UNIVERSITÄT KOBLENZ · LANDAU

Fachbereich 4: Informatik

# Engineering Data Protection-Aware Health Data Intelligence Systems by Design

## Masterarbeit

zur Erlangung des Grades Master of Science (M.Sc.)
im Studiengang Informatik

vorgelegt von

Martin Hollmann

Erstgutachter: Dr. Marco Konersmann
(Institut für Softwaretechnik, AG Jürjens)

Zweitgutachter: Dr. Qusai Ramadan
(Institut für Softwaretechnik, AG Jürjens)

Koblenz, im August 2022

# Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

| | Ja | Nein |
|---|---|---|
| Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden. | ☒ | □ |

Koblenz, 17.08.2022

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

(Ort, Datum) (Unterschrift)

# Acknowledgements

I would like to thank very much Dr. Marco Konersmann and Dr. Qusai Ramadan who kindly acted as my supervisors and provided me with valuable feedback and advice during this project. My warmest thanks also go to my family in general for their continued support, and Elisabeth and Michael in particular for proofreading this thesis. All remaining mistakes are entirely my own.

## Abstract

The project "KI und Covid" by the Institute for Software Engineering at Koblenz University plans the development of a so called Health Data Intelligence System (HDIS) which is supposed to provide authorities with comprehensive insight into the main and side effects of pandemic events and the actions taken to address them in order to optimize their crisis response measures. To this end, the system will use various kinds of data generated and provided by public and private sources. As the name suggests, this data will also include health data which often concerns individual persons and is considered private data under European and German data protection law [1, 2].

To use this data legally, information systems in general and HDIS in particular have to conform with a large number of requirements. Data protection, however, is often treated as an afterthought during development, as identifying these requirements and finding ways to implement them into a system architecture by design can be a time consuming process that has to be conducted for the development of each individual system [3, 4]. With this thesis, we try to facilitate this process by creating a reference architecture that can serve as a reusable framework for ensuring basic compliance of any HDIS with data protection regulations and encourage the consideration of data protection at every stage of the system's development lifecycle.

For this, we compile a list of general data protection requirements and conduct a dedicated analysis of threats specific to HDIS. Lastly, we design a high-level reference architecture for such systems which takes data protection requirements and threat mitigation strategies into account. Our solution is a set of research artifacts that can be applied through a standardized process and aim to facilitate the inclusion of data protection into the development processes at the development stages of requirements engineering and architectural design.

# Contents

# Chapter 1

# Introduction

## 1.1 Motivation

The ongoing SARS-CoV-2 pandemic has confronted the world with novel problems due to its unprecedented nature in the context of globalization. While the successful use of digital tools in combating the pandemic like the Corona-Warn-App or the digital vaccine certificate has intensified the current drive for digitalization throughout all parts of European healthcare systems, that drive hasn't permeated the legislative system yet [5, 6, 7, 8]. Facing sudden and exponential increases in infection rates, governments were forced to make sweeping decisions based on very limited background knowledge as to their effectiveness and possible side effects, resulting in inconsistent containment policies and a continuous decrease of public acceptance of these decisions [9, 10].

The planned project *"KI und COVID: Erklärbarkeit und Entscheidungsunterstützung durch KI in Pandemie-Situationen"* (en. AI and COVID: Explainability and Decision Support by AI during Pandemic Events) by the Institute for Software Engineering at Koblenz University aims to improve the legislative process behind the making of containment policies for the current as well as future epidemic events. To achieve this, it proposes the design and development of an AI-driven decision support system for recommending and evaluating courses of action for populations, governments and healthcare services, based on various types of local and global data [11].

When designing such a system - which in the following we will call a *"Health Data Intelligence System" (HDIS)* - there are multiple aspects that need to be taken into account. Identifying the data protection aspects that need to be considered for the development of HDIS is of vital importance, as they are likely to rely on sensitive and personal data which must be protected from general disclosure and misuse. Laws and regulations regarding data protection and its various aspects like privacy and security are becoming ever more extensive in order to cover the ever-growing application space and can therefore be hard to grasp due to the accumulation of regional, national and international regulations. In addition, there are professional, data protection-related standards for specific aspects of AI-systems which may not be legally binding, but should still be considered as they provide a professional source for more specific requirements and solutions.

Identifying these standards and regulations and converting them into usable requirements for the design and implementation stages of HDIS is a time consuming process which currently has to be repeated for each new development process. Since systems that process sensitive data in general and health data in particular are likely to share a lot of common concerns with regards to data protection, it is our goal to facilitate these processes in order to reduce costs and development time as well as improve the quality of the final product. We address this challenge in three general steps. First, we propose a list of general data protection requirements that can be reused as a foundation for the requirements engineering process of each HDIS development project. Secondly, we analyze potential threats that are specific to HDIS. Lastly, we propose a reference architecture that serves as a general framework for including data protection into the overall system architecture by design.

## 1.2 Defining Data Protection

While the term *Data Protection* has been widely used for several decades now, it lacks a unified definition and requires some initial clarification. Its main application nowadays is in the context of public discourse as well as data protection policy, where it serves as an umbrella term for various aspects concerning the regulation and restriction of handling sensitive data, particularly data containing personal information [5]. These policies can either be legally binding, like national and international data protection legislation, or self-obligated as in the case of organizational data protection policy, and can differ widely in their exact definitions. Following these two categories, we distinguish between two notions of data protection for the purposes of our research. We define *Mandatory Data Protection* as the compliance with all applicable and legally binding requirements and restrictions regarding the collection, storing, handling and safekeeping of data. *Supplementary Data Protection* on the other hand, we define as all measures in the same context that go beyond the baseline of mandatory data protection.

