Example of Transaction (from [16])

While the most common use of transactions in blockchain based systems is to transfer digital assets, transactions can be used to transfer generic data or post something publicly online in a permanent way by adding it to the payload of a transaction. On blockchain systems which support smart contracts, transactions can be used to send data to smart contracts for processing and can be used as a means of storing results of that processing on the blockchain. For example, in case of a smart contract leveraging system governing access to a building, a transaction could be sent when a user tries to open a room with his credentials and if they check out, have the smart contract create a new transaction on the blockchain recording his access to that room.

Ledger

Ledgers are tools to keep track of exchanged goods and services, from physical pen and paper, to digital databases, usually owned and operated by centralized trusted third parties on behalf of a community of users.

A *ledger* is a collection of transactions. The approach taken by blockchain technology allows for a distributed ownership and distributed physical architecture, providing increased reliability, security and trust when compared to ledgers with a centralized ownership. The main advantages of having a decentralized ledger using blockchain technology are the following:

- **Distribution by design** allows for several backup copies, all synced between peers. Some blockchain implementations can even support private transactions or channels, without taking away the advantage of having a distributed ledger, as it is possible to verify the existence of such transactions without revealing the actual content or participants of these transactions.
- **Increase security against attacks** as it is possible to have heterogeneous software, hardware and infrastructures maintain the nodes and ledger. This means that an attack on one node is not guaranteed to work on another, increasing security and reliability. Furthermore, by having distributed validation of new blocks a node which has been compromised cannot alter the state of the ledger without the rest of the network knowing and validating these changes.

- **Increased resilience and availability** by allowing geographically dispersed nodes and supporting the loss of a node or even a region of nodes.
- **Transparency and tamper resistance** which comes from having distributed ownership of ledgers and ledger validation. This guarantees that a single entity cannot commit malicious blocks without other nodes verifying and validating them first.

Smart Contracts

The term smart contract comes from the definition given by Nick Szabo in 1994:

"*A smart contract is a computerized transaction protocol that executes the terms of a contract. The general objectives of smart contract design are to satisfy common contractual conditions (such as payment terms, liens, confidentiality, and even enforcement), minimize exceptions both malicious and accidental, and minimize the need for trusted intermediaries*" [17].

Smart Contracts are a collection of code and data deployed using cryptographically signed transactions on the blockchain network. They are executed by nodes participating in the blockchain network. As they are deployed on the blockchain, they are effectively tamper-evident and tamper-proof and, as such, can be considered a trusted third party and can be used for any purpose which requires verifiable calculations, information storage or other operations which state needs to be publicly attestable. Smart contracts must also be deterministic, given the same input they should always produce the same output. Similarly, to regular transactions, all nodes must approve on the new state produced from the execution of the smart contract code. As such smart contracts can only operate on data directly passed as parameter.

Depending on the implementation, smart contracts can be executed simultaneously with the publication of new blocks (i.e. Ethereum) or they can be executed by separate nodes which send the results to publisher nodes which validate and publish the output based on several executing nodes results (i.e. Hyperledger Fabric).

2.3 Anomaly Detection

Anomaly detection is an important area of data mining research that involves discovering anomalous or abnormal data in a given dataset. It is an area widely studied in statistics and machine learning and it is also termed as outlier detection, novelty detection, deviation detection and exception mining [18].

To understand anomaly detection, it is first necessary to understand what an anomaly is. For this we can look at the widely accepted definition given by Hawkins [19]:

"An anomaly is an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism"

As Hawkins explains in his definition, detection of anomalies is relevant as anomalies are significant but rare events that can indicate the need of critical actions in the domain in which they are detected. Anomaly detection has been successfully applied to several different domains, such as public health, fraud detection, industrial damage, image processing, sensor networks, robot behavior and astronomical data [20], but a large part of available literature addresses network anomaly detection using Network Intrusion Detection Systems (NIDS),

as with the evolution of computer networks and the increase in the availability of hacking and cracking tools internet security has become a recurring topic and the interest in machine learning techniques for network intrusion detection has grown significantly.

NIDSs can be classified according to their style of detection as either [18][21]:

- **misuse-detection** - systems which monitor activity with precise descriptions of know malicious behavior.
- **anomaly-detection** - system which form a view of normal activity and identify deviations from that profile.

The latter are interesting in the scope of this thesis, as what we are setting out to implement is similar to an intrusion detection system, identifying possible attacks or patterns which may indicate risk of PII leakage or misuse. It is also important to note that anomaly detection systems are *not* targeting to identify malicious behavior but any pattern at all that has not been observed before, be it malicious or not.

There are a wide variety of learning techniques for anomaly detection, proposed for different NIDS problems. These can be broadly categorized in three groups: supervised, semi-supervised and unsupervised. These groups will be explained further in the subsection "Learning groups".

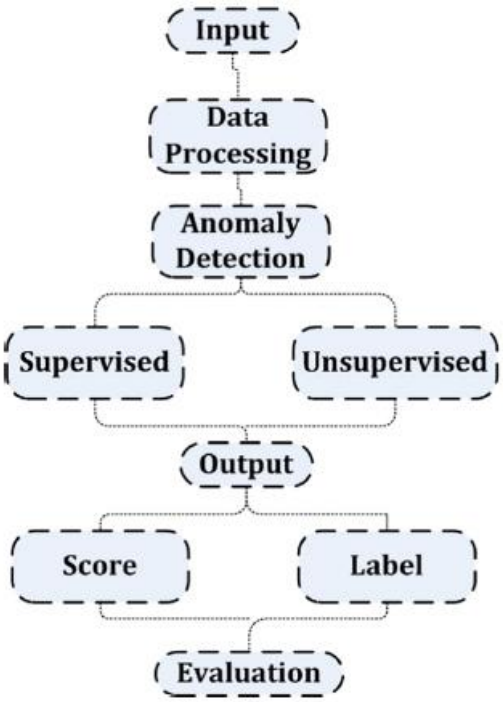

Figure 2.2 Generic framework for anomaly detection [22]

Figure 2.2 shows the generic framework used for network anomaly detection systems. First, input data is processed as the data arriving at the anomaly detector is usually from different types of sources and comes in different formats, e.g. measurements from sensors, network logs or operational logs from a system module.

This processing is usually associated to the specific anomaly detection techniques employed by the anomaly detector, as data must be in an acceptable format in order to be analyzed by each model.

After data is processed and analyzed by each model, the output of the analysis is evaluated, either by the means of a score or a label, each with their respective advantages and disadvantages, which will be further explained in the following sections.

Types of Anomaly

Anomalies are patterns which do not conform to well-defined characteristics of normal data patterns for a given domain or system. They can represent several abnormal activities and can range from harmless but not recognized behavior to critical threats such as cyber-attacks or component failures.

Ahmed et al. categorized anomalies in three different categories[23][24]:

- **Point/Spatial anomaly**: When a particular instance of data deviates from the normal pattern of the dataset, i.e. when a person usual makes 3 calls a day but suddenly data shows 30 calls made on the same day.
- **Contextual/Temporal anomaly**: When a particular instance deviates from the normal pattern in a given context, i.e. if an outdoors thermostat reports 0 degrees Celsius in the Winter in Portugal, it may not be anomalous due to the usual decrease in temperatures, but if it reports the same value during the Summer, it is a strong indicator that the thermostat might not be working properly.
- **Collective anomaly**: When a collection of similar data instances deviates from the normal pattern of the entire dataset, i.e. a lost packet by itself might not indicate a problem in a server but if packets are lost for a continuously long period of time, it may indicate the server has gone down or we are in the presence of a network attack.

Output evaluation: Score vs Label

The output of the anomaly detection techniques is usually the data instances which were identified as anomalies and their respective classification. The classification of these instances is usually represented in two ways[23]:

- **Scores** - Each data instance is assigned a score indicating the severity of the anomaly. These scores are then ranked for an analyst to select from or are selected by a previously identified threshold.
- **Binary/Label** - Each data instance is assigned a binary label, such as anomalous or normal.

Binary labels are computationally more efficient than scores since the latter do not need to be calculated for each data instance but in several cases, such as in unsupervised learning, scoring can help partitioning the identified anomaly data set and provide valuable information for further retraining since in unsupervised learning we do not have a labelled training data set and making a distinction between real anomalies and normal behavior is in most cases non-binary, with data instances varying in probability of being anomalies or not.

Learning groups

Anomaly detection techniques require some sort of learning from historical data in order to be able to reason over identified patterns and classify them as anomalous or normal. According to the methods used to build a prediction model of the data under study, anomaly

detection techniques can be split into three learning groups: *Supervised*, *Semi-supervised* and *Unsupervised* learning.

Supervised learning requires some previous knowledge provided by an external source, such as labelled data for both normal and anomalous classes. In well know domains these techniques can be extremely good at detecting well known anomalous patterns (i.e. DDoS attacks, distributed network scanning, ping flooding, etc.) but suffer against unknown patterns and zero-day vulnerabilities. This can be mitigated to an extent by having an architecture which allows re-training of working models with manual input provided by security analysts which manually identify new vulnerabilities or anomalous patterns and insert them into the system, but this does not scale well into large systems or networks. A typical approach for supervised training involves building a predictive model against a training dataset, contained aforementioned data instances labelled as normal and anomalous and then using another testing dataset in order to evaluate the model's performance[25].

Supervised learning faces two major practical challenges. First, in general, anomalous instances are in great lesser number than the normal instances of the dataset. Issues which occur due to imbalanced class distributions have already been addressed by a number of data mining and machine learning literature[26][27][28]. Second as explained earlier, it is difficult to obtain accurate and representative labels for anomalies that are not known in advance.

Semi-supervised learning usually involves building a model for the class corresponding to normal behavior and using the model to identify anomalies in the data under study. There are also some approaches which are trained with only the anomalous data instances instead, but these are not commonly used since it is difficult to obtain a training set with all of the possible anomalous behavior.

Since these techniques do not rely on both classes of data instances, they are applicable to situations where no data regarding attacks or anomalies is available. Due to this fact, they are also frequently called novelty detection, since no anomalous data was used for training.

Unsupervised learning anomaly detection techniques do not require training data and as such are more widely applicable to new domains.

The frequent approach to these techniques consists in analyzing the datasets from all scenarios and sources combined and labelled respectively. Then a sample of the dataset is selected as training data this sample is used to build a model according to the technique used. Another sample is selected from the remaining data instances (excluding the previously selected for training), is selected to serve as test data for cross-validation of the model.

Unsupervised techniques operate on the implicit assumption that the dataset used contains a much greater number of normal instances than anomalous ones. If that is not the case, then the model will report a high number of false positives as it will consider normal instances as possible anomalies.

Many semi-supervised techniques can be adapted to operate in unsupervised mode by using a sample of the unlabelled dataset as training data, provided that this dataset contains a small number of anomalies and the resulting model correctly identifies these few instances as anomalies.

Unsupervised techniques usually fall into two categories, *clustering* and *nearest-neighbour* techniques.

Clustering assumes that data is clustered, as in normal data can be separated into different clusters while anomalies appear as outliers and either fit in a small cluster or none at all. Frequently, unsupervised learning is simply referred as clustering.

Nearest-neighbor assumes that new anomalies are closer to known anomalies and new data instances are evaluated according to the distance to other anomalies or density of anomalies near them.

Ensemble Learning

The ensemble learning theory consists in combining multiple single machine learning models in order to improve the model's predictions[29][30]. Using multiple learning algorithms, it aims to obtain a better predictive performance than any single constituent algorithm by itself. In theory, if no single model can cover the true prediction behind the data, an ensemble can give a better approximation of the true prediction model. In addition, an ensemble of models displays higher robustness when facing uncertainties in training data[31].

Instead of deciding on the best model to explain the data, ensemble techniques focus on constructing a set of models and then deciding between them with some combinatorial approach, in order for each model to complement each other and compensate for their individual limitations.

As such, ensemble learning has a higher computational cost and complexity since it involves training and applying multiple models to the data under analysis but achieves a better diversity of predictions.

There are several approaches to ensemble learning, for supervised, semi-supervised and unsupervised learning, being the most common supervised approaches bagging[32] ,boosting[33]and stacking[34]

Bagging short for Bootstrap Aggregation, generates additional training data from the original dataset in order to decrease the variance of the prediction model. Using random subsets of the training set, bagging trains each model in the ensemble, which are then combined in an equal-weight majority voting mechanism. This does not increase accuracy of the predictions but narrows the variance by strongly tuning the outcome.

Boosting involves building an ensemble by training each new model according to the performance of previous models. In this two-step approach, first a subset of the original data is used to produce multiple models, then these models are combined, also using majority voting in order to boost their performance.

Stacking consists in using several different base models in order to obtain a better understanding of the input data, as opposed to bagging and boosting, which normally use the same base model in all training steps. Whereas bagging and boosting usually rely on majority voting to decide over the base models results, stacking is the method which actually makes use of a meta learner. This learner uses the output of the base learners in order to explore the data inputs through the different properties of each model. In this way, since each particular model is capable of capturing some aspects of the data but not the whole picture, it is possible to obtain better results by taking this in consideration and stacking a meta learner on top of these base models. Stacking is less popular than bagging and boosting but recent literature [35]and results in model competitions[36] has demonstrated the capabilities and outstanding results of this technique.

Unlike supervised ensemble approaches, there are no theories behind the success of clustering ensemble approaches and these approaches face the challenge of label correspondence as relating clusters between different algorithms is not a trivial task. Experimental results show that cluster ensembles are better than single models in practice but there is no universally successful ensemble method.

Stream vs batch analysis

Depending on the nature of the system under analysis, there are two approaches that can be taken, in respect to the way in which the data processing is performed and how the dataset is provided to the anomaly detection system, these are:

- **Batch Processing** - consists in the processing of large volumes of data collected over large periods of time. It is the most common approach to data analysis and has been the standard for big data analysis in the past years, specially due to the announcement of Google's MapReduce programming paradigm for extremely scalable processing and generation of big data sets in a parallel and distributed manner\cite{MapReduceLAMMEL20081} and its counterpart open source implementation by the Apache foundation\cite{Hadoop}. Although it is a very efficient way to process large amounts of data it is not a good approach when analysis results are necessary in real-time.
- **Stream processing** - involves analyzing continuous sequences of data occurring in real-time. With the rise of Internet of Things (IoT) and growth of real-time data sources which stream large quantities of time-series data, stream analysis is becoming a growing trend and some works have been published exploring this field[37][33] [38][23]but there is still no over-arching reliable solution. Stream analysis is especially useful when the detection of anomalous behavior is necessary in real-time, such as in cyber-security or fraud detection. The main challenges with stream processing or stream analysis are that streaming data inherently exhibits concept drift, which means the statistical properties of the data inputs change over time in unforeseen ways, causing the predictions to become less accurate as time passes. Another problem is that as opposed to batch processing, the full dataset is not available.

Challenges in anomaly detection

Despite seeming straightforward, detecting data that does not follow the normal pattern in a given dataset is a complex task as patterns vary from domain to domain as well as data types under study and its dimensionality. As such the following challenges can be identified in current anomaly detection techniques[26]:

- There is no universal anomaly detection technique, selecting the best machine learning model for a particular problem is a complex task;
- Noisy data usually contain anomalies which can be difficult to correctly identify and separated from regular noise;
- Normal behaviors and patterns are continually evolving and as such current detection techniques may not be effective in long term.

Besides these three over-arching challenges across the anomaly detection field, depending on the domain of application of these techniques, other challenges can be identified. One of

particular interest for this thesis appears in network monitoring applications, requiring near real-time processing of very large amounts of data which is heterogeneous by nature.

Anomaly detection approaches

There are many approaches to the problem of anomaly detection using machine learning and it is explored extensively in literature. There are several surveys which give an overview of machine learning models used in network anomaly detection [33][24]papers which evaluate some of the proposed models in different scenarios, such as in detection of application anomalies and cellular Quality of Experience (QoE) prediction[26]and in detection of attacks using NIDSs[36][39] In general, the majority of available literature explores approaches using batch analysis in a supervised and semi-supervised manner, with recent papers tackling the challenge of stream analysis using both supervised[37][40] and unsupervised[36][41] learning. To summarize a few of these techniques, Table 2.1 was compiled in order to aid in understanding the current most common and some recently proposed algorithms for each learning category and processing approach.

This is by no means a complete list, it is a compilation based on the literature read during the development of this work. The absence of entries in one of the table's cells also does not mean that there are no solutions for that learning group and processing scenario intersection.

There are many more algorithms or techniques which can be used for anomaly detection, making the task of choosing the appropriate technique for the situation at hand an even more daunting task. As mentioned before, there are no one-size-fits-all solution and a technique that might work very well for one scenario might not achieve the same results at all in another. Therefor several algorithms should be selected and tested for each specific problem, as an attempt to achieve the best result. If technical and performance constraints allow it, ensembles can be used to further improve predictions.

Recent works have shown that for network analytics, supervised *decision trees* are computationally efficient and accurate and have favorable properties such as model visibility and comprehensiveness[42]. Still within the supervised learning field, *Stochastic Gradient Descent* and *Random Forests* (an ensemble of decision trees) have also been used in network anomaly detection in both stream and batch scenarios with the stream variants achieving results as good or better than the batch variants [37].

For the RMM specific problem, explained in Chapter 4 , a selection of unsupervised algorithms is the best starting approach as there is no training data available. Streaming and batch variants are also favorable, due to the architectural approach chosen, also detailed in the same chapter. Clustering algorithms *K-means* and *expectation-maximization* algorithm in the context of *Gaussian Mixture Models* (GMM) are well known and tested algorithms, with variants used in both stream[37][1] and batch processing[43] and are good candidates for this scenario.

Learning Group	Batch	Stream
Supervised	Logistic and Lasso Regression SVM Decision Trees k-Nearest Neighbours Multi-layer Perceptron Naïve Bayes Stochastic Gradient Descent Hoeffding Trees PCA	Hough Hoeffding Adaptive Trees Incremental k-NN Stochastic Gradient Descent
Semi-Supervised	OCSVM k-Nearest Neighbours	Incremental k-NN
Unsupervised	Isolation Forest k-means clustering x-means clustering k-medoids	Hierarchical Temporal Memory Twitter's Anomaly Detection Etsy's Skyline Multinomial Relative Entropy EXPoSE Bayesian Online Changepoint detection Watchmen Anomaly Detection Incremental K-Means
Ensemble Supervised	Random Forest Decision Trees with Bagging and Boosting Stacking Majority Voting Stacking with GML	Adaptive Random Forests
Ensemble Unsupervised	Hypergraph Partitioning Majority Voting Mutual Information Co-association based functions Finite Mixture Model	

Table 2.1 Compilation of several algorithms used in anomaly detection

2.4 Log Anomaly Detection

Most anomaly detection algorithms require data to be structured in a way that the algorithm can comprehend and work it. Normally this is in the form of a set of numeric features described by a feature vector.

In order to build a feature vector to feed the prediction models, it is necessary then to convert our data sources into numeric features. In the case of the Risk Management Module, data arrives in the form of raw system and PII operation logs. In order to build a feature vector from this information, the generic framework used in log-based anomaly detection is applied. This consists of four main steps [12]:

- **Log Collection:** systems constantly produce a stream of logs describing system states and runtime information. These logs contain a set of parameters depending on the used logging framework, and a raw text log message describing the event which resulted in the respective log. These logs can be stored for later use or fed directly into a stream analyzer for real-time anomaly detection.

- **Log Parsing:** since logs consist of free form text, it is necessary to parse them and extract a group of structured event templates. The event templates consist of the most common combination of log parts/segments, which define the constant part of a log. Once a set of event templates is selected, each log can then be considered an occurrence of one of the selected events, with added specific parameters, which constitute the variable part of the log.