## 1.3 Research Questions and Tasks

The overall objective of this thesis is to facilitate the requirements engineering and architectural design process for Health Data Intelligence Systems with regards to establishing data protection. To structure the process of achieving this, we have derived the following five research questions which we discuss further below, as well as the methodology employed to answer them:

**RQ.1** Which are the relevant sources for data protection requirements for Health Data Intelligence Systems?

**RQ.2** What are common data protection requirements for Health Data Intelligence Systems?

**RQ.3** What are potential risks of and threats to Health Data Intelligence Systems?

**RQ.4** How can these requirements be considered in a reference architecture?

**RQ.5** How can this reference architecture be applied?

### 1.3.1 RQ.1 – Sources for Data Protection Requirements

Useful and precise system requirements are usually discovered through a cooperative process including the developer, the customer, potential users, and other stakeholders. It involves various techniques to generate a set of requirements which describes the system as accurately and completely as possible. In our case, we specifically focus on data protection requirements. Since processing personal data in general and health data in particular is highly regulated by various layers of local and international law, basic legal compliance is the most important characteristic that we aim for in the finished requirements document. Thus, we search for sources containing legally binding requirements for Health Data Intelligence Systems, as well as any industrial standards they might refer to. Furthermore, we want our requirements to be applicable to all future implementations of HDIS as to not restrict the individual development teams more than necessary. Lastly, we want our requirements to be stable, meaning we focus on sources that retain their relevance and validity over the longest possible amount of time in order to reduce the need for constant revision of these requirements due to changes in the underlying sources.

To determine the relevance of individual sources to HDIS, we have to take inherent characteristics of such systems into account. Therefore, we first have to define as many of those characteristics as possible. Since the project is still in an early stage, our main source for this is the project proposal [11]. The main goal of this task is to create a rough outline of the intended system, corresponding processes and data flows. Since it might be necessary to make assumptions in cases where essential characteristics are still undefined, it is important to point those out and give a rationale for each. The resulting system outline will be important for all subsequent steps of this thesis as well, as the answers to all our research questions are to some degree dependent on the systems inherent characteristics.

Afterwards, we begin our search for requirements sources. To achieve the three qualities of legal relevance, ubiquity and stability, we conduct a comprehensive review of relevant legal sources and up-to-date technical standards. Thematically we focus on best-practices targeting the development of AI and data-intensive systems as these characteristics are both vital for HDIS as well. Considering the scope of the main project, we focus on German and European Union jurisdiction when covering legal documents, starting out from the General Data Protection Regulation and branching out into more specific laws and regulations. Besides our aspiration to provide thorough legal coverage, we prioritize stable sources, which will remain relevant for a considerable period of time. Ultimately, we aim to compile a list of laws and regulations that impose legally binding standards on our system, as well as technical and industrial sources that will aid us by concretize abstract legal requirements into more tangible technical and organizational requirements. It should be noted that we can't guarantee the list of legal requirements to be complete, but rather aim to provide a solid foundation of relevant documents that is supposed to be expanded and completed by law professionals where necessary.

### 1.3.2 RQ.2 – Extraction of Data Protection Requirements

The next step after identifying relevant sources is to extract the requirements they contain. Achieving and maintaining data protection and security is a complex task that goes beyond soft- and hardware systems. Legal obligations are usually abstract and often don't specify where and how they are supposed to be implemented. Thus we need to look beyond just defining system requirements but also consider organizational and process requirements that might result from the same law.

To extract actual requirements from the selected sources, we first review different methods and techniques from the field of requirements engineering in order to find the most suitable format with regards to structure and presentation of the requirements document. Since it is likely that the majority of requirements will be rooted in national and international law, we also consider techniques from the field of regulatory requirements engineering, which address the unique difficulties of translating abstract law into concrete requirements [12, 13]. We further search for adequate tools that could support the extraction process. Lastly, the identified methods and tools are applied to compile a set of data protection requirements which is supposed to provide the best possible balance between containing all necessary constraints while retaining as much flexibility as possible with regards to implementation approaches.

### 1.3.3 RQ.3 – Potential Risks and Threats to HDIS

Systems handling sensitive and personal data are always at risk of that data being illegally disclosed or otherwise compromised through malicious attacks or internal mishandling and errors during processing. It is therefore necessary to take potential types of attacks and vulnerabilities into account during the requirements engineering and design stages of development in order to mitigate costs and risks during operation. This is even more crucial since Health Data Intelligence Systems' outputs may influence decisions with far reaching consequences for people's lives and freedoms. Furthermore, from a legal perspective, the applicability of specific regulations to a system might depend on the risks associated with its operation. Thus, we need to analyze the Health Data Intelligence System's concept with respect to potential technical vulnerabilities and attack paths that should be considered during development, as well as its inherent qualities that might constitute a legal risk by making the system liable to possible violations of data protection law.

Taking all these different aspects into account, we analyze inherent threats to and weaknesses of Health Data Intelligence Systems by conducting a comprehensive analysis of our system model and its inherent qualities with regards to possible data protection liabilities. For this we consider legal sources like the General Data Protection Regulation (GDPR) and the Bundesdatenschutzgesetz (BDSG), as they not only contain legal requirements for such risk assessments, but also define specific aspects as inherently risky which might not appear as such from a purely development-based perspective [1, 2]. We then identify potential types of external threats and attacks against which the system has to be secured. For this, we consider current scientific research as well as reports and statistics of recent security incidents. This allows us to evaluate these threats with regards to their individual severity and consequent priority for consideration during development.

### 1.3.4 RQ.4 – Designing a Reference Architecture

Comprehensive data protection can not be achieved solely on an implementation level, but instead has to permeate all stages of development. Since optimal low-level solutions are often very specific to an individual system, we instead want to know how data protection can be achieved on the various abstraction levels of system architecture, as well as search for technologies and frameworks that can support its implementation.

To answer this question, in addition to a standardized requirements repository, we propose a high-level reference architecture for realizing the extracted data protection requirements following a modular building block system. To achieve this, we research state of the art-approaches to realize different subsets of requirements, taking into consideration established architectures and technologies as well as approaches in advanced stages of scientific validation. From this we aim to compile a framework of solutions that can be applied selectively by the developers during the design and implementation of concrete Health Data Intelligence Systems. We also want to present each solution's specific advantages and disadvantages to enable developers to make an informed decision if they want to include this solution into their implementation.

### 1.3.5 RQ.5 – Application of the Reference Architecture

Since we aim to create research artifacts that can be used in practical development, the question of validation and evaluation naturally comes up. Of course, an empiric evaluation of the benefits of our work to an actual HDIS development process is out of the scope of this thesis, as it can only be done during such a development process. Instead, we conduct a basic evaluation of our artifacts' usability and show how we intend them to be applied. For this, we apply our architecture to an exemplary system model consisting of various realistic, high-level components (e.g. data sources, data processing-systems, databases) and show how our results could be used in combination to design an actual data protection architecture on top of them.

## 1.4 Thesis Structure

The chapter structure of the thesis closely follows the research questions, dedicating a separate chapter to each. Chapter 2 covers the first task of RQ.1, where we establish the available baseline information about Health Data Intelligence Systems upon which we base all further work in the following chapters. Chapter 3 covers our search for requirements sources, first discussing available source categories before evaluating and selecting the best sources from these. After selecting our sources, we continue with RQ.2 by covering the extraction of the actual requirements in Chapter 4. For this we introduce some important general concepts of requirements engineering and describe our methodology before presenting and discussing the finished requirements repository. Chapter 5 contains the threat- and risk analysis with which we aim to address RQ.3. Again we first have to introduce some general terminology before presenting our research methodology. Afterwards we present the discovered threats, starting with those specific to the application of artificial intelligence before concluding the chapter with a discussion of more general threats to system security.

The actual reference architecture, which is the primary subject of RQ.4, is presented in Chapter 6. After quickly explaining its structure and the various documents constituting the architecture, we continue by discussing the relevance of different requirement categories to each type of system component. We then introduce the new reference architecture itself, as well as its various components and discuss how they improve data protection on a system if applied during its development. Chapter 7 covers RQ.5 by describing a methodology by which our various research artifacts could be applied by a development team in an actual design process. Lastly, Chapter 8 contains a final assessment of our results and to what degree we were able to answer each research question. We also discuss any aspects that could not be addressed in our research and could therefore motivate future research.

## Chapter 2

# Initial Specifications for Health Data Intelligence Systems

In this chapter we define some basic systemic and organizational characteristics Health Data Intelligence Systems are supposed to have. We base this on information provided by the KI und Covid project, in the context of which this thesis is situated. The project team provides a proposal as well as a schema (see Figure 2.1) showing how Health Data Intelligence Systems should be structured and operated, what input data they use, and what kind of output they are supposed to generate [11]. Since proposal and schema only define the general architecture on a very abstract level, there are aspects that are not yet specified but are important for technical and legal concerns regarding data protection. In such cases we extend the information provided in the schema with reasonable assumptions while trying to keep the general concept as broad as possible for later research and development. The technical terms related to data protection used during this description are partially adapted from the GDPR and BDSG to ensure an adequate mapping of concepts between our system outline and legal texts [1, 2]. Where necessary, the terms are supplemented by additional terms to account for HDIS-specific concepts that lack a legal equivalent.

In the following, we will define our system outline from multiple perspectives. First, we will consider basic characteristics in Section 2.1 establishing the purpose of HDIS and on what legal basis it is likely to operate. This is followed by Section 2.2, where we look at information about the organizational structure and stakeholders behind the system's operation. From there we continue with technical characteristics. Section 2.3 considers existing information on the types of processing that will take place within Health Data Intelligence Systems. Afterwards we establish the different types of data that will be used during those processings in Section 2.4 and finally combine both aspects in Section 2.5 by discussing the basic data flows that will likely be part of any implementation of HDIS.

## 2.1 Purpose and Legal Basis

According to Articles 6 and 9 of the GDPR, any processing of personal data is only legitimate in the presence of at least one out of a list of predefined circumstances specified by the regulation [2]. These can take various forms from voluntary publication or explicit consent by the Data Subject, a signed contract up to a legal mandate by the governing authority of an EU-member state. This is important as, depending on the respective legitimization, different legal