- **Feature Extraction:** once event templates are selected, the logs collection is then sliced into several sequences of log events, which are then encoded into numerical feature vectors. In log anomaly detection, feature vectors generally consist of a count of occurrences of a certain event within the sequence of log events being analyzed. These vectors can then be passed as input to machine learning models. There are several ways to slice the logs, such as using fixed windows, sliding windows or other types of grouping, depending on the domain and desired outcome. Fixed and Sliding Windows are based on time and timestamps. Each log has an associated timestamp corresponding to the time the log was issued. A fixed window is determined by a fixed time span or duration for each window. All logs occurring within a predefined duration (e.g., an hour) are regarded as a sequence and analyzed together. These windows are consecutive, the end of one window corresponding to the beginning of the next one. Sliding windows are also grouped by time, but in this case, there is also a sliding step, which is typically smaller than the window size. This step determines the movement of the window over time, as the beginning of a window corresponds to the beginning of the previous one, plus the step size or duration. In practice, this means that it is possible to have windows with overlapping logs (i.e. if we have a 1-hour window with a 10 minutes step, we will have consecutive windows where the only new logs being added on each slide are the logs received in those new 10 minutes). Logs occurring within each window are here too regarded as a sequence.

- **Anomaly Detection:** the last step consists of feeding the feature vectors created in the previous phase into machine learning models for training and construction of a model for anomaly prediction. The constructed model can then be used to determine if a new

incoming log sequence is considered an anomaly or not by analyzing the predicted value for each incoming log sequence.

A graphical example of this process can be seen in the Figure 2.3, provided next.

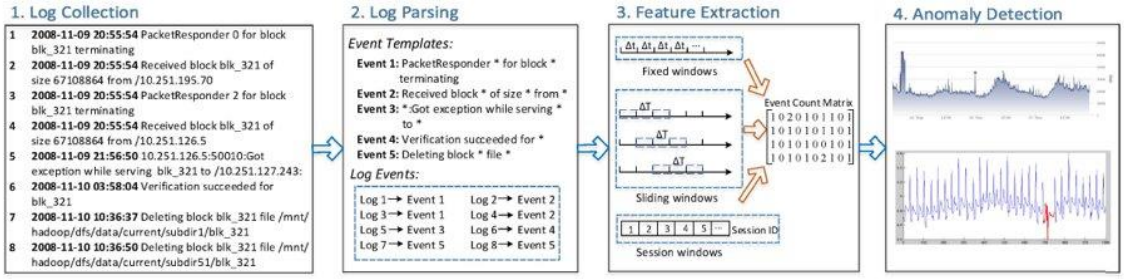

Figure 2.3 Graphical example of log anomaly detection [12]

2.5 Summary

This chapter presented the state-of-the-art regarding anomaly detection, as well as the several approaches that can be taken for anomaly detection, from supervised to unsupervised learning, usage of ensembles to optimize results and the differences between batch and stream analysis. Furthermore, an overview of anomaly detection techniques available for each approach was presented, chosen based on the results of several surveys and availability of implementation and frameworks. A brief overview on GDPR and blockchain was also presented. Together this creates a solid basis for understanding both the key background technologies of PoSeID-on's vision and the details necessary to understand the choice of frameworks, architecture and implementation options for the RMM, that will be discussed in Chapter 4.

Chapter 3 PoSeID-on Architecture

As already mentioned, the work associated with this thesis was performed in the scope of the project PoSeID-on [44]. In this project the University of Coimbra participates in several activities, including the leading of the platform specification and design, as well as the design and development of a couple specific components (including the RMM).

This section contextualizes the reader with a more technical perspective of the overall PoSeID-on platform, therefore complementing the project briefing already provided in Section 1. While the PoSeID-on architecture is provided here essentially for allowing the reader to better understand how the RMM fits into the platform, it should be noticed that the author of this thesis actively participated in the concept definition and architecture discussions that took part in the first months of the project, significantly contributing to its definition and leading the edition of PoSeID-on Deliverable D2.2 (Systems Requirements and Architecture [Annex 1]).

3.1 Envisioned solution

As explained in Chapter 1 , the PoSeID-on project aims at designing, implementing and validating a Privacy Enhancing Dashboard for personal data protection, a platform able to manage all PII exchanges between a data subject or controller of a data subjects' personal data and private or public entities which request it. Entities requesting access to PII assume the role, within PoSeID-on, of data processors.

The general architecture of the PoSeID-on system was the result of an iterative process, built upon the initial concept already presented in Chapter 1 in an interactive process, including the design of Use Cases for each of the 4 planned pilots, users requirements definition, system requirements definition, and system architecture design and specifications.

Figure 3.1 provides a simplified perspective of the platform architecture, identifying main components and main actors. Those actors, components and use cases are briefly summarized in the next sections.

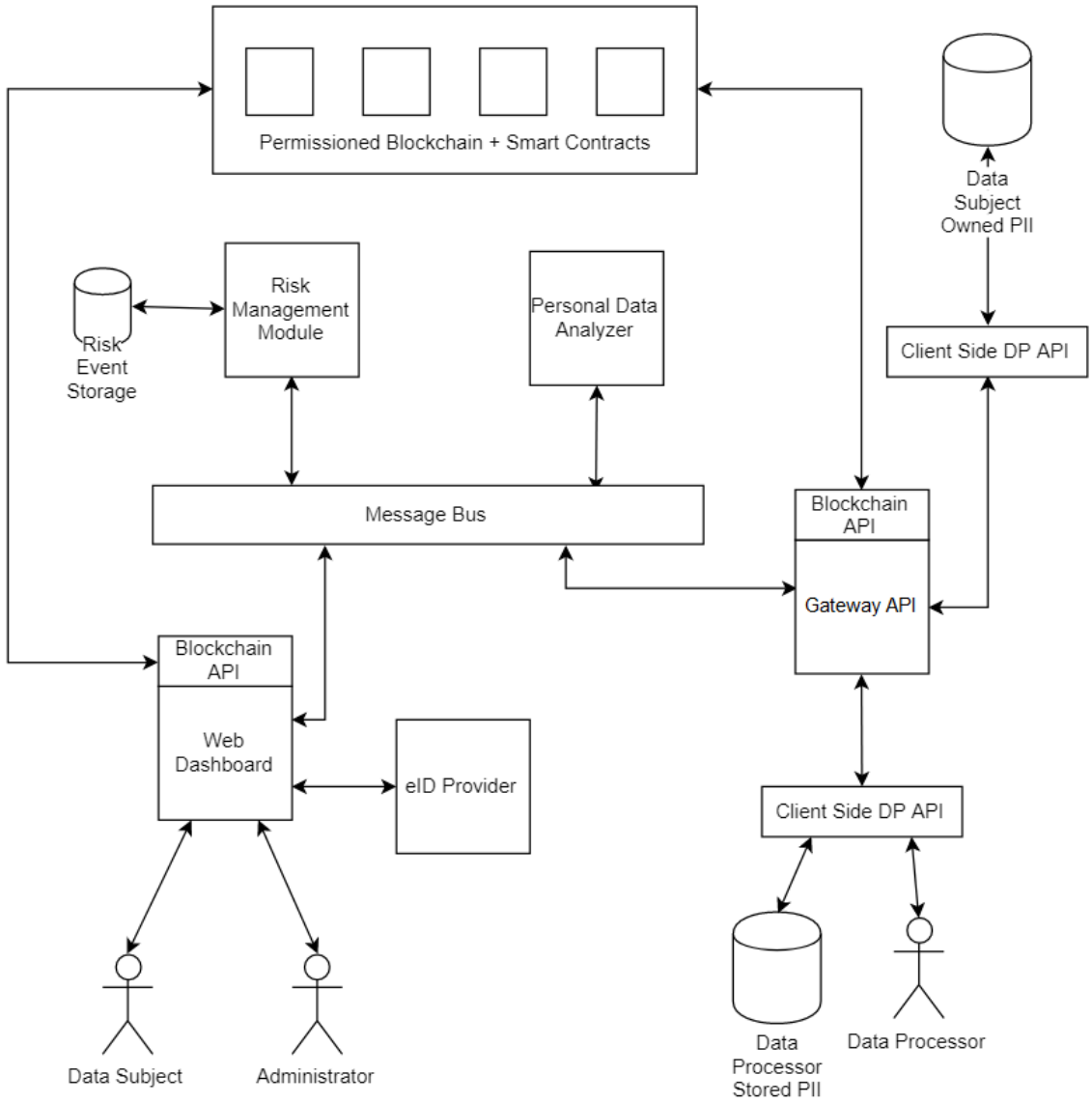

Figure 3.1 PoSeID-on architecture proposal

3.2 System actors

Data Subjects

Data subjects are people that represent the primary target of the GDPR. They represent the majority of PoSeID-on's users and will have their PII managed within the system. PII is a valuable resource for third-parties and its uncontrolled storage, processing and propagation represents a privacy risk for the data subject. PoSeID-on answers their need for having a single platform where it is possible to control and monitor PII accesses for the inevitable situations in which a data subject is required to share its PII with a third-party, such as banks, insurance companies, employers, airlines, health institutions or any other service that makes use of such information. These exchanges can be either direct, when the data subject directly provides the PII, or indirect, when one data processor directly provides PII to another data processor, with appropriate consent by the data subject. Examples of such sharing of PII are, for instance, tax services retrieving information from the subject's bank directly, or one retail

group proving a customer's PII to a marketing company for directed campaigns.Both these cases will be considered by PoSeID-on.

Data Processors

Data processors are third parties storing and exchanging PII with the data subject or between themselves. In the context of this project, this term encompasses the original designations given by GDPR for both data processors and data controllers (cf. section 0). Data processors will have multiple operations available with the data subjects PII, upon proper authorization (cf. section 3.4 for more details). PII storage mechanisms employed by data processors are outside the scope of PoSeID-on, which serves primarily as moderator and auditing tool, interfacing with the data processors' information systems but without directly managing data processor's data.

This means PoSeID-on does not claim to cover all the technical functionalities to ensure GDPR compliance (this is still the role of data processors' information systems), focusing instead on providing a means to mediate and register interactions between data subjects and data processors (e.g. provisioning of data, consent, requests for data correction or deletion) and between data subjects (e.g. sharing PII). Compliance with the GDPR is nevertheless mandatory and data subject's requests and permissions regarding retention, deletion and processing of their PI,I must be met by the data processors based to regulatory and legal obligations.

Administrators

Administrators are the operators managing the PoSeID-on platform. The platform is operated by a preassigned entity – typically a public or semi-public organization with the mission of providing a PII management framework such as PoSeID-on. While this entity is usually considered as a trusted third-party, the whole platform design tries to minimize the risk of intentional or accidental breaches of privacy originating from the management entity. Administrators manage the PoSeID-on platform on behalf of this entity.

3.3 System components

Web Dashboard

The dashboard is a Web interface through which administrators and data subjects access the PoSeID-on platform. It allows data subjects to grant, modify and revoke permissions for all or individual data processors, to check their PII access history and to check risk levels associated with PII shared with each data processor, as well as to receive notifications of possible privacy risks. Access to the web dashboard will be typically (though not always) based on the data subject's national eID or similar credentials. Administrators will be able to analyse system wide risks and check operational logs through this interface.

Data Processor API

This API is the access point for data processors communicating with PoSeID-on. It is essentially a secure API through which data processors may integrate PoSeID-on functionalities into their legacy information systems .

Client-side Data Processor API

This is the client-side interface for data processors to communicate with the Data Processor API. It communicates with the PoSeID-on system and accesses the Data Processor API, data

processors will have to provide an interface to their PII store in order to allow access to data subjects PII in an automated way and allow PoSeID-on to manage the transport of PII and revocation of permissions. The PoSeID-on project will also implement an example of this interface for a PoSeID-on PII storage which will store PII which does not have an authority controlling and providing it to the system. This storage will be a regular data processor for what concerns the rest of the system.

Permissioned Blockchain and Smart Contracts

Blockchain implementation specific to the PoSeID-on system. It is permissioned, transactions are anonymized, and PII exchanges within PoSeID-on are managed through the implementation of smart contracts specific for this purpose. These will describe the management of permissions and PII requests in order to grant, deny and check PII access permissions. The blockchain nodes will be hosted by the PoSeID-on administration entity and possibly other authorized entities (depending on deployment options from Use Case to Use Case).

Blockchain API

This interface translates blockchain operations into a high-level API suitable for integration into other applications. Since the use of Blockchain carries some important implications on how the clients and the servers behave, actions like account management (Data Processor and Data Subject identity on the Blockchain) or system functionality (Smart Contract functions usage) change drastically. For instance, users shall sign locally every call to a Smart Contract, but other components in the overall PoSeID-on architecture, like the Web Dashboard Module, will function as if communicating with a regular web service, despite the existence of a Blockchain network behind it.

The Blockchain API will also generate logs for modules communicating with the blockchain and send them to the Risk Management Module. The Blockchain API is not a web-service, but it will be directly used as a wrapper from the client browser, without intermediaries, providing a direct connection to the Blockchain Module.

Risk Management Module

This module is responsible for monitoring PoSeID-on, both from a system wide perspective and from the point of view of individual data subject's operations. RMM is expected to detect and evaluate possible security and privacy risks. For instance: anomalous behavior from a specific data processor which suddenly begins collecting much more data than usual, from a large set of data subjects (which means the data processor may have been hacked and being used to syphon PII to external attackers) or risks associated with a specific data subject (such as successive attempts to login with his/her credentials). Risk detection is made combining machine learning algorithms, that analyze multiple sources of information about transactions, user-level behavior and system-level behaviour. When the data subject provides explicit consent, transaction-specific data and PII may also be used for this analysis. High risk levels may trigger alerts to PoSeID-on administrators and Data Subjects – depending on the RMM settings.

Personal Data Analyzer

Monitors personal data transactions and warnings generated by the blockchain platform, to detect anomalies regarding PII exchanges. As opposed to the RMM, this module will analyze the personal data being exchanged with the aim of discovering non-identified PII, or PII not covered by the involved data processor's permissions. This component requires explicit

consent for operation, due to the sensitive nature of the data being analyzed, and all data will be discarded by the PDA after analysis is performed. It will also analyze data on demand from data processors, in order to identify possible PII contained in such data.

eID Provider

This provider authenticates data subjects and data processors on behalf of the PoSeID-on system, according to the European eIDAS regulations. Other authentication solutions can be used with PoSeID-on, if necessary, but this is option is the preferred one, mostly due to non-technical reasons.

Data Subject's PII Repository

This repository stores PII manually inserted by the data subject into the PoSeID-on system (using the dashboard), in order to provide this information to data processors. From a technical view this repository acts as a regular data processor within the system and operates within the same rules as general data processors. This guarantees that storage and access to PII within PoSeID-on cannot bypass existing privacy restrictions and integrates seamlessly with the rest of the system. This solution, which in practice means each data subject has its own personal data processor, was deemed as the most elegant to keep the PoSeID-on architecture simple and flexible.

Message Bus

This component provides the messaging infrastructure for PoSeID-on components to communicate with each other in a controlled but decoupled fashion, which will allow for the easy addition and removal of components without affecting the overall system operation. It supports asynchronous communication, facilitates scalability and fault tolerance. The different components of PoSeID-on can be seen as individual applications. Messages are passed between them, through a queue mechanism, each and every one signed by the sender and encrypted for the recipient. When a component is unreachable (e.g., a Data Processor is offline), the message will be kept until either a predefined timeout expires or the recipient comes back online.

3.4 Use Cases

This section briefly describes a representative subset of the platform use cases, in order to complement architectural description given above and to help understand the general interactions between system components. These are non-exhaustive and were used as a guide for the functionalities identified during requirement analysis and evaluation. The use cases are grouped by initiating actor.

3.4.1 Initiated by the Data Processor

Request PII values from a specific or all data subject: Data processors are able to request PII of a data subject through the DP API. A check will be performed on the blockchain to see if the requester has the correct permissions before PII is retrieved from the data processor holding it.

Request access to PII from a specific or all data subjects: Data processors are able to request permission to access PII of a data subject, identified by type and upon proper description of the intended use for that information and how that information will be stored and handled.

This request will be processed via smart contracts and sent to the data subject. A reference to it will be stored in the blockchain.

Send data possibly containing PII to be analyzed: Data processors are able to send data in structured or unstructured formats to the Personal Data Analyzer in order to identify possible PII contained in it.

3.4.2 Initiated by the Data Subject

Grant access to PII to one Data Processor: Data subjects will be able to see all permission requests sent to them by data processors and accept them through the web dashboard. The act of granting access is automatically logged in the blockchain.

Revoke access to PII from one Data Processor: Data subjects will be able to revoke permissions for data processors, provided that the access to the PII they are processing is not mandatory as explicitly stated in the permission request. Upon access revocation, data processors are notified with an API call and are legally obligated to delete the PII in question. Revocations are also logged in the blockchain.

View PII known to one Data Processor: Data subjects will be able to see all PII types and values known to one data processor. These values will be requested from the associated data processors upon view request.

Update PII known to one Data Processor: Data subjects will be able to update PII if the data processors provide the necessary interface for this purpose and if the PII is not read-only. Upon PII update, the data processor which have access to this PII and the primary holder of the PII are notified and sent the new values through an API call. The act of updating PII values is also logged in the blockchain.

3.5 Summary

This chapter contextualizes the PoSeID-on project for the purpose of understanding the contributions made to the project and the development of the RMM's architecture. In this section the overall design of platform is presented. The key components of the platform are also described, as are the main actors and their definition.

A representative set of use cases of the platform are also introduced, organized by initiating actor. This provides the necessary overview to understand the role of the RMM in the project and the functionalities offered by the components which will provide data to it, as well as the flow of interactions leading to the production of such data.

Chapter 4 RMM Design and Specification

The Risk Management Module (RMM) is responsible for monitoring PoSeID-on, both from a system-wide perspective and from the point of view of operations involving individual data subjects. The RMM is expected to detect and evaluate possible security and privacy risks inherent to the processing of personal data, such as the exposure of PII through the malicious use of the PoSeID-on platform.

4.1.1 Methodology

To design the RMM several factors stemming from the original description of the module had to be considered. A set of objectives was extracted from it and evolved during the initial discussion around the overall PoSeID-on concept and architecture, leading to the identification of constraints and necessary attributes for each of the project modules. The following sections will introduce the factors that influenced the design and implementation of the RMM, followed by the solution proposed to address them.

4.1.2 Objectives

From the brief description of the Risk Management Module in the original project description, it was possible to pinpoint the following goals:

1. The module must use machine learning to evaluate services (data processors) connected to the PoSeID-on platform.
2. The module must advise data subjects which of these services to disable.
3. The module must analyze information from PII transactions and external data to enrich them.
4. The module must monitor all warnings generated by the blockchain platform.
5. The module must analyze historical data logs, either received directly or from log management systems.
6. The module must not automatically (re)act based on findings. Instead, it must only provide actionable insights to the data subjects and systems administrators, which will then evaluate those findings and decide on how to react.

4.1.3 High-level Functional Requirements

From the objectives stated above and the discussion with project partners, a set of high-level requirements – provided in Table 4.1 – was compilated.

Requirement ID	Title	Description
R1	RMM Data Sources	RMM will receive information from the blockchain through the blockchain API and information from the API Gateway
R2	RMM Risk Detection	RMM will detect risk events based on blockchain transactions and previous stored risk events information
R3	RMM Risk Events	The RMM will create, update and delete PII risk events
R4	RMM Notifications	Depending on the nature of risk events notifications will be delivered to involved data processors, data subjects and PoSeID-on platform administrators
R5	RMM PII in Risk Events	Risk Events may include PII in case of explicit consent by the data subjects and data processors to the RMM
R6	RMM Risk Event Storage	Risk events will be logged in a secure way for later analysis, forensics and processing
R7	RMM Data Processor Reputation	RMM will associate a reputation score with each data processor based on previous track record
R8	RMM DP Reputation	Reputation scores will be created, updated and deleted by the RMM
R9	RMM Risk Notification Delivery	RMM will notify identified parties in a risk event through the web dashboard or other channels specified for this purpose
R10	RMM Risk Prediction	Risk Management Module will predict risks based on reputation, PII transactions and system logs

Table 4.1 Risk Management Module Requirements

4.1.4 Quality Attributes

From the discussions with project partners and the use case descriptions of each pilot of the project, several attributes were verbally agreed upon for every module of the system. No metrics were formally defined. Nevertheless, this step allowed the identification of important qualities for the module. The attributes identified for the RMM can be seen in the table 4.2.