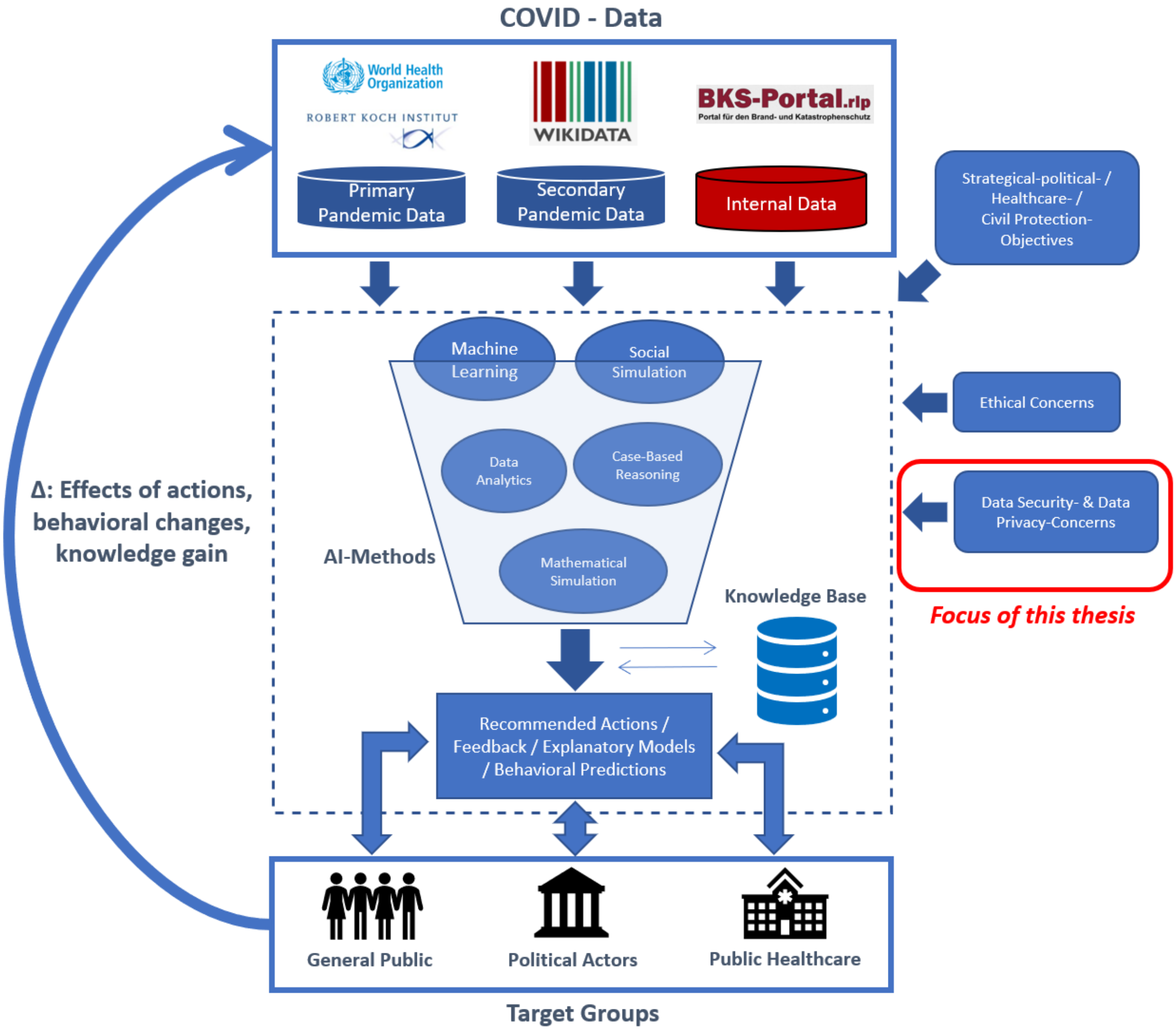


**Figure 2.1:** Solution approach proposed by the "KI und COVID"-project [11]. Three types of pandemic-related data (see 2.4) from various sources shall be used as input for an wide range of AI-related processing methods whose outputs are then combined to generate different kinds of valuable information for the public, as well as political and healthcare authorities.

requirements might apply. As the intended use of the HDIS is to support government and healthcare authorities in decision making processes, thus potentially influencing the lives of large numbers of people, we can assume that a legal mandate will have to be given beforehand to legitimize the operation.

## 2.2 Organization and Stakeholders

Operation of a Health Data Intelligence System will require the establishment of an organizational structure specifying personal as well as institutional responsibilities. As this topic is not addressed in the proposal but has significant implications on legal obligations, we will make some assumptions here as well.

First, we assume that operational oversight of the system will be held by a dedicated research institute or department of such an institute, acting as the Data Controller. In case of a legal mandate, this function is shared with the government body issuing the mandate, which most likely also takes the position of Supervisory Authority. Since the required resources for processing in the context of artificial intelligence depend heavily on the scope of data and the methods employed we prefer to err on the side of caution and assume some form of distributed computation and storage shared between one or more external facilities acting as Data Processors. For public data, procurement can be legally conducted by the controller or processors themselves through data mining. For personal data on the other hand, the data has to be actively provided by either the Data Subject itself or the current Data Holder, which might be a public institution like the Federal Statistical Bureau of Germany or the Robert Koch-Institute.

## 2.3 Processing

Since researching and designing the precise methods and processes of generating a useful output from the different data types described in Section 2.4 will only happen in later stages of the parent project, we have to content ourselves with assuming a high-level process that can be applied to the training and deployment of most model-based methods in artificial intelligence (see Figure 2.2). We first collect all required data and distribute it among the various processors. Those may then apply any forms of cleaning and preprocessing required for their specific method before applying the data to the respective models. The intermediate processing results are then used to train the model parameters and evaluate its quality. Finally, we aggregate the intermediate results to compile the final output and update our knowledge base.

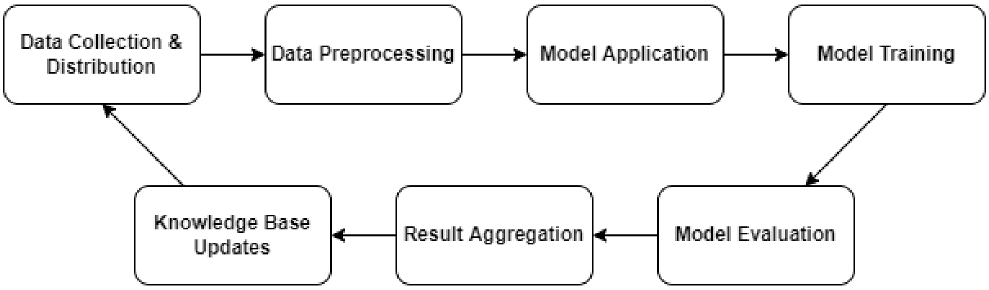


**Figure 2.2:** High-level diagram of the processes forming the pipeline for the artificial intelligence processing as we expect it to appear in Health Data Intelligence Systems.

## 2.4 Data Types

Since the technical data types used to transfer and process data within the Health Data Intelligence System are likely to be quite diverse, we will forego any assumptions with regards to them and instead focus on more abstract characteristics of the data (see Figure 2.3).

The first characteristic, which we label as *Context*, mostly concerns the raw data provided by the Data Holders [11]. It distinguishes between primary data which encompasses information that is directly related to the epidemic situation (e.g. incidence, hospitalization rate, vaccination rate, etc.) and secondary data which means information that doesn't inherently concern the epidemic, but can provide additional context and insight by combining it with primary data (e.g. population statistics, supply chain data, etc.).

Our second characteristic is *Privacy*. From this perspective, data will be considered public data if it can be accessed and used freely due to not containing any sensitive information that might result in personal and legal consequences in case of being compromised or unintentionally disclosed, such as public statistics provided by the Robert Koch-institute [9]. In contrast, we label data as private data if it is not openly available due to containing highly sensitive and/or personal information and therefore requires specific protection from loss, illegal alteration and unauthorized disclosure. Privacy is likely to be the most important data characteristic when considering data protection as the privacy status of a piece of data has direct implications on the design of any component handling it. Thus, if any system component handles both private and public data, we will consider all data handled by it to be private for the purpose of implementing that component in the final reference architecture.

The third and broadest characteristic addresses the *Maturity* of the data within the scope of the HDIS, meaning the state of processing the data is at. raw data encompasses unprocessed data that has been acquired from or provided by Data Holders. input data represents all preprocessed data that is then used for processing in one or more AI systems. The term knowledge is introduced to account for the intention to compile a so called knowledge base consisting of a dynamic ontology and other information generated during processing that can be used during future processing runs [11]. Lastly, we have two types of results. partial results are generated by individual processes and can be compiled into aggregated results which comprise all types of outputs that are supposed to be provided to the data recipients (e.g. recommended actions, feedback, predictions).

## 2.5 Data Flows and Distribution

For the operation of the Health Data Intelligence System, large quantities of data have to be transferred between stakeholders and system components. Figure 2.4 shows a high-level model of the expected data flows between involved entities. This model is not optimized towards data protection yet, but is rather supposed to establish all basic components that we believe are essential to all future implementations of Health Data Intelligence Systems and therefore can be considered in our research without imposing undue restrictions on developers. Even though they are represented as monoliths in this diagram, it is important to note that in practice the groups of Data Holders, recipients and processors are likely to each consist of multiple separate entities. We can expect every flow between parties to involve the crossing of a trust boundary by having to use non-exclusive mediums like the internet to transfer data.

We consider two kinds of trust boundaries, one for the Data Controller and one for each individual processor. Within these boundaries we assume all data transfer to be conducted via

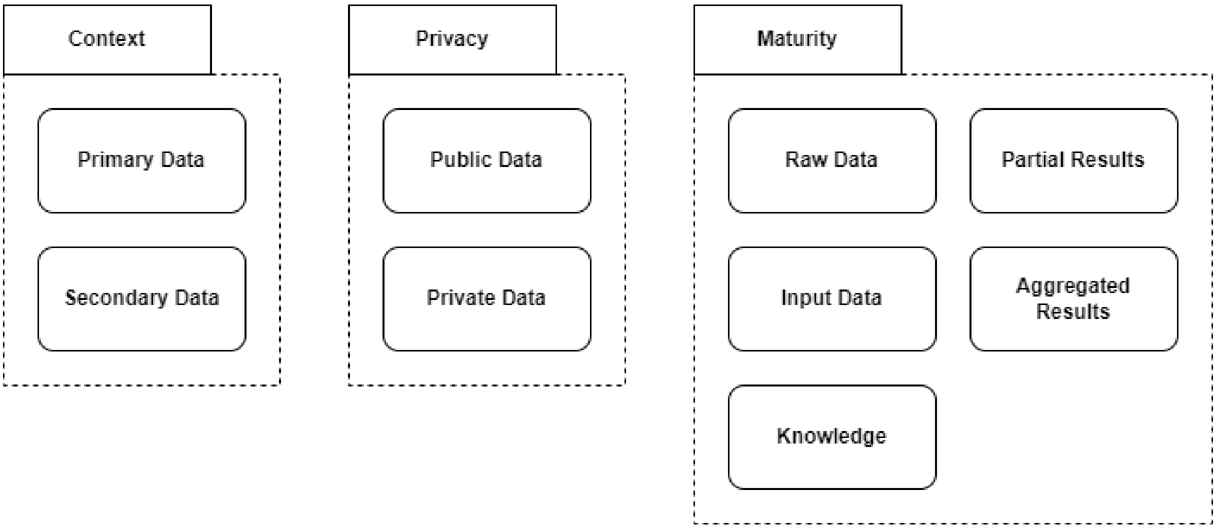


**Figure 2.3:** Characteristics and categories of data handled by the HDIS.

secure physical lines, whereas each data flow crossing between trust boundaries will likely be conducted via insecure public channels, presumably the internet.

Data Holders transfer raw data via the Public and Private Data Ingests to the Data Controller where it is stored in the controller's local database. The Controller Database can then be queried by the different processors for specific raw data which they need for processing. This data is then transferred over the C2P-Raw Data Transfer flow. Additionally, processors can request data from the Shared Knowledge Base, that has been generated during previous processing runs and is provided via the C2P-Historic Knowledge Transfer flow. All queried data is stored in the processor's local database before being fed through the Preprocessing Input into the Preprocessing System which in turn forwards it to the AI- and Mathematical Processing System using the Processing Input flow. All generated outputs are returned to the Processor Database through the Processing Output. The processor returns its partial results as well as any knowledge updates via the P2C-Partial Result and Knowledge Transfer to the controller's Shared Knowledge Base. From here, the aggregated results of all processors are then forwarded to the Result Recipient (e.g. government offices, public health authorities) using the Aggregated Result Transfer flow.