Quality Attribute	Attribute Refinement	ASR ID	ASR
Performance	Latency	QA1	Analysis results must be delivered in near real-time (within a user session) to be actionable. A user session can be considered from 5 to 10 minutes.
Availability	Fault-tolerance	QA2	The module must provide fault tolerance and graceful degradation mechanisms.
Scalability	Multiple Instances	QA3	The module must be able to scale horizontally in order to cope with increasing and decreasing number of users.
Privacy	PII Privacy	QA4	The module must be compliant with the GDPR and minimize the PII needed for analysis.
Security	Data Security	QA5	Information in transit and at rest must be secure.

Table 4.2 Quality Attributes associated with the RMM

4.1.5 Technical Constraints

Since all modules of PoSeID-on need to be integrated into a single platform, partners had to agree on which technologies to use for deployment, development and inter-module communication. These directly translate into technical constraints for the RMM, which are presented in Table 4.3, provided next.

Constraint ID	Constraint	Description
TC1	Language	Most common programming languages are encouraged, for maintainability and debug purposes. Suggestions were Java, Python and Go.
TC2	Operating System	Modules must be buildable and runnable on Linux.
TC3	Message Queue	Modules must communicate through RabbitMQ [45].
TC4	Messaging Protocol	Google's Protobuf [46] will be used for message protocol definition.
TC5	In-Transit Encryption	Libsodium [47] will be used for encryption, for signing and encrypting messages.
TC6	Containerization	Modules must be containerized and compliant with the Open Containers Initiative (OCI) [48].
TC7	Cloud Deployment	Modules must be deployable in a cloud environment. The platform chosen for this was Kubernetes [49]. For local development Minikube [50] will be used.
TC8	Log Manager	The log manager to be used in PoSeID-on will be Graylog[45], each module should send their logs to Graylog for audit purposes.

Table 4.3 Technical Constraints

4.1.6 GDPR concerns

Since the RMM also processes data from PoSeID-on, it must comply with the GDPR. Logs and operational data will not directly contain PII, but they will contain metadata that can be eventually considered as PII. Some examples of such data include data subject's and data processor's identification tags, connection location, time of log issuing and other data of sorts. As such, the principles presented in GDPR still apply. These principles, and the approach to respect them, are:

- **Lawfulness, fairness and transparency:** The intended purpose of the RMM is stated in the terms of service of PoSeID-on and the RMM code will be available as open-source.
- **Purpose limitation:** As above, the purpose of data collection is the analysis of the general system behavior and protection of the data subjects. Public code can guarantee verification of this.

- **Data minimization**: The only PII collected, for security purposes, are the ID's of the entities involved in each log message.
- **Storage limitation**: Logs containing ID's should be easily deleted once the time for holding the data expires or the data subject requests their deletion. Models being trained must not store the PII of data subjects and cannot be used to track a single subject.
- **Accuracy**: Does not apply for the specific purpose of the RMM.
- **Integrity and confidentiality**: Communication between modules must be secure and data storage must support encryption at rest.

4.1.7 Data Sources

The data sources which will supply the RMM with actionable information are the Gateway API, the Blockchain API and the Dashboard. The information provided by each of these sources will be in the form of system logs, and will include:

Operational logs from the Gateway API, including information about the parties involved in the operations, for:

- Connection logs
- Permission requests
- PII requests
- PII exchanges
- Permission lookup requests

Operational logs from the Blockchain API, including information about the party using the Blockchain API, for:

- Read requests
- Write requests
- Permission lookups
- Validated requests
  (i.e. translation from system requests to blockchain transactions is valid)
- Rejected requests
  (i.e. translation from system requests to blockchain transactions is invalid)

Operational logs from Blockchain received through the Blockchain API or other method to be defined, such as:

- Transactions accepted
- Transactions rejected
- Accesses to the chain by the Blockchain API
  (which party accessed it and for what purpose)

Operational logs from the Dashboard including the identification of users performing the operations, for:

- Access location
- Login attempts
- Permission management logs:
  - Permission listing
  - Grant permission (as answer to a request by a DP)
  - Revoke permission
  - Create permission

In most cases, raw system logs will be provided, but for PII operations a field in the message protocol may indicate the type of operation and the issuer of the operation. The format of the messages can be seen in the following section.

4.1.8 Interaction Flow

Figure 4.1 provides a diagram with the generic sequence intended for the interaction between the RMM and other system components. Whenever a message is sent to the RMM, it should analyze the log, store the analysis results and the data and, whenever an anomaly is detected, dispatch a warning to the Dashboard for the data subjects involved in the anomaly. Data processors involved in the anomaly are also identified and their reputation must be updated accordingly. Administrators receive a copy of the warning and can then later on review the warning anddecide either to keep the reputation change or mark the warning as a false positive.

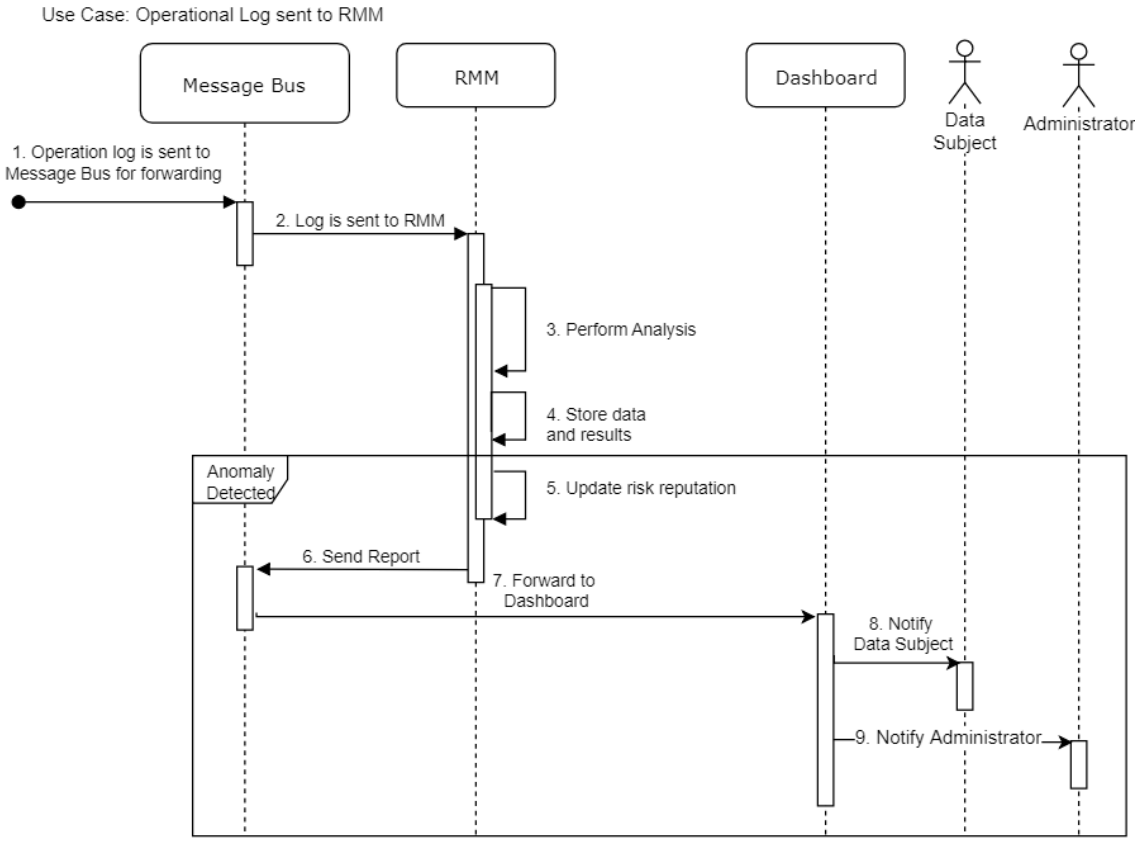

Figure 4.1 RMM System Interaction (General Overview)

4.2 Anomaly Detection Approach

The approach to be taken for anomaly detection must have in consideration the points mentioned in section 4.1.1 , to ensure the goals of the module are achieved without compromising the constraints set by the PoSeID-on platform and project partners. It is necessary to consider:

- **Quality Attributes** – The selected approach must be not only accurate but also able to provide results in a near real-time fashion.
- **Availability of training data** – As this is a novel area and a novel system, there are currently no labelled or even unlabeled data sets which can be use to describe the behavior of a PII management and distribution system. Without training data, the immediate solution is to select an approach which is able to model the behavior of the system without any previous knowledge.
- **Future proofing** – The RMM should also be able to adapt to changes in normal behavior of the system, as any new platform is expected to see a growth in usage with time and consequently also a change in normal behavior. Furthermore, with visibility, the number of possible malicious attacks or misuse of the system is also expected to grow.
- **Data sources** – The way on which data is made available and volume of data being sent to the RMM also influences the approaches that can be taken. In this case, system logs are the default source of information.
- **Available time for research and implementation** – The PoSeID-on project plans a first functioning prototype by September 2019. As such, re-use of existing technologies is encouraged in order to have a functioning and integrated prototype by that date.

From this analysis, we can conclude that a continuous unsupervised learning approach is the best option for the system at hand, combined with stream-based analysis, in order to derive real-time results from the arriving data. The inclusion of semi-supervised algorithms is foreseeable at a later stage of development in order to make use of manually labelled data, once system analysts can give feedback on results.

In addition, the selected approach to anomaly detection will be a combination of domain specific feature extraction and the generic well-known approach for log analysis anomaly detection (cf. section 2.4), since received messages contain structured information, such as operation and data subject ID's and a raw system log.

4.2.1 Log Collection

Logs in PoSeID-on are delivered to the Risk Management Module through the message bus, using the established message protocol based on Protobuf [51] for the message format and Libsodium [51] for encryption. Since a message will be sent for each log using this message protocol, we don't just send a raw description of the log, but also an extra set of parameters that can be useful for analysis.

Examples of this are PII or metadata regarding the data subjects and data processors involved in the operations described in the logs. By separating more sensitive fields from logs, we keep this information out of the general log storage – thus accounting for privacy

issues. It is also important to keep in mind that the RMM collection of this extra PII is optional and subject to consent.

4.2.1.1 Message definition

Table 4.4 summarizes the parameters to be included in the messages received by the RMM. Each message will include some or all the parameters presented in Table 4.4.

Parameter	Description
Module	The module issuing the message
Operation	The operation that is being performed, where applicable. Some logs may be associated with specific operations, such as PII Permission Requests, PII Permission Grants, PII Permission Revocation, PII Transacted/Accessed.
Sender ID	The ID of the issuer of a certain operation, where applicable. If it is a system operation it can be the ID of the module producing the message. If it is a PII related operations it can be a Data Subject ID or a Data Processor ID (only sent when consent is given).
Sender IP	The IP originating the message.
Receiver ID	The ID of the target of a certain operation, where applicable. If it is a system operation it can be the ID of the module receiving the request. If it is a PII related operations it can be a Data Subject ID or a Data Processor ID (only sent when consent is given).
GELF Log	The default fields sent to GELF log manager such as the host, short message, full message (raw log) and syslog level.

Table 4.4 - Message Parameters

The operation description parameter allows us to extract a feature vector consisting of the count of each operation type instance happening per window of collected messages. The raw log from GELF will be used to build a set of log templates which will also be used for log event counting per window. The fields identifying the entities involved in a PII operation, such as the data processors and data subjects involved (supplied if consent is given) will be used to identify the parts at risk or creating risk, when an anomalous pattern is detected in a message window.

4.2.2 Log Parsing

The first step of the anomaly detection pipeline is to structure arriving information in a way that allows us to compare data instances. From the message format we already have structured information that is comparable between instances, but raw logs require some processing before then can be used by models, which in most cases receive as input numerical features.

The quality of the structuring of the raw logs and the creation of the event templates directly affects the anomaly detection. Moreover, good log parsing enables efficient search, filtering

and grouping of logs. Current solutions for automated log parsing have built-in parsing support for common log structures, such as Apache and Nginx logs. Since the PoSeID-on platform will have a collection of logs being generated by several third-party frameworks and tools and custom-built software, with unique logs, there is the necessity of manually configuring custom parsing with regex scripts, grok patterns or a parsing wizard. This does not scale particularly well, and an automated solution is much more flexible.

Moreover, PoSeID-on is a novel platform working in a novel domain, making adaptability to change a must. As such, there was the need for analyzing the current automated log parsing solutions in order to apply one to the Risk Management Module log parsing step. Michael Lyu et al. have performed an in-depth research [52] on 13 representative log parsers proposed in the literature, and have compiled their results as seen in Table 4.2.

Log Parser	Year	Technique	Mode	Efficiency	Coverage	Preprocessing	Open Source	Industrial Use
SLCT	2003	Frequent pattern mining	Offline	High	×	×	✓	×
AEL	2008	Heuristics	Offline	High	✓	✓	×	✓
IPLoM	2012	Iterative partitioning	Offline	High	✓	×	×	×
LKE	2009	Clustering	Offline	Low	✓	✓	×	✓
LFA	2010	Frequent pattern mining	Offline	High	✓	×	×	×
LogSig	2011	Clustering	Offline	Medium	✓	×	×	×
SHISO	2013	Clustering	Online	High	✓	×	×	×
LogCluster	2015	Frequent pattern mining	Offline	High	×	×	✓	✓
LenMa	2016	Clustering	Online	Medium	✓	×	✓	×
LogMine	2016	Clustering	Offline	Medium	✓	✓	×	✓
Spell	2016	Longest common subsequence	Online	High	✓	×	×	×
Drain	2017	Parsing tree	Online	High	✓	✓	✓	×
MoLFI	2018	Evolutionary algorithms	Offline	Low	✓	✓	✓	×

Figure 4.2 Summary of Automated Log Parsing Tools (from [53])

From this list, the following key characteristics must be taken in consideration:

- **Mode:** Log parsers can be categorized into two main modes, *offline* and *online*. *Offline* log parsers are a type of batch processing and require all log data to be available before parsing. On the other hand, *Online* log parsers allow processing of log messages one by one, in a streaming manner. As such, only *online* parsers were considered.
- **Efficiency:** This characteristic is of the utmost importance, considering the potentially large volume of logs being generated per second. As all of the subsequent analysis tasks depend on log parsing, it is necessary that the parser is efficient in order to perform real-time anomaly detection. As result, *High* efficiency is the value that must be aimed at.
- **Coverage:** Indicates the capability of a log parser to successfully parse all input log messages. If a checkmark is present, then the log parser is able to handle not only frequent patterns but also rare events, which is not always the case with log parsers. Ignoring rare events might result in missing important events, which are consequentially missed on the anomaly detection, so full coverage is necessary.
- **Open-source:** The availability of the source-code makes it much easier to adopt the parser, reusing it or even extending its implementation. As such, only open-source parsers are considered for our implementation.

Restricting the selection process to *online* and *highly efficient* automated log parsing tools, only SHISO[54], Spell[4] and Drain[55] were considered. Among these three, all offer optimal coverage but only Drain is open source. Coincidentally, Drain also achieves the best overall accuracy, while also being successfully implemented in a production system in collaboration with Huawei. In addition, it has been successfully extended with Apache Spark [35] for

parallelization, exploiting Drain's data partitioning, although this implementation has not been made open source. This makes Drain an optimal candidate for log parsing in RMM.

The Drain approach to log parsing uses a fixed depth parsing tree, which encodes specially designed rules for parsing. As input, Drain receives a batch of raw text logs and a simple description of the raw text formats, given during configuration, in order to extract components (e.g. the following string: "**<Date> <Time> <Pid> <Level> <Component>: <Content>**").

Optionally, regular expressions can also be given in order to select well-known parameters from the raw text, such as numbers, IP addresses or session ID's.

The output is the parsed log, structured according to the format given as input, with the addition of a template (event) extracted from the free-form part of the log and the parameters removed from that template. Templates are updated in an online manner as new logs arrive. Figure 4.3 demonstrates how a template looks like, with "<*>" corresponding to extracted parameters or variable parts of the log.

EventId	EventTemplate
04137b95	BLOCK* ask <*> to delete <*>
09a53393	Receiving block <*> src: <*> dest: <*>
32777b38	Verification succeeded for <*>
3d91fa85	BLOCK* NameSystem.allocateBlock: <*> <*>
40651754	<*> Starting thread to transfer block <*> to <*>
415a1760	BLOCK* ask <*> to delete <*> <*> <*> <*> <*> <*> <*> <*> <*>
5d5de21c	BLOCK* NameSystem.addStoredBlock: blockMap updated: <*> is added to <*> size
626085d5	<*> Served block <*> to <*>

Figure 4.3 Example of Log Templates

4.2.3 Feature extraction

From the templates created in the previous log parsing step, a numerical feature vector will be created by performing a count of the occurrence of each template in the time window under analysis. If the received message has an operation ID associated with it, a count of operations per window is also performed.

Since the objective is to model the normal behavior of the system by measuring the Euclidean distance between points, normalization of the feature vector is necessary. The approach for normalization that will be taken is to normalize the features to unit length, separately for template occurrences and operation occurrences, since templates and operations are not independent features.

In addition to how templates are selected, how to perform log grouping for occurrence count is also a relevant factor for achieving good results.

4.2.3.1 Time Windowing

Since results must be generated in very short intervals (3 to 5 minutes), if analysis is performed only on those intervals of arriving logs it is possible that most anomalies will be lost. It will be possible to detect point anomalies, such as a data subject being the target of a burst of permission requests, but it will not be possible to detect collective anomalies. Anomalous patterns happening over the course of more than 5 minutes would simply be lost.

For this reason, the approach taken will use sliding windows, where a short step will allow for outputting results in real-time, but a larger window will offer more insight on longer patterns of system operation.

4.2.3.2 Log grouping and RMM goals

Considering the goals of the module, it is also necessary to group PoSeID-on logs in respect to the following aspects:

- **Data Subject's Behavior:** By grouping data by subject identification it is possible to model how a single data subject is normally involved in a collection of operations. It is also possible to directly identify the data subject being targeted since an anomalous grouping of logs will have a single target.
- **Data Processor's Behavior:** By grouping data by data processor the same applies. It is possible to detect which data processor is behaving in an anomalous way and notify administrators that a data processor may be compromised.
- **System Behavior:** By grouping logs by time window only, a model of the overall system operations is created, which will allow for detecting system-wide anomalies. This will make PoSeID-on more secure but won't allow for direct identification of the anomaly cause until the log window is further analyzed. If desired, grouping logs by issuing module may give further insight on the anomaly source.

As such, to have a complete view of PoSeID-on's normal behavior, at least three approaches to log grouping must be taken and three models must be trained, each providing different insights on the system.

4.2.4 Anomaly Detection

For the anomaly detection step of the RMM pipeline, due to the lack of training data and the necessity of a stream-based algorithm with an available implementation working out of the box, K-Means will be used to model system behavior in the prototype version of the RMM.

4.2.4.1 K-Means Clustering Outlier Detection

K-Means partitions the incoming feature vectors into clusters, each incoming vector being added to the cluster with the nearest mean. Clustering is performed by minimizing within-cluster variances (squared Euclidean distances). In the streaming variant of K-Means, it is possible to define a decay parameter which determines how much previous vectors contribute to the current clusters.

The usual approach for anomaly detection using K-Means is to calculate the Euclidean distances from each vector to the center of its assigned cluster and selecting a percentage of each cluster's vectors with the largest distance to be considered an anomaly. In a streaming scenario this would involve storing all the feature vectors and recalculating the distances to the center of each cluster every time the model is re-trained.

As this is impractical for long-running systems, we resort to descriptive statistics to summarize the dataset. As each feature vector arrives, the distance to the center of the predicted cluster is calculated and stored in a sample set of distances for each cluster. As new values arrive old values are discarded. In addition, the number of vectors that have been added to each cluster are counted.

If an incoming vector's distance to the center of its predicted cluster is over a threshold Z-Score, that is, a number of standard deviations over the mean of the descriptive set of distances, that vector is considered an anomaly.

Since a stream approach is used, discarding anomalous vectors may make us lose information in the long term. As a result, anomalous points are still used to recalculate the model.

When a new vector arrives, an extra check is performed, in order to prevent anomaly formed clusters from not triggering anomalies by taking in points similar to the cluster mean. If the cluster predicted has less than a threshold percentage of all points added to the model, that point is considered as anomalous.

4.2.5 Stream and Batch Approach

Recent works, such as [51], combine both stream processing and batch processing by leveraging a simplified lambda architecture [56] in order to accommodate both near real-time and also a slow-rate event processing. Real-time allows for critical alerts which require low reporting latency, while slow-rate processing can make use of the large data sets collected in order to detect anomalous patterns.

A similar approach was taken for the RMM. Unsupervised learning algorithms for batch processing can be used for improving the detection rate and even for improving the stream-based model by training the stream prediction model based on the batch analysis results [57], when using a semi-supervised approach. This is also in line with the possibility of having an administrator or security analyst analyzing the results of the models and providing feedback for further re-training.

As such, a stream and batch approach are taken, using of stream capable unsupervised learning algorithms. These can either be trained on the fly in the stream-layer or using the batch-layer, in periodic intervals. Having a batch layer will allow for more flexibility in what algorithms can be used for analysis, as there are well tested algorithms available for batch analysis which do not have a usable counter-part for evolving stream analysis yet, such as PCA, even though there have been recent proposals for it[58][59].

4.3 Proposed Architecture

The architecture for the Risk Management Module, depicted in Figure 4.4, is based on the proposal by *N. Marz.* for a generic and scalable data processing architecture, which he called Lambda Architecture [60], and the proposal by *Yamato et al.* of an architecture for real-time predictive maintenance [61]. This architecture allows for an efficient processing of the volume of information being constantly generated by the PoSeID-on platform, analyzing data as it arrives in the RMM and providing data subjects feedback regarding exposure risks in a near real-time manner, while also allowing this module to analyze data and extract valuable insight from large volumes of data, without instant feedback for the data subject, by using more complex, time- and resource-intensive methods.

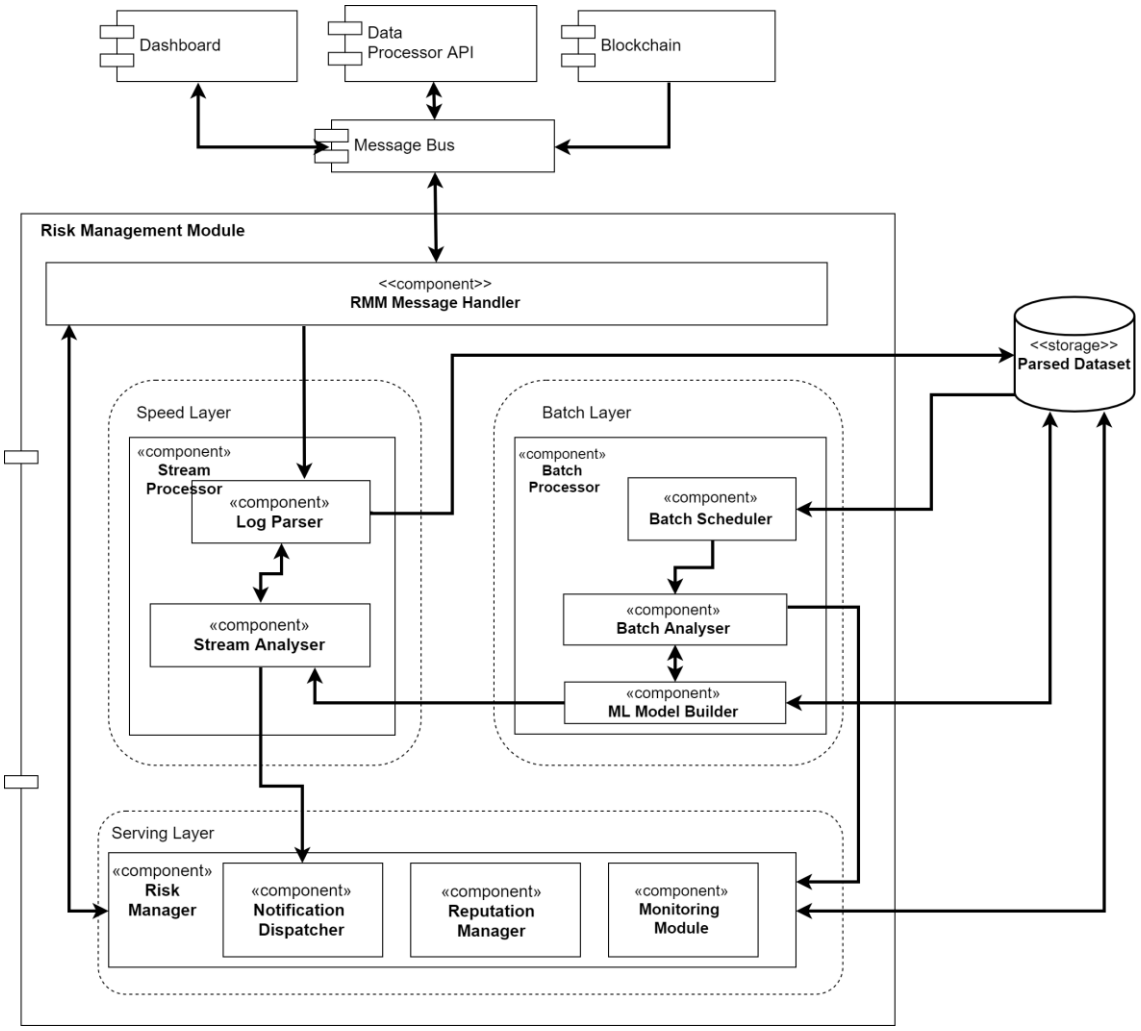

Figure 4.4 RMM Architecture Proposal

This architecture features three layers: a *batch layer*, a *speed layer* and a *serving layer*. As data regarding system operations and blockchain transactions arrives at the Risk Management Module, it is dispatched to the speed layer for parsing. Once the data is parsed, it is sent down-stream for anomaly detection and stored in the module's external database.

The batch layer fetches the already parsed data and uses it for historic risk analysis. The amount of data available will depend on how long the RMM is allowed to retain data, provided the data subject has given explicit consent for this purpose. This more in-depth analysis takes significant time and is not expected to work for near real-time risk detection. Instead, this layer provides warnings in case of risk detection in an "offline" manner and is can also be used for training and updating machine learning models used in speed layer's near real-time analysis.

The speed layer deploys the models created by the batch layer or trained on the go – if the algorithm allows it, as in the case of the streaming K-Means variation. It analyses the stream of data in real-time, by collecting small batches of data every couple of seconds from message queue. In case an anomaly is detected, both layers dispatch a warning to the serving layer.

The serving layer is in charge of receiving the results from both the batch and the speed layers, and notifying the respective entities about risk exposures, through designated channels. Right now, this is performed through the message queue, with the Dashboard as

receiver and router of messages to the end-target of the warning. The serving layer is also in charge of receiving feedback from administrators, regarding such risk notifications, and updating the associated risk reputation for each data processor.

4.3.1 Components

Each internal component has a small set of responsibilities in order to keep components loosely coupled and the RMM easily customizable. The responsibilities of each component are discussed next.

4.3.1.1 RMM Message Handler

The Message Handler is responsible for subscribing to the Message Bus, decrypting and decapsulating the incoming stream of messages, to then pass to the speed layer. Secure communication between the RMM and Dashboard will also be managed by this module, forwarding incoming remote procedural calls to the risk manager and sending back results and notifications to the respective parties. The message handler can be further extended to other communication channels as necessary.

4.3.1.2 Batch Processor

The batch processor is the component were batch analysis is performed. This analysis will make use of the parsed and structured messages already parsed by the stream layer.

Batch Scheduler: subcomponent that manages the scheduling of batch tasks and storage access. Whenever a batch run is due, it fetches information from the external storages and passes it to the analyzer component.

Batch Analyzer: subcomponent which will perform complex event processing and dispatch results to the ML Model Builder and Risk Manager for further action.

Machine Learning Model Builder: this subcomponent will create and update machine learning models to further improve detection of anomalous behaviors and risks based on the analysis results. Feedback from system administrators on identified anomalies will also improve the models when using semi-supervised or supervised machine learning models.

4.3.1.3 Stream Processor

The stream processor is the component performing near real-time analysis. As data arrives in the system, it is temporarily stored and analyzed.

Log Parser: subcomponent which will parse the incoming messages into structured data to be used further down the pipeline.

Stream Analyzer: subcomponent responsible for the analysis of the structured data. It must be able to evaluate risk in a near real-time fashion, being able to provide risk warnings within a short time window after the occurrence of a PII exchange or anomalous system behavior. Structured data grouping and windowing happens in this component. All prediction models will also be deployed here.

4.3.1.4 Risk Manager

The Risk Manager is the component which will act on the results provided by both Batch Processor and Stream Processor. All reasoning from the analysis is passed to this component, which will then take the necessary procedures for each evaluated case. This can be:

- **Notifying the respective parties at risk:** Depending on which model detects an anomaly, the data processor behavior, data subject or system behavior model, a warning will be dispatched to the respective party.
- **Updating a data processor reputation:** Whenever an anomaly is tied to a certain data processor, their reputation should be updated accordingly. Administrators may manually confirm or deny reputation updates later on.

This component also receives messages from the message handler regarding feedback on an identified risk**.** When an administrator verifies an anomaly, they can send a message to the RMM confirming or denying the prediction.

Notification Dispatcher: subcomponent in charge of sending notifications to the dashboard or other specified channels.

Reputation Manager: subcomponent which is in charge of creating, updating and maintaining risk reputation of data processors and also provide this information to the dashboard and other components of the RMM.

Monitoring Module: subcomponent which will provide a correction mechanism for predictions and possibly a direct way to access the data storage and analysis results. No formal requirements have been discussed for this component, but it is possible to foresee these uses and thus it has been included in the architecture.

4.3.1.5 External Storage

An external storage is necessary to keep:

- **Parsed data** – The data provided by the several system components and parsed using the drain parsing algorithm.
- **Analysis results** – The results for each analyzed window of logs.
- **Feedback from security analysts** – Manually given inputs on data instances which were previously analyzed and stored in the system.
- **Learning models** – Models trained and deployed by the respective analyzer components.
- **Parsing tree** – The data structure used for parsing logs in the log parser algorithm.

Having this storage external to the RMM instance allows for several instances of the RMM to share models and data structures for the log parser, and provides more flexibility as to how and where storage is deployed. This external storage must be able to be deployed in a distributed manner and have highly performant writing speeds, in order to be able to cope with the volume of data output by the RMM.

4.3.2 Interfaces

Initially, interfaces planned for the RMM included the message system for the PoSeID-on platform and a direct interface with the blockchain module, through one of the Ethereum nodes, in order to listen to smart contract-triggered events.

Later on, focus shifted to having every log message delivered through the message bus. As such RMM instances connect to the message bus and receive all communication through the RMM queue. This results in a simpler and cleaner platform architecture, while also providing enhanced support for implementations using parallel instances of the RMM for load-balancing.

4.3.3 Parallelization

The potential number of data subjects managed by a single PoSeID-on platform is in the order of millions. For example, in the Italian Ministry of Economy and Finance's use case, the PoSeID-on will be used to manage PII associated with the payrolls of approximately 2.1 million public sector employees. Moreover, many of the concepts of PoSeID-on only reach their full potential if the system covers a wide number of data subjects and data processors (e.g. a government-sponsored PoSeID-on platform for a whole country).

The PII access patterns for this type of operation are still unknow, so the load on the RMM cannot be estimated accurately. Nevertheless, it must be designed with performance and scalability in mind in order to support this magnitude of users.

To support this, the RMM must be able to have several instances running independently, without compromising the quality of the predictions. Models can be shared between instances by persisting them in the database, and the same case applies to the parsing structure.

Synchronizing the training of these models, however, becomes a not so trivial task if there are several instances running in parallel, and becomes impractical for the streaming machine learning models that are trained as data is received. In order to overcome this, for the specific case of the RMM, introducing an orchestrator and only synchronizing the batch layer is proposed.

4.3.3.1 Orchestrator for data separation

Since every incoming log message has an identifier (such as data subject ID, data processor ID or module issuing the message), it is possible to create a routing table were each RMM instance receives all logs respective to a certain ID. An orchestrator can manage all active RMM instances and a routing table, balancing the number of ID's assigned to each instance. This way the load is distributed while maintaining patterns' consistency, since all logs respective to a single entity are sent to a single instance.

In addition, by storing the parsed data with a reference to the RMM instance ID, data is also logically separated and each RMM instance can fetch and perform batch training or prediction without need for synchronization between instances. Figure 4.5 illustrates this while using the message queue for routing logs.

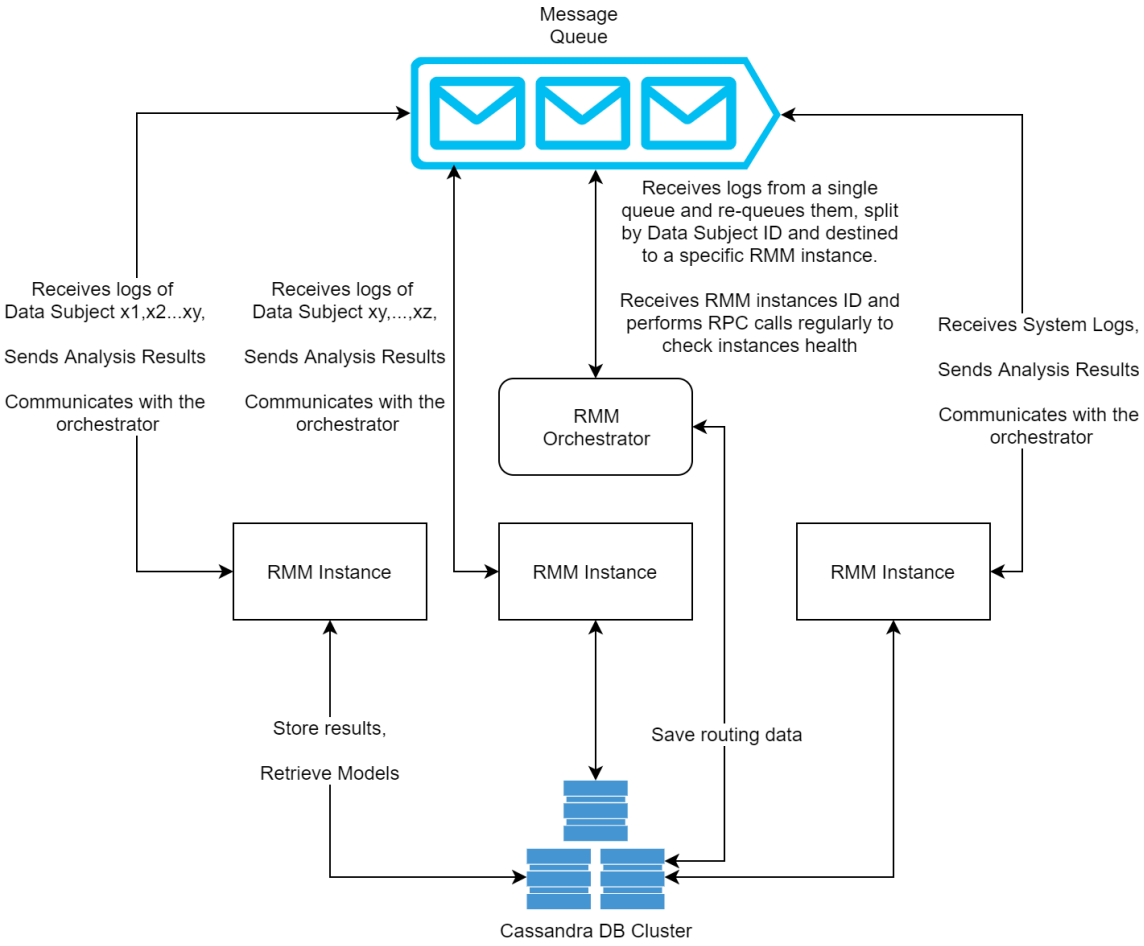

Figure 4.5 Orchestrator Data Flow Using Message Queue

Re-using the message bus for routing will double the number of messages circulating in the system, but as PoSeID-on will benefit of distributed deployment, this shouldn't be an issue. The orchestrator can be parallelized if the routing table and connected instances information is stored in an external storage. Deployment can then be optimized by deploying in clusters, such as having a dashboard, storage and message queue node paired with an orchestrator and couple RMM instances per cluster. Figure4.6 presents the orchestrator architecture, which includes the following components:

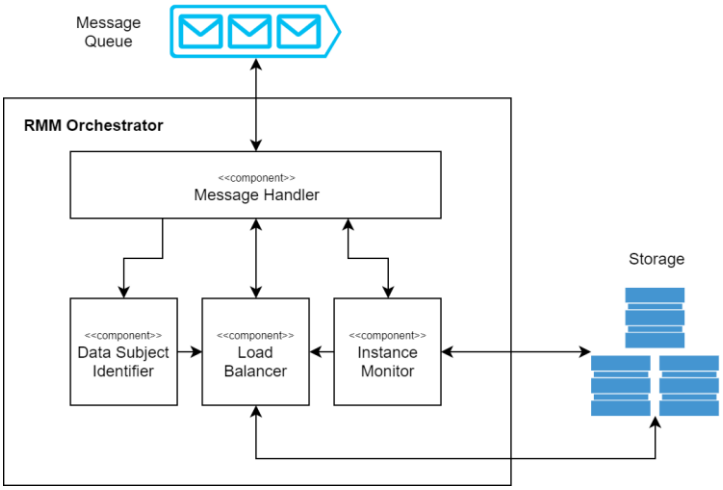

Figure 4.6 RMM Orchestrator Architecture

Message Handler: manages the message queue incoming and outgoing communication, decrypts and decapsulates messages and passes them to the data subject identifier.

Data Subject Identifier: retrieves data subject identifiers by extracting them from the message headers and performing a simple regex retrieval of ID's in the raw log.

Load Balancer: manages the routing of messages by synchronizing with the routing table and active RMM instances, re-queuing the original message in the queue of the instance managing that ID.

Instance Monitor: regularly checks if an instance is still active in order to inform the load balancer and store that information in the distributed storage. In this figure the instance monitor receives regular messages from the queue, sent regularly by the RMM instance but it is also possible to implement this through other means such as sockets.

4.3.3.2 Synchronization of models

Synchronizing the training and loading of stream-based models becomes impractical when the number of instances grows. Since real-time results are necessary, instances cannot wait for another instance to stop training a shared model before it can load, predict and train the model with its own data. As such, streaming models are not synchronized. This can affect predictions in a negative manner but as the volume of PoSeID-on users increase and the system has been running for a while, prediction results across instances should converge.

Alternatively, it is possible to synchronize only the batch layer. Since batch training and prediction can happen in an offline manner, the orchestrator can manage a time schedule for training of a shared model between instances and prediction can happen simultaneously.

4.4 Summary

This chapter presented the design of the Risk Management Module.

The RMM objectives, requirements, quality attributes and constraints were presented as means to support the choices made in the RMM architectural design process.

The anomaly detection approach was also described in detail, followed by the proposed architecture, which employs a simplified lambda architecture to allow both stream and batch prediction. Afterwards, the parallelization approach is also discussed, and a proposal is given to manage instances and synchronization.