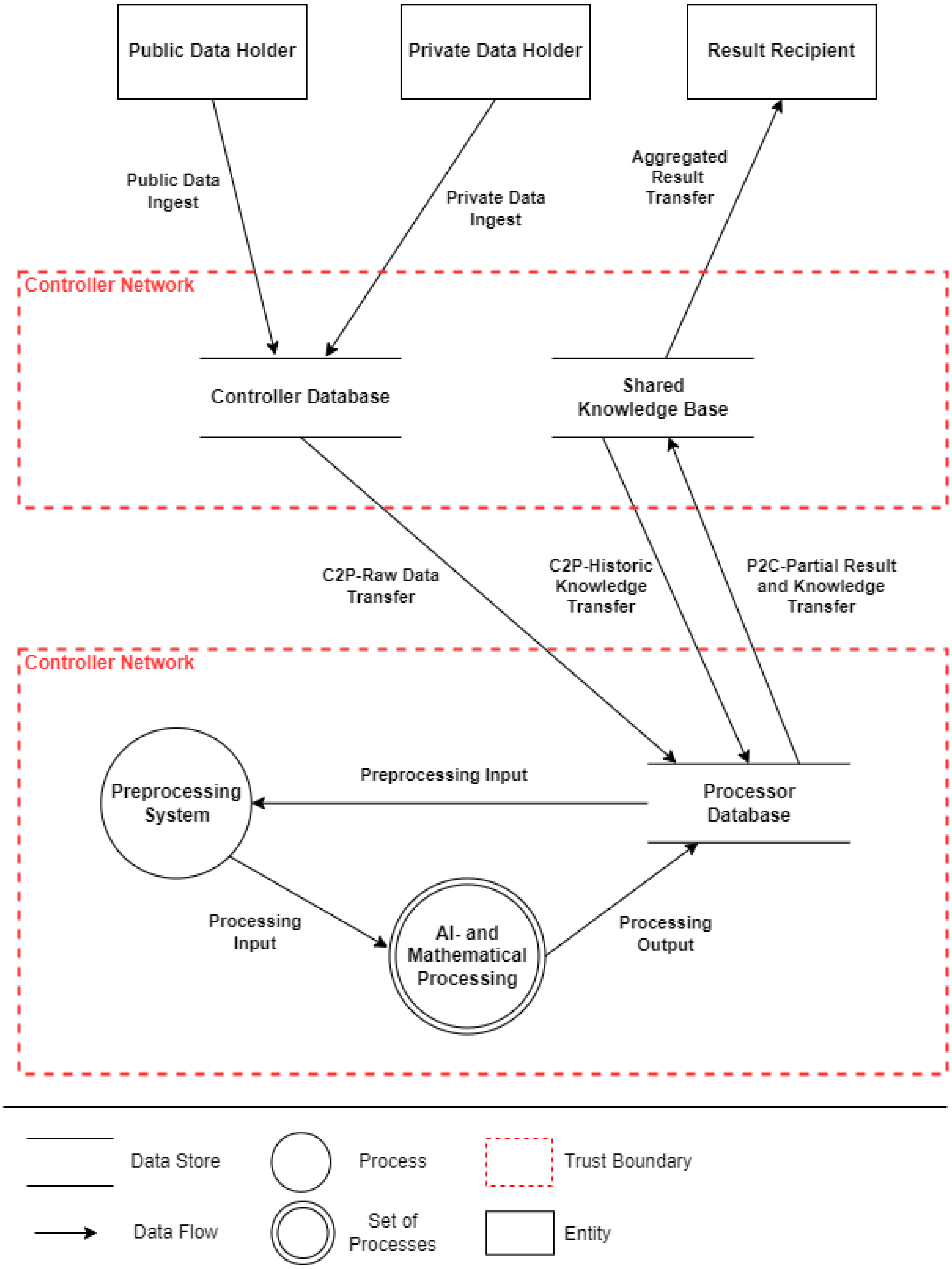


**Figure 2.4:** Data flow diagram (DFD) as defined by [14], showing flows between stakeholders during operation of the HDIS.

# Chapter 3

# Requirements Sources

In this chapter we discuss our chosen sources of data protection requirements and the reasons for their selection. As stated in Section 1.3.1, the most important source for common development projects is the stakeholders involved, particularly the customer who ordered the system or software. Given the fact that Health Data Intelligence Systems are still in a conceptual stage, we don't have fixed stakeholders yet that could be consulted, and thus will have to elicit our requirements from other sources. Considering our three priorities of legal coverage, ubiquity and source stability, national and international legal acts constitute the most reasonable point of entry and will be the first category discussed in this chapter. Secondly, we will consider a number of industrial and governmental standards which focus on system security and data protection, in the hope of providing a standardized point of reference for designing secure systems within different scopes and contexts. These standards usually contain some form of requirement list as well as guidelines on how to implement them.

## 3.1 Legal Sources

Since the enactment of the world's first data protection law by the German federal state of Hessen in 1970, regulations governing the use and distribution of personal data have become ever more wide spread, spurred in particular by the ever growing prevalence of digital technology and the internet [15]. Nowadays, a majority of countries in the world have some form of national data protection legislation, supplemented by international regulations to facilitate trade and transfer of digital goods across state borders. For our purposes, laws and regulations constitute the most important group of requirement sources as any system operated within the borders of a country has to first and foremost abide by applicable data protection regulations to avoid legal consequences that might reach from financial penalties to forced cancellation of operations.

As we assume that Health Data Intelligence Systems are likely to be developed and deployed primarily with initial deployment in Germany in mind, we decided to direct our research on relevant regulations in the same direction [11]. German data protection law is shaped primarily by two specific factors: its membership in the European Union and its governmental system of federal states. This means that we have to consider three levels of legislation with potentially complex interactions and hierarchies. Figure 3.1 shows a basic distribution of responsibilities of the most important laws for different sectors in German data protection legislation and their level of specificity. The GDPR provides the foundation, is relevant for all three sectors and has to be applied in full by all EU member states. It however delegates certain regulatory