Chapter 5 Implementation and Integration

This chapter reports the implementation efforts of the RMM, by describing the technological choices made and the components from the proposed architecture that were successfully implemented. Integration and deployment efforts are also described.

The work developed in the scope of this thesis was necessarily aligned with and constrained by the planification of the PoSeID-on project. For this reason, thesis priorities had to follow closely the priorities of the project, which focused on having a fully integrated and deployable platform as early as possible. As such, development of secure communication interfaces, containerized deployment and setup for Kubernetes deployment took priority over the development and refinement of the anomaly detection pipeline and orchestrator. The PoSeID-on overall workplan reflects this, with a first version of the platform components, with a reduced set of features per component, scheduled for delivery in July 2019 for a first integration round, followed by deployment at the trial sites and, afterwards, a return to component development to deliver the complete version of each component by May 2020.

Moreover, the fact that overall platform deployment is late (due to issues and delays in the blockchain-related components to be provided by Tecnalia), with the subsequent delay in the kick-off of the four planned pilots, has prevented taking advantage of having the PoSeID-on platform systematically running with real data to refine the RMM.

5.1 Technological Stack Choices

The prototype for the RMM was developed in Java 8. This version is explicitly necessary as the framework used for data processing does not support newer versions of Java yet. The choice of language derived from personal preference between the choices presented in technical constraint TC1. In addition, Java being a compiled language, is generally faster and more efficient than Python.

The technological stack derived from the technical constraints presented in Section 4.1.5 , as well as a comparison of similar technologies and personal preferences. This section gives an overview of the chosen frameworks for the core aspects of the RMM module, such as logging, data processing and communication, and provides insight on what drove such choices.

5.1.1 Adopted Frameworks

5.1.1.1 Apache Spark and Apache Spark Streaming

Apache Spark [62] is the de-facto standard for big data processing. It started as a batch processing but eventually also incorporated streaming into its framework by introducing DStreams, an abstraction which represents a continuous stream of data that can be treated as a very small batch. As such, using Spark and Spark Streaming [63] becomes very similar and accommodates the lambda architecture of the RMM.

There are alternatives such as the recent Apache Flink [64], which boasts faster stream processing times and also allows for batch processing. Nevertheless, Apache Spark is still better established, with several success cases in production systems, as well as boasting a large array of open-source libraries and community support.

Additionally, Spark Streaming was initially encouraged as there was discussion, within the PoSeID-on technical core, whether RabbitMQ or Apache Kafka would be used as message queue system. Eventually RabbitMQ was chosen, but the initially discussion encouraged the use of Spark Streaming as there are well documented and tested connectors already implemented for Kafka to Spark communication.

5.1.1.2 Junit 5

JUnit 5 was chosen for its simplicity and convenience for writing unit and integration tests.

5.1.1.3 Apache Log4J2

Logging framework for Java. Chosen due to its popularity and wide support. In addition, Graylog, the log manager choice for PoSeID-on, is compatible out-of-the-box with Log4J2.

5.1.2 Adopted Libraries and Packages

5.1.2.1 Apache MLlib

Apache MLlib is a machine learning library developed specifically for Apache Spark, making it a natural choice to use in combination with Apache Spark. Furthermore, it has a parallelized implementation of Kmeans++ for both batch and stream processing.

5.1.2.2 Lazysodium and Libsodium (C library)

Libsodium is an open-source cryptography library widely used in C and was a constraint introduced by the integration team for secure communication. Lazysodium is a wrapper over that C library, which allows its use in Java, with minor configuration efforts.

5.1.2.3 RabbitMQ AMQP Client and Spark Streaming Receiver

RabbitMQ Spark Streaming Receiver[65] is an open-source receiver which connects to RabbitMQ through RabbitMQ's AMQP Client library and fetches data in stream-like fashion, through micro-batches. Although it is an outdated open-source library, it is the current receiver implemented for Spark Streaming and is compatible with the latest versions of RabbitMQ and Spark.

5.1.3 Adopted Tools

5.1.3.1 Google Protobuf

Protobuf was collectively chosen by involved project partners as the message protocol defining utility. Protobuf makes it possible to define a clear communication protocol in a language and platform neutral extensible way. It provides a set of tools for compiling a

Protobuf file into classes of an array of languages, such as Java or Python, which perform serialization and deserialization of data according to the Protobuf file definition.

5.1.4 Adopted Storage Database

5.1.4.1 Apache Cassandra

Apache Cassandra[66] is a NoSQL database which supports high scalability while maintaining high performance due to its data partitioning and clustering model. The biggest advantage of using Cassandra over other NoSQL databases is the higher write speeds, due to the fact that more than one master node can be deployed, not only increasing availability but allowing for parallelized writing. This pairs well with the distributed deployment strategy of PoSeID-on. In addition, due to the lambda architecture approach, write operations will be vastly superior in number than read operations.

One downside of using Cassandra is the necessity of defining a data structure in beforehand, but since our log parsing step creates a structured output, it is possible to setup a data structure to accommodate the parsed data with minor effort.

5.2 Implemented components

Table 5.1 summarizes the development status of the RMM by breaking down the module into each component and its current status. It should be noted that having non-implemented components is not a sign of delay or under-achievement for this thesis, since as already mentioned the whole RMM is planned only for May 2020. Moreover, the delays in the four planned project trials led to the absence of real users and real data – which creates obvious difficulties in the refinement of the RMM.

Component / Layer	**Sub-Component**	**Status**
Message Handler		Fully Implemented
Stream Processor	Log Parser	Fully Implemented
	Stream Analyzer	Fully Implemented
Batch Processor	Batch Scheduler	Not Implemented
	Batch Analyzer	Not Implemented
	ML Model Builder	Not Implemented
Service Layer	Notification Dispatcher	Fully Implemented
	Risk Manager	Partially Implemented
	Monitoring Module	Partially Implemented
Orchestrator		Not Implemented
Storage		Fully Implemented

Table 5.1 RMM Component Implementation Status (January 2020)

The adopted development approach, in addition to the chosen frameworks, allows for easy customization of the module's pipeline as the code for operations on the data stream is separated into logical steps, as seen in Figure 5.1. These can be switched in a straightforward way, with minor code changes and more layers or branches can be added to the pipeline.

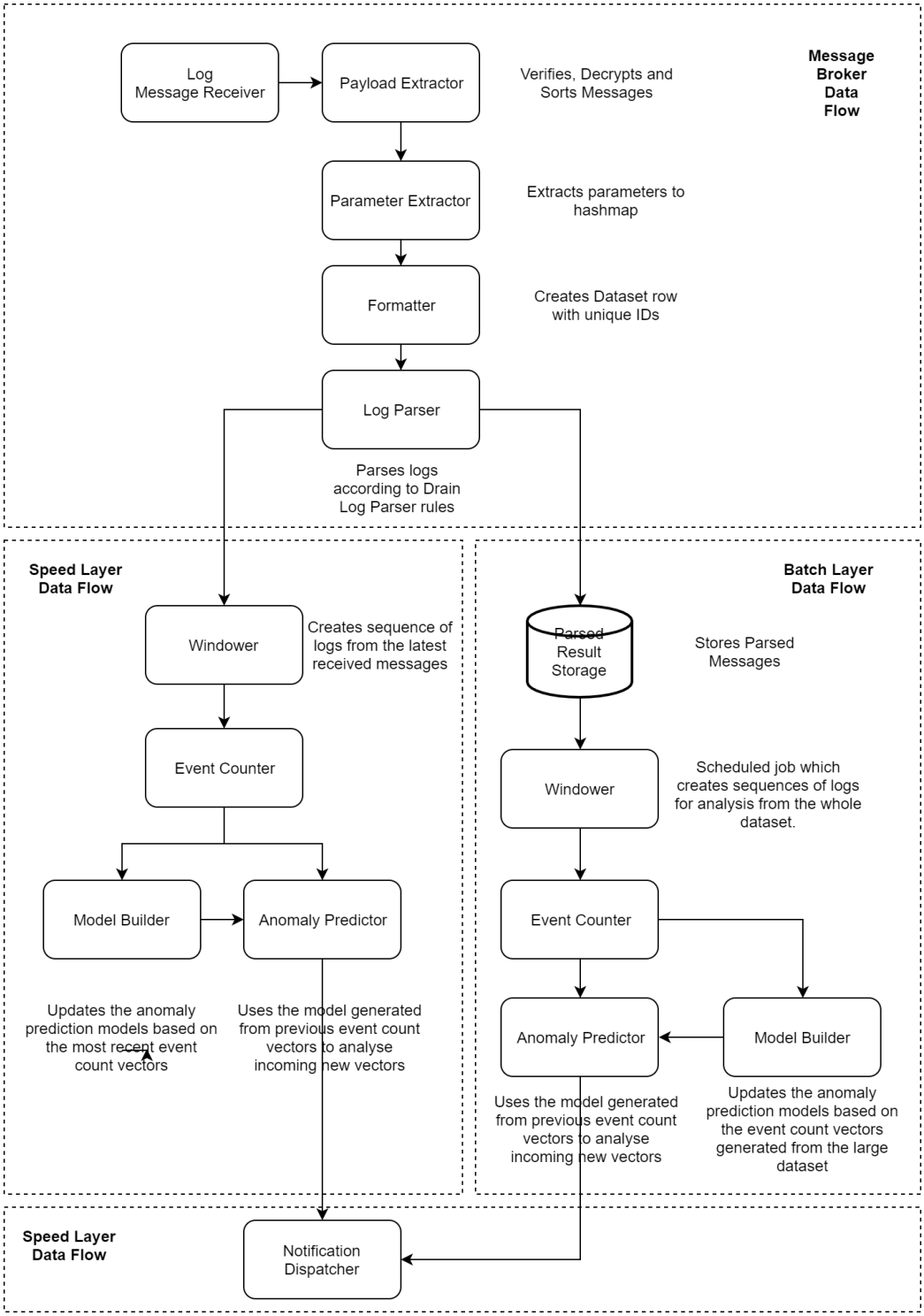

Figure 5.1 Implemented Data Flow

Although some components of the RMM have not been implemented yet, the pipeline leading to these components has already been set and has been implemented successfully from data reception, processing, storage, analysis and returning results to the PoSeID-on

platform, through the stream processor. The following sub-sections further detail each component's implementation status.

5.2.1 RMM Message Handler

The Message Handler was fully implemented, handling all communications inbound and outbound of the RMM.

Messages received by the RMM are decrypted and checked to make sure the RMM is the correct recipient. They are also validated through checking the message signature and the sender's certificate is validated by confirming the certificate authority's (CA) signature over the sender's public key.

All outbound messages incur the inverse process, signing, encryption and encapsulation according to the protocol described in Section Chapter 5 .

5.2.2 Stream Processor

The stream processor pipeline has been fully implemented, according to the RMM architecture design. Once verified and decrypted by the message handler, messages are structured into datasets organized by columns. Each parameter received in the message is turned into a column. The raw log column is then passed onto the Drain Log parser, to extract the corresponding template.

After extracting the template and structuring the raw log parts, the structured parts are re-joined to the previous dataset and sent down the pipeline.

5.2.2.1 Log Parser

As Drain is only available in python, a java implementation of the algorithm was specifically developed to facilitate the integration with the Risk Management Module pipeline and data analysis frameworks of Apache Spark. This implementation was also further extended in order to work for the specific purpose of the RMM.

In order to maintain log template consistency across batches and across instances, the original implementation was changed to support loading and exporting the parsing tree, and log ids are no longer saved in the data structure, to avoid excessive and unnecessary information from being stored in the database. In addition, other minor changes were made to accommodate Spark Streaming and Spark data structures in Java.

5.2.2.2 Stream Analyzer

The stream analyzer was successfully implemented using the Spark Streaming framework. Once the structured logs are received, logs are first grouped by time and then by ID. Once this is done an event count is performed based on the extracted template and parameters and normalized by row, using the total parameter and template counts.

Once this feature vector is normalized, it is passed into MLlib's implementation of K-means++, which returns a prediction for the cluster the vector belongs to. Once a prediction has been made, the result is passed to the anomaly predictor, which verifies if the cluster is an anomalous cluster or if the distance to the predicted cluster corresponds to an anomaly.

Slightly afterwards, in order not to train the model with the data being predicted, the model builder trains the model with the created feature vector.

5.2.3 Risk Manager

The Risk Manager, corresponding to the service layer, was partially implemented.

The Notification Dispatcher is currently working. Using the Message Handler, a message is created and sent to the ID corresponding to the feature vector.

The Monitoring Module did not have defined requirements and, as such, currently there is no functionality associated with it. The RMM however has already developed methods for handling RPC calls through the RabbitMQ queue and access to the structured data and results storage has already been implemented. As such, functionalities such as consulting more details about a certain anomaly can easily be added to the module.

5.2.4 Batch Processor

The Batch Processor has not been implemented yet, but the underlying structure has been set to accommodate its addition. Storage of the structured and parsed data has been fully implemented as has been access to stored data.

5.2.5 External Storage

External persistent storage and means to access it have been implemented fully, with the following information currently being stored in the Cassandra database:

- **Parsed Logs:** The predefined components of a parsed log, as setup in Drain, plus the parameters extracted.
- **Log Templates:** The determined templates obtained from Drain Log Parser.
- **Feature Vectors:** The feature vectors obtained from performing event count over a window of logs.
- **Analysis Results:** The prediction results for a window of logs.

The data model developed accommodates the distributed use case and tables were modelled according to the queries that are going to be performed, since the distributed nature and NoSQL model, makes introducing new or complex queries rather difficult. Making use of Cassandra's distributed nature, partition keys and clustering keys were defined to allow querying mainly to obtain results and information resulting of the feature vector analysis and to allow re-use of the parsed data, which will be read by the batch layer for further anomaly detection. Data is partitioned by RMM instance ID and further clustered by the columns which will be the defining targets of a query. Table structure can be seen in Appendix B .

RMM instances when paired with a Cassandra node will have their generated data naturally partitioned by locality. This means each instance will process only the data generated by itself, unless assigned to another instance's data by the orchestrator. This reduces the need for data shuffling inside the Cassandra cluster, increasing read speeds and decreased the load on the system.

5.2.6 Orchestrator

For the first release of the RMM a single instance implementation was the priority, with load balancing and parallelization efforts planned for release 2. As such, the orchestrator was not implemented. Nevertheless, much of the RMM's instance implementation can be re-used in the orchestrator. The Message Handler can be re-used fully as the same functionalities used to receive and verify messages for a single instance of the RMM are the same for the Orchestrator. Storage access is also implemented, requiring minor changes to accommodate the orchestration data. Instance monitoring can also be performed through RPC messages sent through RabbitMQ, functionality already available and implemented in the RMM.

5.3 Integration Work

A fully integrated PoSeID-on platform prototype was planned for September 2019, and as result most of the early RMM development effort was put into the necessary technologies for this purpose. In order to deploy PoSeID-on all modules needed to be deployed in a straightforward and easy way and work out of the box, with little to no configuration from the platform administrator deploying it. In addition, the platform must be deployed in Kubernetes, in order to allow cloud deployment.

As such, a great part of the RMM development included a deep dive into Kubernetes configuration and deployment as well as Docker, the technology chosen to containerize each module. In order to deploy the RMM and Cassandra storage in a seamless manner in Kubernetes, Minikube, a local installation of Kubernetes was used to test module integration and deployment. Several scripts were written to automate the build, containerization and deployment steps of the module.

5.3.1 Development Environment

The development environment for the whole project was setup on a local machine in the datacenter of the University of Coimbra. Minikube was run on this machine, and project partners were given access to it in order to develop their own modules.

5.3.2 Module Configuration

Module configuration and deployment is done completely through developed scripts and Kubernetes customization files. The following files were developed:

Makefile: A makefile was created to build all necessary dependencies and packaging them into an uber-jar which can be submitted as a Spark job. This make file also pulls the latest docker image for ubuntu and builds the container which will run the RMM instance.

Docker File: A docker file was created to install the necessary dependencies into the container which will run the RMM instance and setup default environment variables used by the RMM.

Kubernetes Customization Files: Kubernetes configuration files were created to setup the deployment of the RMM instances and a Cassandra operator, used to manage the Cassandra cluster. The files setup RMM and Cassandra nodes discovery and the necessary port forward to access developer tools such as the Spark Dashboard and Cassandra CQL shell.

Deployment Scripts: Initial deployment scripts for the whole platform were shared by Jibe the project partner in charge of integration efforts. These scripts needed to be extended for the specific deployment of the RMM. This initial script targeted Linux as it was the common platform agreed by partners at the beginning of the project. As an additional effort, since the author of this thesis chose by personal preference to develop using Windows, an additional script was created for Windows deployment, through Windows Linux Subsystem. This was a valuable contribution as one of the project partners was later required to deploy the platform in Windows OS.

5.3.3 Message Protocol

The message protocol defined for PoSeID-on communication was implemented and tested successfully in the RMM and was proven completely functional by exchanging messages between the RMM and the PDA.

The structure for communication through the RabbitMQ was defined by Jibe to achieve the following objectives:

- End-to-end encryption for all the data exchanged in PoSeID-on
- Authentication of the data transported
- Verification of the data transported
- Identification of the sender and recipient of a message
- Concealment of the sender from all entities but the recipient
- Identification of message type (one- or two-way communication)
- Serialization of message parameters

These objectives were achieved by defining a protocol that specifies the following:

Member identification, authentication and verification

This was achieved through the use of asymmetric encryption, certificates and the use of a certificate authority (CA). The technology used for encryption was NaCl (Salt)[66], a collection of modern, verified cryptographic primitives abstracted into easy to implement, usable components. For identification of parties, an asymetric key pair is generated using Ed25519 signature scheme, which is later signed by the CA. The members public key is combined with the CA signature through Protobuf, resulting in the certificate used for identification.

Queue routing rules

The queue definition for each instance participating in PoSeID-on is based on the recipient's public certificate. A SHA-256 hash is calculated over the certificate binary and presented as a decimal string, which serves as the queue name for this recipient. The except to this rule are data subjects, for which messages are delivered to the Dashboard queue and then routed by the Dashboard to the end-user, to avoid the creation of a gigantic amount of queues.

For Remote Procedure Calls (RPC), two queues are necessary since queues are only one way, as such when an RPC message is sent a correspoding queue is declared, uniquely named for the response and using the AMQP 0-9-1 flag.

Packet format

The packet format consists of consecutive layers with signatures and encryption where needed. Protobuf, as explained previously is used to define the packet format. The following figure shows a graphical representation of this format:

Figure 5.2 Graphical Representation of the Message Packet

First layer contains the recipient's certificate, unencrypted for identification purposes within the Dashboard, together with an encrypted message body, sealed using Libsodiums sealed boxes. This encryption scheme uses X25519 and XSalsa20-Poly1305 for encryption. The X25519 key used is derived from the ED25519 key found in the recipient's certificate.

The encrypted body contains a payload and a signature over that payload. This signature is also created using libsodium's public-key signature and the sender's private Ed25519 key.

Inside the payload, the certificate of the sender can be found, to verify the payloads authenticity. The payload itself, can be of two message types, direct message or RPC message. A direct message contains the message identifier, such as the id respective to a permission request, permission grant, etc, and a set of parameters structured as ordered key-value pairs.

5.3.4 Integration Testing

Unit and Integration testing for the RMM has not be formally defined or implemented as the envisaged component Technology Readiness Level is TRL 6 and focus on complete code coverage is not required. Regardless, an initial test setup has been developed to cover the communication protocol implementation.

Currently automated testing covers message decryption and encryption, certificate signing and validation, payload signature and validation and finally message signature and validation.

5.4 Challenges

During the development of the module several challenges arose which delayed or affected to some extend the modules development.

Choice of Message Queue tools: the choice of RabbitMQ as the message queue system determined the usage of Spark Streaming, instead of the latest Spark Structured Stream framework, which allows for a larger set of operations possible over a data stream. This happened since there are no open-source connectors implemented for Spark Structured Streaming and implementing and testing a connector from scratch would substantially increase the workload – and that was acceptable from the project point of view. An outdated RabbitMQ connector was used with Spark Streaming and even this connector had to be modified and built locally to work with the Spark Streaming 2.4.0.

Data Sources Definition: data sources definitive decisions (by part of the project consortium) did not happen until late December 2019, and even then plans changed from having log messages being received directly from the Blockchain through Smart Contract event listeners to having everything delivered through the message bus, rendering the significant effort spent researching into blockchain technologies not applicable in the RMM.

Integration and trials are delayed: although the RMM has been fully integrated into PoSeID-on and the communication protocol tested and verified by communicating with the PDA and the Dashboard, several partner modules have suffered severe delays and by the time of delivery of this thesis (January 2020) the blockchain has not yet been deployed in the PoSeID-on platform. Furthermore, the Dashboard and pilot's Data Processor implementation are still not completely functional – and what is functional does not produce RMM-usable logs yet. As result, the RMM has no real data in transit to analyze, and there are no perspectives of real logs being set to the RMM soon. In order to test the RMM data analysis pipeline a public log dataset had to be used, as discussed in Chapter 6 .

5.5 Summary

This chapter presents the development and integration work performed in the scope of the Risk Management Module. First, component status is presented, in light of the proposed architecture. The Message Handler and Speed Processor have been completed, as well as part of Risk Manager. The Batch Processor development has not started although the groundwork for the data access and pipeline setup has been done. Afterwards, integration efforts are described, with emphasis on the developed scripts which automate the deployment and setup of the PoSeID-on platform and the RMM. Finally, implementation challenges and solutions, when available, are described.

Chapter 6 Evaluation

The work developed during this thesis falls between Software Engineering work and Research and Development, as there was not a single focus throughout the thesis. Activities ranged from pure architectural definition and design, to research into the machine learning area of expertise and implementation of the main module considering the PoSeID-on project priorities. An innovation in the anomaly detection field was not expected or set as an objective. The objective was to set up a module within PoSeID-on that can apply existing technologies for the purpose of protecting data subjects and their PII.

In this perspective, to evaluate the results of this thesis, thesis objectives, module requirements and attributes and experimental results for the anomaly detection pipeline must be considered.

6.1 Evaluation Strategy and Challenges

To evaluate the RMM, firstly the experimental setup, challenges and preliminary results of the anomaly detection pipeline will be presented. Secondly, fulfilled requirements and quality attributes will be discussed, taking in consideration what has been designed and what has been implemented accordingly. Finally, PoSeID-on project milestones for the interim version of the RMM will be analyzed, as well as the challenges associated with them.

6.1.1 Experimental Setup and Evaluation

Evaluation of the RMM anomaly detection had a few challenges. First of all, by the time of evaluation, no real data was circulating within PoSeID-on, as already explained in Section 5.4 . An alternative had to be found to test the RMM. In order to test the module, a public dataset was chosen, based on the type and structure of the logs.

Furthermore, the staging environment, running in Minikube and used to deploy the PoSeID-on platform, suffered several setbacks during the time of testing, from mid-December 2019 to early January 2020. Due to power-shortages in the UC datacenter (where the machine running the staging environment was placed) and development issues with project partners, the staging environment was down for a majority of the time. The partner in charge of managing the staging environment (Jibe) did not have any available resources to fix this due to the vacation period. In addition, the deployment script was missing a key aspect for making the Minikube available to external access, and as such the UC team could not setup the environment by itself. Due to these circumstances, tests were run on the same machine but not in Minikube.

Finally, Spark Streaming, as opposed to Spark Structured Streaming, does not allow for complex windowing operations, such as windowing data by timestamp. As such, the adopted dataset had to be tested by creating a script to send logs one by one to the RMM, according to the timestamp. This emulated real working conditions, as logs arrive in the interval that they would have arrived in a real-scenario and be windowed accordingly.

The following sections describe the experimental setup and achieved results.

6.1.1.1 Dataset

To the best of the authors knowledge, there are no datasets available from PII management systems. As such, an alternative had to be found. By using an alternative dataset, we cannot determine if the results obtained will be similar in the PoSeID-on context, but we can validate the anomaly detection approach and pipeline, since the log anomaly detection approach is generic. The chosen HDFS dataset was made available by the LogPAI research team [66], and its characteristics are summarized in Table 6.1.

Description: Hadoop Distributed File System log set is generated in a private cloud environment using benchmark workloads, and manually labelled through handcrafted rules to identify the anomalies. The logs are sliced into traces according to block ids. Then each trace associated with a specific block id is assigned a ground-truth label: normal/anomaly	
Time Span	38.7 Hours
Number of Messages	11 175 629
Number of Blocks	575 061
Number of Normal Blocks	558 223 (97,1% of blocks)
Number of Anomalous Blocks	16 838 (2,9% of blocks)

Table 6.1 HDFS Dataset Characteristics

The characteristics of the chosen dataset have three important factors:

- **It is a distributed system:** logs are generated by several distributed components.
- **There is an identifier associated with each log:** as with PoSeID-on, logs have an identifier corresponding to the block each operation is referring to, and a single block can be the target of multiple operations or logs, much like PoSeID-on will have an identifier associated to a Data Subject or Data Processor.
- **Anomalies are labelled according to the identifier:** the dataset is labelled in respect to blocks, such as a window of the RMM will label a window of logs corresponding to a data subject or data processor as anomalous or not.

These factors allow for the logs to be analyzed using the same approach that the RMM uses for PoSeID-on system logs.

6.1.1.2 Machine and Deployment

The machine where the RMM was deployed was the same machine where the staging environment was running. A Cassandra and a RabbitMQ node were run on the same machine, to deliver the logs through the message queue, as would happen in the PoSeID-on deployment and to save results.

Applications run on this machine remained the same for all runs of the experiments. The characteristics of the machine are provided in Table 6.2.

Brand and Model	Dell Precision 5820 Tower X-Series
Operating System	Ubuntu 18.04.1 LTS
Processor	Intel(R) Core (TM) i9-7920X CPU @ 2.90GHz
Random Access Memory	132 GB, 2666 MHz, DDR4
Graphics	NVIDIA Quadro P1000, 4 GB GDDR5
Storage	1TB 7200rpm SATA Hard Drive

Table 6.2 Machine Characteristics

The RMM was packaged into a Java Archive (JAR) file containing all necessary dependencies and ran with the Java Virtual Machine (JVM) configuration presented in Table 6.3.

Parameter	**Value**
Maximum memory allocation (-Xmx)	64 Gigabytes
Initial memory allocation (-Xms)	64 Gigabytes

Table 6.3 Java Virtual Machine Parameters

In addition, Spark was run in local cluster mode and as such the enforced configuration is of one Driver, with the Driver performing all the execution (instead of an Executor). The parameters set for running the RMM in local mode are summarized in Table 6.4.

Parameter	**Value**
Spark Logical CPU Cores	24
Driver Memory	48 Gigabytes

Table 6.4 Spark Local Cluster Configuration

6.1.1.3 Evaluating Drain Log Parser implementation

To evaluate the implemented Java version of Drain Log Parser, several runs were conducted were the outputs of the parsing were compared to the original Drain implementation results. This was done with a sub-set of the HDFS dataset, containing two thousand logs, also made available by the LogPAI team. The configuration for Drain was the one used in the original Drain paper[54] to parse the same sub-set of data, as presented in Table 6.5.

Parameter	Value
Log Format	“<Date> <Time> <Pid> <Level> <Component>: <Content>”
Regex List	BLOCK_ID = "blk_(|-)[0-9]+" IP = "(/|)([0-9]+\\.){3}[0-9]+(:[0-9]+|)(:|)" NUMBER = "(?<=[^A-Za-z0-9])(\\-?\\+?\\d+)(?=[^A-Za-z0-9])|[0-9]+$"
Similarity Threshold	0.5
Depth of Tree	4
Maximum Childs	100

Table 6.5 Drain Configuration

The results obtained from the implemented Java version, already included in the RMM and running inside the anomaly detection pipeline, were the same as the original implementation, as shown in Table 6.6.

Log Template	Occurrences
BLOCK* ask <*> to delete <*>	2
Receiving block <*> src: <*> dest: <*>	292
Verification succeeded for <*>	20
BLOCK* NameSystem.allocateBlock: <*> <*>	115
<*> Starting thread to transfer block <*> to <*>	1
BLOCK* ask <*> to delete <*> <*> <*> <*> <*> <*> <*> <*> <*>	1
BLOCK* NameSystem.addStoredBlock: blockMap updated: <*> is added to <*> size <*>	314
<*> Served block <*> to <*>	80
BLOCK* ask <*> to replicate <*> to datanode(s) <*>	1
<*>Got exception while serving <*> to <*>	80
BLOCK* ask <*> to delete <*> <*> <*> <*> <*> <*> <*> <*> <*> <*> <*> <*> (…)	2
BLOCK* NameSystem.delete: <*> is added to invalidSet of <*>	224
Received block <*> src: <*> dest: <*> of size <*>	2
Deleting block <*> file <*>	263
PacketResponder <*> for block <*> terminating	311
Received block <*> of size <*> from <*>	292

Table 6.6 Drain Java Implementation Results

6.1.1.4 Evaluating Anomaly Detection approach

In order to evaluate the anomaly detection approach, tests were run using a single RMM instance, under the conditions described in section 6.1.1 . The full HDFS dataset was used for testing.

As explained previously, since Spark Streaming does not allow windowing by timestamp. For this reason, a script was run to read the HDFS dataset file and send each log one by one, according to its timestamp, to the message queue. The RMM instance fetched this information every 5 seconds from the queue. Since the dataset spanned over approximately 39 hours, in order to minimize testing times, a speed-up parameter was added to the instance. For all testing cases, this speed-up was of 4 times faster than the real time. As such, all time-based operations ran 4 times faster. Messages were sent 4 times faster to the queue and processing happened 4 times faster than the parameters defined for the test. Although this increases the load on the system, it was a necessary trade-off for shortening testing times. The result should not have major differences, since all time-based operations are scaled accordingly.

Several parameters need to be set in order to test the RMM instance. The K-Means algorithm parameters need to be set, as well as the Java Descriptive Statistics initial parameters, and general pipeline configuration. In order to test how the number of clusters would affect results, the RMM instance deployed four models, with increasing number of clusters.

For K-Means the parameters provided in Table 6.7 were used to initialize the models.

Parameter	**Value**
Cluster initialization	Random
Number of Clusters Per Model	4, 8, 12, 16
K-Means Half Life	24 Training Batches

Table 6.7 K-Means Initialization

As for the starting clusters centers, these were set using a random seed. The same seed was used for all models running inside the same RMM instance, for result comparison.

Regarding the Descriptive Statistics setup, the 100 000 most recent distances were saved for each cluster of each model. The parameters for K-Means initialization and Descriptive Statistics were kept constant across all tests. Since it is not possible to be sure that a predicted anomaly is in fact an anomaly or not, all received logs were used for training of the model.

The parameters that suffered variations during testing were those related to the pipeline and anomaly detection approach, more specifically:

- **Whether or not feature vector overlap was considered for training.** Since sliding windows are used for prediction, it is possible to train the model with the feature vector calculated for every slide or to train with the feature vector for an actual window, avoiding repetition of logs.

- **What percentage of points a cluster must have to be considered an anomalous cluster.** Since small clusters are hypothesized to contain anomalies, a threshold must be set to define what is a small cluster. This is done by defining a percentage of the total points to be considered anomalous. If the cluster as less than that percentage of points, it is considered anomalous and a point predicted to be in that cluster is considered anomalous.
- **The Z-Score threshold.** How many standard deviations a vector's distance to the cluster center must be over the cluster mean in order to be considered an anomaly.
- **The window length.** What window is considered for training and analysis. Shorter windows will be able to detect point anomalies, while larger windows will detect collective anomalies.

Sliding length is another potential parameter, but since it is necessary to have predictions output under a user session, a fixed parameter was chosen (5 minutes) as it is the predicted time a user would spend on the platform, so it was not considered for evaluation. The chosen parameters for evaluation can be seen in Table 6.8.

Test	Overlapping	Anomaly %	Z-Score Threshold	Window (Minutes)	Training Frequency (Minutes)	Anomalies Considered
T01	Yes	3	2.5	30 minutes	40 minutes	Yes
T02.1	No	3	2.5	30 minutes	40 minutes	Yes
T02.2	No	3	2	30 minutes	40 minutes	
T03.1	No	3	2.5	60 minutes	70 minutes	Yes
T03.2	No	3	2	60	70	Yes

Table 6.8 Parameters Tested

The results of the tests took in consideration correction of anomalies detected. Since sliding windows of 5 minute are used, an anomaly detected in the first 5 minutes of a collection of logs in respect to certain ID may be considered an anomaly, but as more time passes and more logs for that ID are received, it may prove to be a normal pattern of logs. As such, only the latest prediction for an ID is considered.

First, tests were run to test whether models trained with overlapping feature vectors achieved good results. As expected, this proved to be false, as models trained with the result of sliding windows become over trained due to the repetition of anomalous patterns. The models simply failed to detect any anomalies as every data point was considered normal, as seen in Table 6.9, test T01. As result, no further testing was done with overlap training.

Due to still unresolved issues with Spark Streaming's garbage collection, in the RMM implementation for model training without overlapping of training windows, it was not

possible to analyze the full dataset as with T01. Regardless, results were collected for partial analysis of the dataset, corresponding roughly to 30% of the full set. These results can also be seen in Table 6.9.

Test	Cluster	TP	FP	TN	FN	P	R	F1
T01	K=4	0	0	558223	16838	NaN	0	NaN
	K=8	0	0	558223	16838	NaN	0	NaN
	K=12	0	0	558223	16838	NaN	0	NaN
	K=16	0	0	558223	16838	NaN	0	NaN
T02.1	K=4	624	0	102697	4726	**1**	0.116	0.207
	K=8	624	0	102697	4726	**1**	0.116	0.207
	K=12	324	0	102697	5026	**1**	0.060	0.113
	K=16	331	0	102697	5019	**1**	0.061	0.114
T02.2	K=4	2691	409	122694	3294	0.868	**0.449**	**0.591**
	K=8	1624	409	122694	4361	0.798	0.271	0.404
	K=12	2360	407	122696	3625	0.852	0.394	0.538
	K=16	1624	410	122693	4361	0.798	0.271	0.404
T03.1	K=4	444	0	104273	4983	1	0.081	0.149
	K=8	444	0	104273	4983	1	0.081	0.149
	K=12	328	0	104273	5099	1	0.060	0.113
	K=16	328	0	104273	5099	1	0.060	0.113
T03.2	K=4	1508	8	102689	3842	0.994	0.281	0.438
	K=8	638	6	102691	4712	0.990	0.119	0.212
	K=12	1503	6	102691	3847	0.996	0.280	0.437
	K=16	1506	6	102691	3844	0.996	0.281	0.438

Table 6.9 Test Results

As sowhn in Figure 6.1, a combination of a Z-Score threshold of 2 and a window of 30 minutes achieved the best F1-Score for a model with 4 clusters.

It is also possible to see that the increase in the number of clusters seems to affect tests negatively in most cases, which is an indicator that using a smaller number of clusters may be the best option performance and result wise.

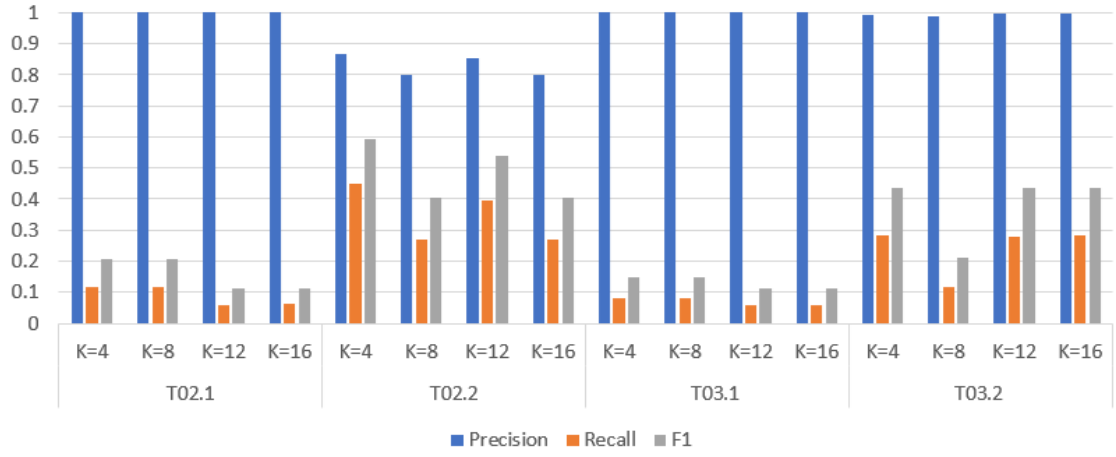

Figure 6.1 Graphical Comparison of Experimental Tests

No more tests were carried out before the writing of this document, due to the tests not being conclusive for the real use case of the RMM, as data collected is inherently different as is the natural windows over which anomalies occur. Regardless, these tests helped validating the anomaly detection approach and resulted in two useful conclusions:

- Training with overlapping data will produce worse results, due to over-training with anomalies.
- The anomaly detection approach works for the implemented speed layer.

In addition, tests with the full HDFS dataset brought up problems with garbage collection and memory management, which root cause is still under evaluation.

6.1.2 Fulfilled module requirements and quality attributes

Looking at the requirements and quality attributes previously shown in Table 4.1 and Table 4.2, respectively, it is possible to make an analysis of the status of the RMM at the end of this thesis – as illustrated in Figure 6.2.

Requirement ID	Title	Requirement Fulfillment Status
R1	RMM Data Sources	Has been fulfilled, all defined data-sources are considered, and information is successfully collected and used for analysis from them.
R2	RMM Risk Detection	Has been fulfilled, albeit the proposed design has not been completely implemented. Regardless the module already detects anomalies and is ready for testing with real data.
R3	RMM Risk Events	Has been fulfilled. Results are created and kept in an external database.
R4	RMM Notifications	Has been fulfilled. Current defined channels for notifications (Message Queue) are used successfully, as is the identification of the ID's to whom the anomaly is in respect to.
R5	RMM PII in Risk Events	All parameters passed in a log to the RMM can be consulted with proper authorization, as is any possible PII sent to it.
R6	RMM Risk Event Storage	Has been fulfilled by saving results in an external database which supports encryption at rest and in transit.
R7	RMM Data Processor Reputation	This requirement has not been fulfilled. Reputation scores are not being created yet for identified data processors.
R8	RMM DP Reputation	This requirement has not been fulfilled. Reputation scores are not being created yet for identified data processors.
R9	RMM Risk Notification Delivery	Has been fulfilled since the communication protocols and ID identification for this have been implemented.
R10	RMM Risk Prediction	Has been partially fulfilled since reputation is not taken in consideration yet.

Figure 6.2 Requirement Fulfilment Status

From these requirements only R7 and R8 have not been fulfilled at least partially. This is due to the fact that real data is still not available in PoSeID-on and the dataset used for testing does not have the same ID correlation as is expected in the PoSeID-on system. Block ID's are unique as opposed to Data Processor ID's, so it is not possible to create a history of operations in order to associate a score to results obtained for a block ID. Regardless, the implementation of these metrics will be a minor effort as all underlying requirements, such as database access and anomaly detection by ID have been implemented.

As described in Table 6.10, Fault-Tolerance has not been implemented yet, and as such has not been fulfilled. Multiple instances are supported by the designed architecture, but the orchestrator will be implemented only in the final release of the RMM.

Quality Attribute	Attribute Refinement	ASR ID	Quality Attribute Status
Performance	Latency	QA1	Has been fulfilled, results are provided every 5 minutes.
Availability	Fault-tolerance	QA2	Has not been implemented yet but Spark and the Orchestrator design should accommodate for this.
Scalability	Multiple Instances	QA3	The module has been designed to work with multiple instances, but current implementation does not support this yet, although efforts have been made so the underlying pipeline supports loading of models.
Privacy	PII Privacy	QA4	Has been fulfilled, only the strictly necessary amount of data is collected, the database supports time-to-live on data and deletion of PII from the system can be done since all entries can be deleted without affecting the module.
Security	Data Security	QA5	Has been fulfilled all secure communication protocols have been implemented.