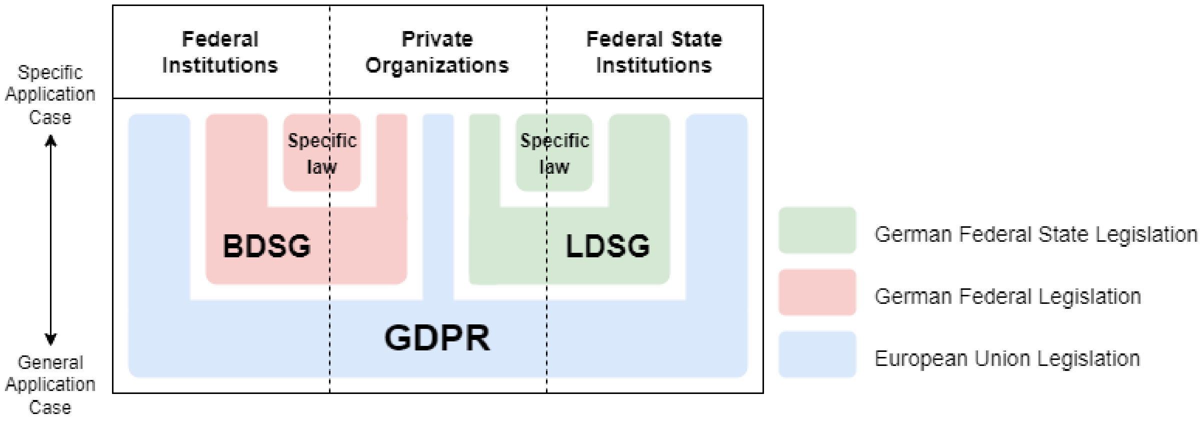

Federal Institutions
Private Organizations
Federal State Institutions
Specific Application Case
General Application Case
Specific law
Specific law
BDSG
LDSG
GDPR
German Federal State Legislation
German Federal Legislation
European Union Legislation

**Figure 3.1:** Relevance of EU and German data protection legislation to different sectors.

responsibilities to the states, with Germany splitting those responsibilities between the federal government and federal states. Both provide data protection legislation for their respective governmental institutions as well as private organizations. On the lowest level, we have laws specific to certain application cases. Unlike the GDPR, the BDSG and LDSGs don't delegate specific areas to other regulations, but are rather intended to serve as a general fallback for all cases for which no specific law exists yet, otherwise referring completely to those laws [2, 16]. We will now take a closer look at these different levels of legislation and discuss their scope and relevance with regards to HDIS.

**EU Legislation.** Regulation (EU) 2016/679, better known as General Data Protection Regulation, was passed in 2016 and became effective on May 25th 2018, replacing the EU Directive 95/46/EC, which had been in force since 1995 [2]. EU legislature in general distinguishes between directives and regulations [17]. The former defines a set of goals that must be achieved, but allows each member state to reach these goals through their own legislative acts, thus allowing for individual solutions for a common goal. Regulations, on the other hand, are legally binding for all EU members and have to be applied in full and unchanged. The upgrade of EU data protection law to the status of a regulation means that it now constitutes the top level of the hierarchy of data protection law in the EU, thus making it our most important source for requirements in order to achieve legal compliance by design [15]. It is made up of 99 articles, covering regulations for and limitations to the collection, distribution, storing, and processing of personal data by public and private entities. It also defines the rights of data subjects, as well as the respective responsibilities of contractual parties and supervising authorities.

Beyond the GDPR, the European Commission is also currently in the process of designing a regulation specifically targeting the use of artificial intelligence for civilian use called the Artificial Intelligence Act[1] [5, 18]. Since Health Data Intelligence Systems will rely heavily on artificial intelligence to fulfill their proposed tasks, this regulation will be highly relevant for the design and operation of any such system when it is enacted at some point in the future and its state of development should be monitored by the research group in the coming years.

[1] https://artificialintelligenceact.eu/

**German Federal Law.** Although it defines a base line that has to be observed by all members, the GDPR does allow for states to extend and specify its provisions with their own laws and even mandates some of its requirements to be implemented in national law as stated in Article 6 (2) [2]. In German federal legislation, this task is mainly assumed by the Bundesdatenschutzgesetz (en. Federal Data Protection Act), which was amended following the passing of the GDPR and enacted on the same day in 2018 [1]. It covers similar subjects as the GDPR, either reinforcing certain points or specifying aspects that had been delegated to the discretion of the member states.

Section 1 (2) BDSG confines the application of the act to only those cases for which no specialized regulation exists. Due to the relative novelty of the BDSG, the number of laws that fall into this category is still rather limited and we weren't able to find any that applied to the specific case of Health Data Intelligence Systems. Even though the Patientendaten-Schutz-Gesetz (PDSG) (en. Patient Data Protection Act) appears relevant on first glance, it is mainly concerned with the handling and exchange of patient data between healthcare service providers, like insurance companies, pharmacies or hospitals and thus not subject to our research in particular [5, 19]. Other regulations are more directed to the interaction with certain institutions that might be relevant as data sources for the HDIS. For instance, the access to pooled data from health insurance providers, which is held by multiple research data centers, is regulated in the fifth book of the Sozialgesetzbuch (SGB) (en. Social Legislation Book) which states various requirements for legitimate use of that data in particular [20]. Since we don't yet know the concrete data sources used by the HDIS, we choose to omit such specific legislation from our requirements sources for the time being. However, this does not eliminate the necessity to revisit and further inspect the landscape of German legislature for similar niche cases in the future, when the circumstances of the HDIS have become more concrete.

**German Federal State Law.** The most granular level of German data protection legislation is enacted by the 16 federal states in the form of the various Landesdatenschutzgesetze (LDSG) (en. Federal State Data Protection Acts). They act as a supplement to the BDSG and GDPR, with their main purpose being to regulate the handling of personal data by federal state authorities and institutions [16]. Therefore, they generally don't apply to private enterprises unless they act as contracted processors for said authorities and institutions as defined by Article 28 GDPR [2, 16]. As the geographical circumstances and legal foundation of any concrete HDIS is only determined in the early phase of its development process and could be situated anywhere within Germany or beyond, it wouldn't make sense to include any particular LDSG into our list of sources for ubiquitous data protection requirements. Instead, they will have to be accounted for on a case-by-case basis by the developers themselves.

## 3.2 Requirement Catalogs

In addition to legal sources, there are various kinds of catalogs published in different contexts that contain requirements for system security and data protection on different levels of technical abstraction and granularity in scope. In the following section we will discuss a number of relevant categories of catalogs and their respective relevance for this project.