Table 6.10 Quality Attributes Fulfilment

All technical constraints, as shown in Table 4.3, have been respected.

6.1.3 Fulfilled PoSeID-on milestones

In addition to the evaluation criteria already identified, in this section we dig into the PoSeID-on project milestones involving the Risk Management Module (during the timeframe of this thesis), in order to assess its success in the scope of the project. Table 6.11 identifies all relevant milestones.

Activities / Milestone	Due Date	Description
Deliverable D2.2	31/12/2019	PoSeID-on System Requirements & Architecture Design
RMM Exploratory Studies & Detailed Design	28/02/2019	Detailed RMM design and implementation plan
RMM Component Implementation - Interim	31/07/2019	Initial module implementation
Deliverable D4.3	31/07/2019	Report on RMM and PDA interim implementation
1st Integrated Platform Evaluation	30/09/2019	Preliminary prototype ready to be deployed in the pilots

Table 6.11 PoSeID-on Milestones

All milestones were accomplished, successfully and on time, with the possible exception of the last one: while there is an integrated platform with the first release of each PoSeID-on component, it is still not fully ready for trials with real users due to issues related with the blockchain components, which are to be solved by Tecnicalia. Moreover, the first formal review of the PoSeID-on project, which took place in Brussels in late October 2019, already evaluated and approved all technical outcomes related with this thesis (especially the architecture design and the interim RMM implementation).

6.1.4 Publications

Even though the publication of research papers was not a direct objective of this thesis, the way the work evolved led to a few publications, in addition to the already mentioned co-authorship of project deliverables. More specifically, the following publications were produced:

- Casaleiro, R. and Paulo Silva and Simões, P. and Boavida, F. and Edmundo Monteiro and Marilia Curado and Tiago Cruz and Nuno Antunes and Marco Vieira and Riccio, G.M. and Verzillo, M.P. and Marek, P. and Goncalves, L. and Bagnato, A. and Valentini, A. and Intonti, B. and Manzo, R. and Posta, V.D. and Zampolini, L. and Rooij, J.v. and Houf, R. and Rios, E. and Iturbe, E. and Gutierrez, I. and Anguita, S. and Gomez, C. and Echevarria, J. and Houf, H. and Nicoletti, L. and Lotti, R. and Natale, D. and Pizzo, L.d. and Pane, F. and Schiavo, F. , "*Protection and control of personal identifiable information: The PoSeID-on approach*", (accepted in) Journal of Data Protection & Privacy, vol. 3, 2020.

  This paper provides a first overview on the PoSeID-on's concept and its architecture design. Since we were the lead editor of Deliverable D2.2 and one of the lead moderators of the discussions that lead to the definition of the architecture, we were also the first author and one of the main contributors to this collective paper.

- Casaleiro, Rui, Silva, Paulo, Simões, Paulo, Antunes, Nuno, Curado, Marilia, Monteiro, Edmundo, Boavida, Fernando, "Gestão e Análise de Riscos na Plataforma de Proteção de Dados Pessoais POSEIDON", (accepted in) "Congresso Luso-Moçambicano de Engenharia" (CLME2020).

This dissemination paper describes the PoSeID-on concept, with a focus on the components provided by University of Coimbra (i.e. the RMM and the Personal Data Analyzer).

- P. Silva, R. Casaleiro, P. Simões, N. Antunes, M. Curado, E. Monteiro, F. Boavida, "Risk Management and Privacy Violations Detection in PoSeID-on's Data Privacy Platform", submitted to the Annual Privacy Forum 2020 (under review). This technical paper provides an overview of the PoSeID-on concept and more detailed content about the RMM and the Personal Data Analyzer. The Privacy Forum is one of the more relevant European fora addressing data and personal privacy.

In addition to those publications, a more ambitious paper specifically addressing the RMM is also planned, once PoSeID-on trials enable more extensive evaluation studies.

6.1.5 Fulfilled thesis objectives

By looking at the objectives introduced in Section 0 and the analysis presented in the previous sections, it is possible to conclude that all objectives initially set out for this thesis have been achieved. The contributions given to the PoSeID-on project resulted in the accepted deliverables. The RMM design supports the technical constrains and requirements, and the prototype version of the RMM was successfully integrated in the PoSeID-on platform. The module accomplishes most of the objectives required for the module, as expected for the interim version, and the implemented approach was tested and validated with a synthetic dataset – even though the constraints induced by the lack of real usage of the platform so far (due to issues outside our direct responsibility) leaves a bittersweet feeling regarding full blown demonstration of the RMM potential.

6.2 Summary

This chapter presented the evaluation strategy for the module and work developed in this thesis. First the experimental setup was described, and results were presented, in order to validate the implemented anomaly detection approach. Afterwards functional requirements fulfilment status for the current implementation of the RMM was presented, as well as the status for quality attributes. Finally, PoSeID-on milestones and their fulfillment is analyzed as well as the accomplished thesis objectives.

Chapter 7 Conclusions and Future Work

7.1 Conclusions

The work developed during this internship focused on delivering a proof-of-concept for the Risk Management Module, a PoSeID-on component which is intended to analyze system logs and PII transaction logs in order to provide platform administrators and data subjects (whom personal data is being exchanged) with actionable information regarding the safety of their information.

This work began by contributing to the design and development of the initial PoSeID-on architecture and the refinement of each module's characteristics, objectives and scope. In parallel, the state of the art of anomaly detection techniques was studied, as well as the basis for blockchain technologies and their variants. The General Data Protection Regulation was also explored in order to guarantee that the decisions made for the RMM and the PoSeID-on platform would comply with the regulation.

With that preliminary steps gathered and documented, a design for the RMM was proposed, taking in consideration all the requirements for the module in the context of the project and the available tools for materializing the proposed design. A proof-of-concept was developed in accordance to the design proposed, and testing of the RMM revealed that the proposed design – even if only partially implemented, since the internship timespan does not entirely cover the project timespan – already achieves moderately accurate results using a synthetic dataset. This validates the RMM concept and, once real data starts circulating in the platform, its implementation can be further refined to fully achieve the purpose of the module within the PoSeID-on system.

During the internship, all project milestones involving the UC team and the author's work were successfully fulfilled and the author provided valuable contributions for this purpose. Furthermore, considering the work described in the previous chapters, it is possible to conclude that all objectives initially conceived for this internship were successfully achieved. There were several obstacles and challenges along the way but these were overcome with perseverance and ultimately culminated in the work that was presented here.

In appendix, a detailed work plan can be found (Appendix A), where the detailed management of this work is reported, as well as obstacles encountered during the period of its development and their resolution.

The work developed, while not delving deep into pure research work, created the opportunity to acquire a different set of knowledge and new interests as this area of expertise, anomaly detection and machine-learning, was relatively unknown to the author. This, combined with the exploration of tools that are in demand (such as Kubernetes, Spark, and RabbitMQ) and the management and soft-skills developed through the participation in the PoSeID-on's project consortium proved a very valuable outcome for the author, both academically and professionally.

7.2 Future Work

Despite the successful proof-of-concept implemented as result of this internship, there are still some work that needs to be done in different areas of the Risk Management Module and PoSeID-on project. Most of the future work tagged as *future* is already considered in the overall project timespan (that goes beyond this internship).

Regarding the PoSeID-on and the RMM status, the orchestrator and parallel deployment of RMM instances still need to be implemented, as well as the designed purpose for the batch layer. A reputation system needs to be developed and tested, once real data is circulating in PoSeID-on. The documentation, implementation and this internship report provide valuable inputs for this task.

Bibliography

[1] "Protection and control of Secured Information by means of a privacy enhanced Dashboard » PoSeID-on." [Online]. Available: https://www.poseidon-h2020.eu/. [Accessed: 21-Dec-2019].

[2] "Encryption | General Data Protection Regulation (GDPR)." [Online]. Available: https://gdpr-info.eu/issues/encryption/. [Accessed: 13-Dec-2018].

[3] "Increasing value of personal data a 21st century challenge." [Online]. Available: https://www.computerweekly.com/news/252452162/Increasing-value-of-personal-data-a-21st-century-challenge. [Accessed: 21-Dec-2019].

[4] "Spark Streaming - Spark 2.4.4 Documentation." [Online]. Available: https://spark.apache.org/docs/latest/streaming-programming-guide.html. [Accessed: 21-Dec-2019].

[5] "Protection and control of Secured Information by means of a privacy enhanced Dashboard Deliverable 4.3 Risk Management Module & Personal Data Analyser Interim implementation." [Online]. Available: https://www.poseidon-h2020.eu/wp-content/uploads/2019/08/D4.3-RMM-and-PDA-V1.0-Final.pdf. [Accessed: 20-Jan-2020].

[6] F. B. Rui Casaleiro, Paulo Silva, Paulo Simões, Nuno Antunes, Marília Curado, Edmundo Monteiro, "Protection and control of personal identifiable information: The PoSeID-on approach," *Journal of Data Protection & Privacy*, vol. 3, no. 2, pp. 1–34, 2019.

[7] "EUR-Lex - 32016R0679 - EN - EUR-Lex." [Online]. Available: https://eur-lex.europa.eu/eli/reg/2016/679/oj. [Accessed: 21-Dec-2019].

[8] F. B. Paulo Silva, Rui Casaleiro, Paulo Simões, Nuno Antunes, Marília Curado, Edmundo Monteiro, "Risk Management and Privacy Violations Detection in PoSeID-on's Data Privacy Platform."

[9] "GDPR FAQs – EUGDPR." [Online]. Available: https://eugdpr.org/the-regulation/gdpr-faqs/. [Accessed: 13-Dec-2018].

[10] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System."

[11] D. Yaga and D. Yaga, "Blockchain Technology Overview Blockchain Technology Overview."

[12] "Proof of Work vs Proof of Stake: Basic Mining Guide - Blockgeeks." [Online]. Available: https://blockgeeks.com/guides/proof-of-work-vs-proof-of-stake/. [Accessed: 21-Dec-2019].

[13] "Ethereum." [Online]. Available: https://ethereum.org/. [Accessed: 21-Dec-2019].

[14] "Hyperledger – Open Source Blockchain Technologies." [Online]. Available: https://www.hyperledger.org/. [Accessed: 21-Dec-2019].

[15] "Smart Contracts." [Online]. Available: http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html. [Accessed: 21-Dec-2019].

[16] R. A. Ariyaluran Habeeb, F. Nasaruddin, A. Gani, I. A. Targio Hashem, E. Ahmed, and M. Imran, "Real-time big data processing for anomaly detection: A Survey," *International Journal of Information Management*, vol. 45. Elsevier Ltd, pp. 289–307, 01-Apr-2019.

[17] D. M. Hawkins, *Identification of Outliers*. Dordrecht: Springer Netherlands, 1980.

[18] M. Ahmed, A. Naser Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, Jan. 2016.

[19] R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning For Network Intrusion Detection."

[20] M. Zhang, B. Xu, and J. Gong, "An Anomaly Detection Model Based on One-Class SVM to Detect Network Intrusions," in *2015 11th International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*, 2015, pp. 102–107.

[21] M. Ahmed and A. N. Mahmood, "Novel Approach for Network Traffic Pattern Analysis using Clustering-based Collective Anomaly Detection," *Annals of Data Science*, vol. 2, no. 1, pp. 111–130, Mar. 2015.

[22] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, "Unsupervised real-time anomaly detection for streaming data," *Neurocomputing*, vol. 262, pp. 134–147, 2017.

[23] V. Chandola, "Anomaly Detection : A Survey," 2009.

[24] N. S. Arunraj, R. Hable, and M. Fernandes, "Comparison of Supervised, Semi-supervised and Unsupervised Learning Methods in Network Intrusion Detection System (NIDS) Application," *Anwendungen und Konzepte der Wirtschaftsinformatik*, no. 6, 2017.

[25] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network Anomaly Detection: Methods, Systems and Tools," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 303–336, Spring 2014.

[26] M. Ahmed, A. Naser Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60. Academic Press, pp. 19–31, 01-Jan-2016.

[27] Y. Freund, R. E. Schapire, Y. Singer, and M. K. Warmuth, "Using and combining predictors that specialize," in *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing - STOC '97*, 1997, pp. 334–343.

[28] T. G. Dietterichl, "Ensemble learning." 2002.

[29] P. Casas, J. Vanerio, and K. Fukuda, "GML learning, a generic machine learning model for network measurements analysis," in *2017 13th International Conference on Network and Service Management (CNSM)*, 2017, pp. 1–9.

[30] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, Aug. 1996.

[31] Y. Freund and R. E. Schapire, "Experiments with a New Boosting Algorithm," 1996.

[32] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, Jan. 1992.

[33] P. Casas, "MLNET - Machine Learning Models for Network Analytics," *undefined*, 2018.

[34] A. Töscher, M. Jahrer, and R. M. Bell, "The BigChaos Solution to the Netflix Grand Prize," 2009.

[35] Y. Yamato, H. Kumazaki, and Y. Fukumoto, "Proposal of Lambda Architecture Adoption for Real Time Predictive Maintenance," in *2016 Fourth International Symposium on Computing and Networking (CANDAR)*, 2016, pp. 713–715.

[36] P. Mulinka and P. Casas, "Stream-based Machine Learning for Network Security and Anomaly Detection," in *Proceedings of the 2018 Workshop on Big Data Analytics and Machine Learning for Data Communication Networks - Big-DAMA '18*, 2018, pp. 1–7.

[37] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, "Unsupervised real-time anomaly detection for streaming data," *Neurocomputing*, vol. 262, pp. 134–147, Nov. 2017.

[38] T. Qin, X. Guan, W. Li, P. Wang, and Q. Huang, "Monitoring abnormal network traffic based on blind source separation approach," *Journal of Network and Computer Applications*, vol. 34, no. 5, pp. 1732–1742, Sep. 2011.

[39] P. Casas, F. Soro, J. Vanerio, G. Settanni, and A. D'Alconzo, "Network security and anomaly detection with Big-DAMA, a big data analytics framework," in *Proceedings of the 2017 IEEE 6th International Conference on Cloud Networking, CloudNet 2017*, 2017.

[40] M. Mdini, A. Blanc, G. Simon, J. Barotin, and J. Lecoeuvre, "Monitoring the network monitoring system: Anomaly Detection using pattern recognition," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2017, pp. 983–986.

[41] B. Aaron, D. E. Tamir, N. D. Rishe, and A. Kandel, "Dynamic Incremental K-means Clustering," in *2014 International Conference on Computational Science and Computational Intelligence*, 2014, pp. 308–313.

[42] E. Bigdeli, M. Mohammadi, B. Raahemi, and S. Matwin, "Incremental cluster updating using Gaussian mixture model," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2015, vol. 9091, pp. 264–272.

[43] "Messaging that just works — RabbitMQ." [Online]. Available: https://www.rabbitmq.com/. [Accessed: 05-Jan-2020].

[44] "Protocol Buffers | Google Developers." [Online]. Available: https://developers.google.com/protocol-buffers. [Accessed: 05-Jan-2020].

[45] "Introduction - Libsodium documentation." [Online]. Available: https://libsodium.gitbook.io/doc/. [Accessed: 31-Dec-2019].

[46] "Home - Open Containers Initiative." [Online]. Available: https://www.opencontainers.org/. [Accessed: 05-Jan-2020].

[47] "Production-Grade Container Orchestration - Kubernetes." [Online]. Available: https://kubernetes.io/. [Accessed: 05-Jan-2020].

[48] "minikube." [Online]. Available: https://minikube.sigs.k8s.io/. [Accessed: 05-Jan-2020].

[49] "Setup — Graylog 3.1.0 documentation." [Online]. Available: https://docs.graylog.org/en/3.1/pages/auditlog/setup.html. [Accessed: 05-Jan-2020].

[50] "Protocol Buffers | Google Developers." [Online]. Available: https://developers.google.com/protocol-buffers. [Accessed: 31-Dec-2019].

[51] J. Zhu *et al.*, "Tools and benchmarks for automated log parsing," *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice*, pp. 121–130, 2019.

[52] M. Mizutani, "Incremental mining of system log format," in *Proceedings - IEEE 10th International Conference on Services Computing, SCC 2013*, 2013, pp. 595–602.

[53] M. Du and F. Li, "Spell: Streaming Parsing of System Event Logs," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*, 2016, pp. 859–864.

[54] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: An Online Log Parsing Approach with Fixed Depth Tree," in *2017 IEEE International Conference on Web Services (ICWS)*, 2017, pp. 33–40.

[55] N. Marz and J. (James O.) Warren, *Big data : principles and best practices of scalable real-time data systems*. .

[56] P. Yang, C.-J. Hsieh, and J.-L. Wang, "History PCA: A New Algorithm for Streaming PCA," Feb. 2018.

[57] M. Grabowska and W. Kotłowski, "Online principal component analysis for evolving data streams," in *Communications in Computer and Information Science*, 2018, vol. 935, pp. 130–137.

[58] N. Marz and J. (James O.) Warren, *Big data : principles and best practices of scalable real-time data systems*. .

[59] Y. Yamato, H. Kumazaki, and Y. Fukumoto, "Proposal of Lambda Architecture Adoption for Real Time Predictive Maintenance," in *2016 Fourth International Symposium on Computing and Networking (CANDAR)*, 2016, pp. 713–715.

[60] "Apache Spark™ - Unified Analytics Engine for Big Data." [Online]. Available: https://spark.apache.org/. [Accessed: 05-Jan-2020].

[61] "Spark Streaming | Apache Spark." [Online]. Available: https://spark.apache.org/streaming/. [Accessed: 05-Jan-2020].

[62] "Apache Flink: Stateful Computations over Data Streams." [Online]. Available: https://flink.apache.org/. [Accessed: 05-Jan-2020].

[63] "GitHub - Stratio/spark-rabbitmq: RabbitMQ Spark Streaming receiver." [Online]. Available: https://github.com/Stratio/spark-rabbitmq. [Accessed: 05-Jan-2020].

[64] "Apache Cassandra." [Online]. Available: https://cassandra.apache.org/. [Accessed: 05-Jan-2020].

[65] "NaCl: Networking and Cryptography library." [Online]. Available: https://nacl.cr.yp.to/. [Accessed: 19-Jan-2020].

[66] "LogPAI - Log Analytics Powered by AI." [Online]. Available: http://www.logpai.com/. [Accessed: 12-Jan-2020].

Appendix

Appendix A Workplan

This appendix presents the expected and real workplan followed during this internship. As mentioned in the previous sections, during this thesis it was necessary to manage two different set of goals, the goals set for the Risk Management Module and the milestones of the PoSeID-on project. These goals were not always in alignment, which led to the need of prioritize some task in the determent of others. This will be detailed in this appendix, in addition to challenges found which influenced the work developed and an analysis of the expected workplan versus the real workplan.

PoSeID-on Milestones

The PoSeID-on project has a strict set of milestones which needed to be taken in consideration while developing this thesis. The milestones that directly influenced this thesis are the following, as also seen in Section 5.4 :

Activities	Milestones	Description
D2.2 - Deliverable Submission	31/12/2019	PoSeID-on System Requirements and Architecture Design
Exploratory Studies & Detailed Design	28/02/2019	Detailed RMM design and implementation plan
RMM Implementation - Interim	31/07/2019	Initial module implementation
D4.3 - Deliverable Submission	31/07/2019	Report on RMM and PDA interim implementation
1st Integrated Platform Evaluation	30/09/2019	Preliminary prototype ready to be deployed in the pilots

Table A.1 PoSeID-on Milestones (Reminder)

During the first semester there was only one milestone directly influencing the Risk Management Module, which was the delivery of D2.2 the document specifying the architectural description and requirements. UC was the leader of this task and the majority of the work of compiling information and writing of this deliverable was designated to the author of this work, as the team member most involved the technical part of the project at that time, due to being the developer for the RMM.

During the second semester the milestones aligned with this thesis's goals as the task being developed was task T4.2, the development and reporting of the interim version of the RMM. Implementation was to continue till the end of June when focus changed into writing deliverable D4.3, Risk Management Module and Personal Data Analyzer Interim Implementation, containing the description of the interim version of the RMM and the PDA.

What deviated from the thesis goals during the second semester was the necessity of having an integrated platform as soon as possible before September, in order to accommodate the interim platform analysis in the end of September and allow at least a month of integrated development using a staging environment. With this, priorities had to switch from developing the anomaly detection pipeline to focusing on implementing the communication protocol and deploying the module on the staging environment.

First Semester

The workplan for the first semester, as seen in Figure 1.1, did not take in account the time necessary to participate in the PoSeID-on project related activities such as the participation in the discussion of system architecture and requirements and writing of the respective document.

Task Name	Task ID	Sub-task	Start date	End Date	Duration
State-of-art Study	T1.1	Blockchain Technology	17/09/2018	28/09/2018	12
	T1.2	General Data Protection Regulation	29/09/2018	02/10/2018	4
	T1.3	Anomaly Detection Study	03/10/2018	17/10/2018	15
Study of PoSeID-on Project	T2.1	PoSeID-on Description of Work	18/10/2018	03/11/2018	17
RMM Architectural Design	T3.1	Requirements Specification	04/11/2018	10/11/2018	7
	T3.2	Architectural Design	07/11/2018	17/11/2018	11
	T3.3	Methodology Definition	04/11/2018	17/11/2018	14
Implementation and evaluation	T4.1	Framework Exploration	17/11/2018	30/11/2018	14
	T4.2	Initial Implementation	25/11/2018	25/12/2018	31
Thesis Writing	T5.1	Intermediate Report	29/09/2018	31/12/2018	94

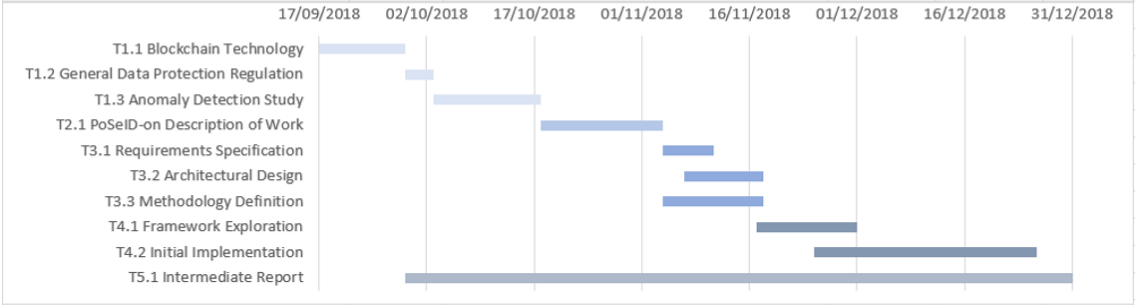

Figure A.1 First Semester Workplan

Furthermore, the deployment and testing of a permissioned blockchain solution using Hyperledger was also not in the plan (Task T1.4 in Figure A.2), as it was something that stemmed from the discussion of possible frameworks to be used as the permissioned blockchain choice in PoSeID-on. For the previous reasons, in addition to the necessity to define clearly the state of the whole PoSeID-on platform, as the initial concepts were very open-ended, resulted in a delay of the anomaly detection and feature extraction study, which had to be delayed to the second semester.

Task Name	Task ID	Sub-task	Start date	End Date	Duration
State-of-art Study	T1.1	Blockchain Technology	17/09/2018	28/09/2018	12
	T1.2	General Data Protection Regulation	29/09/2018	02/10/2018	4
	T1.3	Anomaly Detection Study	22/10/2018	21/11/2018	31
	T1.4	Hyperledger Fabric Network Implementation	08/10/2018	21/10/2018	14
Study of PoSeID-on Project	T2.1	PoSeID-on Description of Work	02/10/2018	15/10/2018	14
RMM Architectural Design	T3.1	Requirements Specification	03/11/2018	10/11/2018	8
	T3.2	Architectural Design	07/11/2018	17/11/2018	11
	T3.3	Methodology Definition	03/11/2018	17/11/2018	15
Implementation and evaluation	T4.1	Framework Exploration	17/11/2018	17/12/2018	31
Thesis Writing	T5.1	Intermediate Report	01/11/2018	43487.00	83
PoSeID-on Architectural Definition	T6.1	Participation in weekly discussions	15/10/2018	27/12/2018	74
	T6.2	Writing deliverable D2.2	20/10/2018	27/12/2018	69
PoSeID-on Activities	T7.1	October General Assembly	23/10/2018	26/10/2018	4

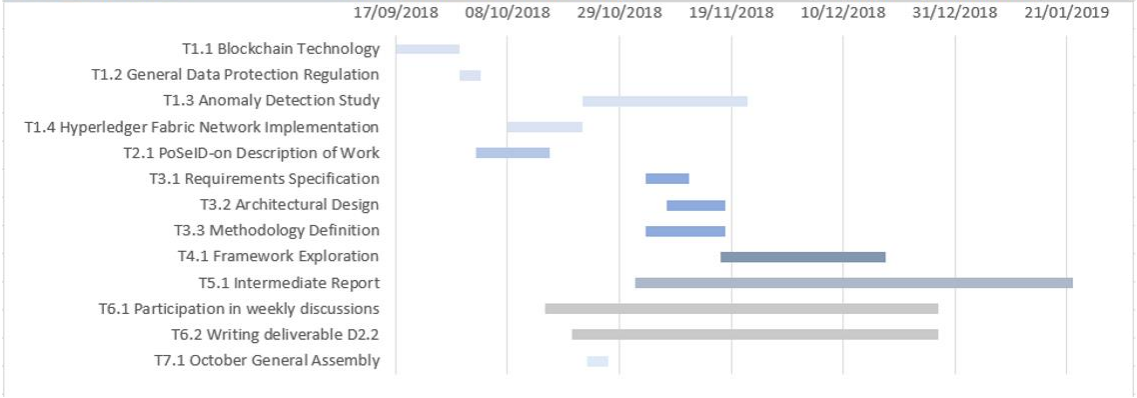

Figure A.2 First Semester Effective Work

Initial implementation of the module was also postponed to the second semester, as development tasks for PoSeID-on platform as a whole did not start until mid-February. Figure A.2 shows the effective schedule for the first semester, including the tasks that were not account for in the initial planning.

Second Semester

For the second semester, the workplan can be seen in Figure A.3. During the beginning of the second semester it was necessary to determine what would be the integration constraints and RMM data sources as it was still not clear what information each module would be providing to the RMM and what technology stack would be used to do so. This was necessary in order to start implementing the module and to explore the possibilities for feature extraction. Afterwards, the expected work was to configure the chosen frameworks and start the implementation of the anomaly detection pipeline based on that information, followed by the writing of deliverable D4.3. Once implementation of the anomaly detection pipeline was complete as well as the definition of the dataset for testing, validation of the module and writing of this thesis were the following tasks.

Task Name	Task ID	Sub-task	Start date	End Date	Duration
Identification of Implementation Constraints	T7.1	Integration Constraints	24/01/2019	15/02/2019	23
Definition of the RMM data sources	T8.1	Data source identification and discussion	24/01/2019	15/02/2019	23
	T8.2	Feature extraction study and definition	16/02/2019	28/02/2019	13
Configuration of Frameworks and Libraries	T9.1	Spark, Apache Kafka, Protobuf, Libsodium	01/03/2019	07/03/2019	7
Implementation of RMM prototype	T10.1	Anomaly Detection Stream Pipeline	08/03/2019	01/04/2019	25
	T10.2	Anomaly Detection Batch Pipeline	02/04/2019	01/05/2019	30
Testing and Validation	T11.1	Dataset preparation	02/05/2019	15/05/2019	14
	T11.2	Validation of Results	16/05/2019	31/05/2019	16
Writing	T12.1	Thesis	01/06/2019	30/06/2019	30
	T12.2	Deliverable D4.3	01/06/2019	30/06/2019	30

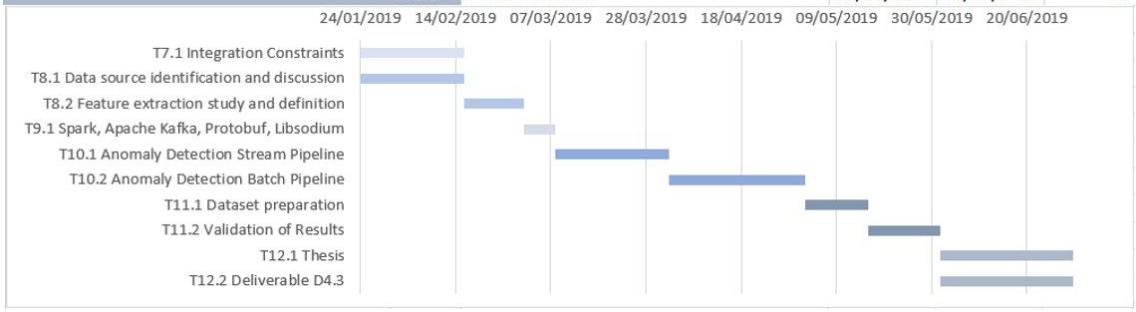

Figure A.3 Second Semester Workplan

Progress did not go exactly as expected, as can be seen in Figure A.4. Since priorities shifted to implementing the integration aspects of the module, more time was spent on configuration and implementation of the message protocol and respective frameworks. Deployment stack such as Minikube, Kubernetes and Docker also needed to be configured and studied. Another obstacle was the switch of the initial thought out choice for the messaging system, from Apache Kafka to RabbitMQ, which created some difficulties while trying to connect Spark Streaming to the message bus and with data processing as explained in Section 5.4 resulting in longer configuration and implementation times for the spark connector.

Task Name	Task ID	Sub-task	Start date	End Date	Duration
Identification of Implementation Constraint	T7.1	Integration discussions	24/01/2019	15/02/2019	23
Definition of the RMM data sources	T8.1	Data source identification and discussion	24/01/2019	15/02/2019	23
	T8.2	Feature extraction study and definition	15/02/2019	28/02/2019	14
Configuration of Frameworks and Libraries	T9.1	Apache Spark Streaming	01/03/2019	07/03/2019	7
	T9.2	RabbitMQ and RabbitMQ to Spark connector	08/03/2019	12/03/2019	5
	T9.3	Docker and Docker Compose	13/03/2019	18/03/2019	6
	T9.4	Minikube and Kubernetes	19/03/2019	26/03/2019	8
Implementation of RMM prototype	T10.1	Message Broker (Encryption & Message Protocol	08/04/2019	09/05/2019	32
	T10.2	Message Parsing and Spark Plumbing	10/05/2019	17/05/2019	8
	T10.3	Drain Log Parser in Java	18/05/2019	09/06/2019	23
	T10.4	Feature vector extraction	10/06/2019	17/06/2019	8
Testing and Validation	T11.1	Datasets study and preparation	01/05/2019	15/05/2019	15
	T11.2	Unit testing frameworks study and configuration	27/03/2019	29/03/2019	3
Writing	T12.1	Thesis	29/07/2019	31/08/2019	34
	T12.2	Deliverable D4.3	01/07/2019	22/07/2019	22
PoSeID-on Activites	T13.1	General Assembly April	03/04/2019	05/04/2019	3
	T13.2	General Assembly July	10/07/2019	10/07/2019	1
	T13.3	Privacy Workshop Participation	11/07/2019	11/07/2019	1
	T13.4	Integration Session	05/08/2019	08/08/2019	4
	T13.5	PoSeID-on Architecture Article Writing	18/06/2019	30/06/2019	13

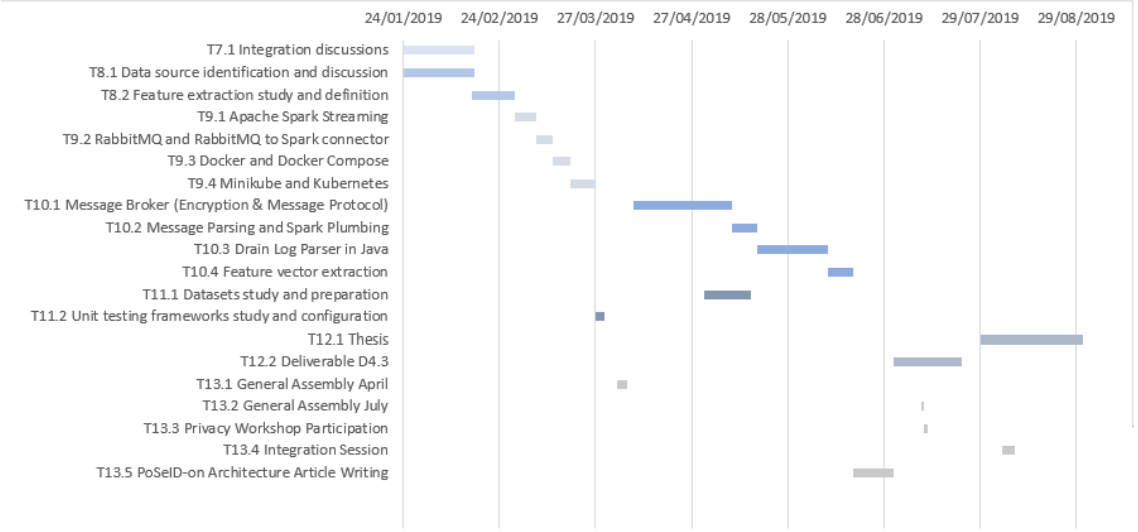

Figure A.4 Second Semester Effective Work

Development of the feature extraction step of the module also took more time than expected as the algorithm chosen in order to create a feature vector from system log messages was not available in Java. In addition, in order to allow for a horizontally scalable RMM, this algorithm had to be adjusted to work with Spark Streaming and allow for the re-use and sharing of the parsing tree used by the algorithm. Participation in several PoSeID-on related activities as seen in task 13.1 to 13.5 were also not accounted for, which resulted in less time to finish the anomaly detection implementation. These activities were necessary, thought, as an integrated platform was the priority at the time and effort also resulted in the articles accepted for publication.

Extra Semester

As result of the project priority changes and the delays in integration, the author made the decision of delaying the delivery of this work for an extra semester, both to accomplish the objectives decided upon for this thesis fully before delivery but also to contribute to the

project milestones set for September and consolidate the integration works without neglecting either of the responsibilities. This resulted in the following workplan, for the effective work accomplished:

Task Name	Task ID	Sub-task	Start date	End Date	Duration
Persistence of data and results	T14.1	Persist feature vectors and parsed data	10/10/2019	17/10/2019	8
Finalize simple stream anomaly detection	T15.1	Cluster model building and prediction	18/10/2019	31/10/2019	14
	T15.2	Predict anomaly using cluster distance and cluster size	01/11/2019	25/11/2019	25
Anomaly detection approach testing using HDFS	T16.1	Evaluate preliminary results using time based approach	25/11/2019	30/11/2019	6
Refine anomaly detection approach	T17.1	Evaluate preliminary results using time and ID based approach	01/12/2019	10/01/2020	41
Writing	T20.1	Thesis	08/11/2019	17/01/2020	71
PoSeID-on Activites	T21.1	Interim Version Evaluation	23/10/2019	26/10/2019	4
	T21.2	Development of the integrated storage solution and deployment scripts	02/09/2019	10/10/2019	39

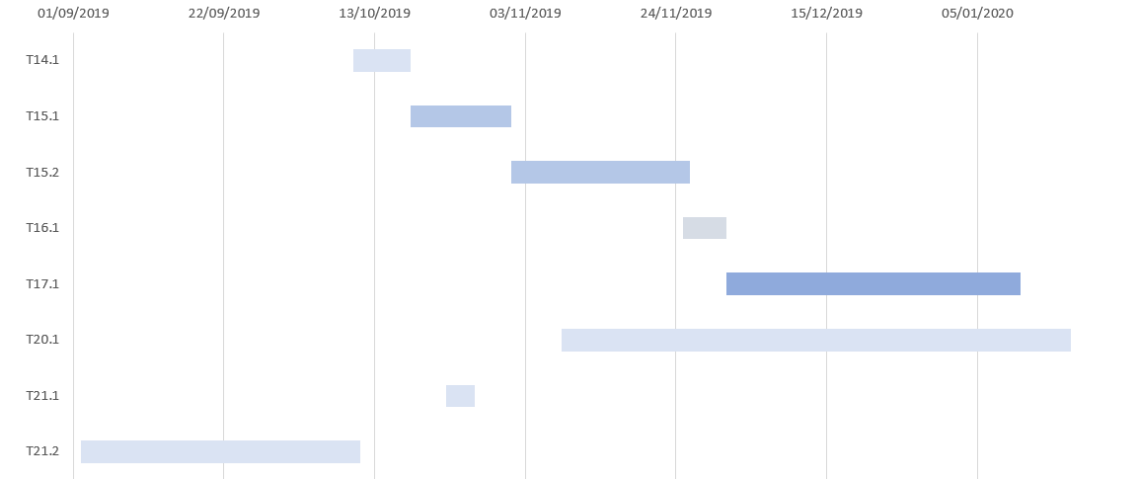

Figure A.5 Extra Semester Effective Work

Appendix B Storage Table Structure

This appendix presents the table structure for the stored data produced by the Risk Management Module and the description of the purpose of each field.

Key Type	Column	Description
Partition Key	Instance ID	ID of the RMM instance that generated this entry
Clustering Key	Window ID	Unique ID of this window of data
Clustering Key	Window Time	Time of processing
Clustering Key	Grouping ID	ID used for further windowing or grouping of logs, Such as Data Subject, Data Processor or Module ID
	Feature Vector	The normalized feature vector that originated this result
Clustering Key	Label	The label of the result, either Anomaly or Normal
	Entity IDs	List of ID's involved in this log window.
Clustering Key	Algorithm	String describing the approach used for this result.
	Algorithm Parameters	List of "Key:Value" strings for the initialization parameters of the algorithm
	Result Parameters	List of "Key:Value" strings used to save relevant information regarding the result, such as the rational leading to the result (e.g. if an anomaly was considered as such due to the cluster size of the predicted cluster, or due to Z-score being over the threshold)

Table B.1 Results Table

Key Type	Column	Description
Partition Key	Instance ID	ID of the RMM instance that generated this entry
Clustering Key	Window ID	Unique ID of this window of data
Clustering Key	Window Time	Time of processing
Clustering Key	Grouping ID	ID used for further windowing or grouping of logs, Such as Data Subject, Data Processor or Module ID
	Total Events	Total count of events, or logs in this window
	Event Vector	The non-normalized feature vector
	Event Vector Schema	The feature vector schema at the time of the result. Each index contains a template/event ID, or a placeholder flag, to zero-pad the vector to a fixed size
	Entity IDs	List of ID's involved in this log window

Table B.2 Feature Vector Table

Key Type	**Column**	**Description**
Partition Key	Instance ID	ID of the RMM instance that generated this entry
Clustering Key	Event ID	Unique ID generated for this event
	Event Template	Template generated by Drain. This template is updated as Drain parses more logs but the table entry and unique ID is maintained.

Table B.3 Template Table

Key Type	**Column**	**Description**
Partition Key	Instance ID	ID of the RMM instance that generated this entry
Clustering Key	Window ID	Unique ID of this window of data
Clustering Key	Window Time	Time of processing
Clustering Key	Message ID	Unique ID of the parsed log.
	Process ID	ID of the process generating the log.
	Severity	Syslog level extracted from the log
Clustering Key	Log Timestamp	Timestamp extracted from the log
	Parameter List	List of parameters extracted from the log (variable parts of the log not included in the main structure defined previously in Drain)

Table B.4 Parsed Log Table