**Industrial Standards.** Norms and standards have enjoyed long standing popularity in many globalized industries that rely on some degree of standardization in order to facilitate processes and ensure a certain level of quality. The same holds true for system security and data protection. Industrial standards are mostly developed and published by private organizations which are usually either for-profit or sponsored by certain interest groups. The largest and probably most influential of these organizations is the International Organization for Standardization (ISO), which sets international standards that are recognized and applied by a large number of different industries. With regards to data protection and system security, their main publications are the ISO/IEC-27000-series and ISO/IEC-15408, which is also known as Common Criteria for Information Technology Security Evaluation [21, 22, 23]. Both were developed in cooperation with the International Electrotechnical Commission (IEC).

ISO/IEC-27000 is a set of currently more than 30 individual standards that each cover different aspects of information security. The core theme of the series is an extensive and in-depth guide for establishing, operating and maintaining a so called Information Security Management System (ISMS), which is a holistic approach to ensure data and system security through all levels of an organization's systems and processes [23]. ISO/IEC-15408 on the other hand – better known as the Common Criteria – is an older standard, devised in cooperation with multiple national authorities, including Germany, France and the United States [21]. Its objective is to provide a framework for the evaluation and certification of a system's technical security measures by supervising authorities. For this it defines an extensive set of requirements for multiple aspects of a system, such as the use of cryptography, authentication methods, or pseudonymization of data. Conversely, this can be used by developers as a reference sheet to ensure their systems' and organization's compliance with the standard. Both of them are widely applied by organizations to demonstrate high levels of information security.

**National Standards.** Besides industrial standards, which are mostly published by private organizations, there are also standards which are released by governmental bodies and institutions. With regards to Germany, the majority of these standards is maintained by the Bundesamt für Sicherheit in der Informationstechnik (BSI) (en. Federal Office for Security of Information Technology) and cover a wide range of different subjects with varying degrees of granularity. The BSI 200-1 and 200-2 standards are based upon the ISO/IEC 27000-series and provide a more practical guide to the establishment of an ISMS [24, 25]. They do contain a long list of requirements for this process, however those are more aimed towards organizational measures than system security. Other standards provide guidance towards more specialized aspects of system security like selection of physical server locations or specific requirements for AI-applications [26, 27]. These documents are regularly updated and supplemented by further publications, the most recent being the IT-Grundschutz Kompendium (GSK) (en. Information Technology Basic Protection Compendium), which tries to provide a list of high-level security requirements, covering a wide range of perspectives on information security, including operational processes, IT-systems, and network security [28].

**Scientific Literature.** The field of scientific literature covering data protection requirements, specifically relating to data protection regulation like the GDPR is rather sparse and we only found one article that actually contained a general catalog without being tied to a specific application case with Ringmann 2018, who provides a list of unstructured requirements derived from legal criteria [29]. This lack of structure makes the list somewhat difficult to use in the context of any specific application context, although it may be usable as a point of reference when trying to verify the completeness of our own repository. Apart from that, most authors are more interested in requirements and data protection law from a methodological point of view, which we discuss further in Chapter 4.

Regarding their usability as a source for stable requirements, scientific articles have the distinct disadvantage of inflexibility, meaning they suffer from quick expiration without any guarantee of further updates by the authors in case of a changes in the underlying law or other sources. For other secondary sources like the catalogs published by national institutions, we can rely on some degree of oversight by government authorities, which have a reasonable interest in keeping them up-to-date and in line with professional standards, both of which are not guaranteed for scientific literature. Therefore, while being potentially useful for comparison with our immediate requirements resulting from this thesis, the lack of future safety excludes them from being referenced as sources for actual requirements themselves.

## 3.3 Source Evaluation and Selection

A summary of our evaluation of requirements sources can be found in Table 3.1, where we rated each on a simple scale with regards to their level of conformity to our desired quality standards. Legal Sources have inherent legal relevance and stability, however some are too specific in scope to fulfill our requirement for ubiquity and have therefore not been considered for the requirements engineering process. Considering usability, all legal sources have the advantage of being well-structured and allowing for precise referencing of individual articles and sections which provides traceability for the extracted requirements. On the other hand, they occasionally suffer from ambiguous wording and a lack of precision, meaning any extracted requirements will require future validation by legal experts and/or clarification through case law.

All industrial and national standards analyzed in the previous section generally perform well in terms of stability and ubiquity as they are regularly revised and updated by their publishing institutions and cover a wide spectrum of aspects relevant to realizing data protection on all technical and organizational levels. They are less useful with regards to ensuring legal compliance, as there is no guarantee that adherence to them is considered sufficient by legal authorities in case of a dispute. Lastly, both industrial standards are lacking in terms of usability due to their extensive scope and methodology. However, this is strictly from the perspective of this thesis as they are specifically intended to be used by corporations with the required resources and expertise to accommodate a more in-depth and complex process. While the two BSI-standards are less affected by this, they still share the same issue of limited legal relevance, although this might be solved in the future as we consider it likely that individual standards might be given legal standing through eventual application in case law. In addition to the very limited amount of applicable sources, the lack of both legal relevance and stability lead us to exclude scientific literature from further consideration as sources for requirements.

Taking all these considerations into account, we decided to use the GDPR and BDSG as our primary sources for the requirements extraction process. Both ensure legal compliance and ubiquity through their status as the two central data protection laws, applying directly to any system operated within German jurisdiction. They also provide a high degree of stability as they have a high likelihood of remaining effective for an extended period of time with any changes to their content being incorporated directly into the original release and clearly traceable through the legislative processes of the European Union and German Federal Government respectively. Among national and international standards the GSK stands out, as its solution-oriented approach combined with its implicit status as a publication of the highest German authority on information security, make it a valuable source for the process of finding architectural solutions that are likely to be considered compliant with German law.

| | **Source** | **Legal Relevance** | **Stability** | **Ubiquity** | **Usability** |
|---|---|---|---|---|---|
| **Legal Sources** | GDPR [2] | ++ | ++ | ++ | + |
| | BDSG [1] | ++ | ++ | ++ | + |
| | PDSG [19] | ++ | ++ | - | + |
| | SGB V [20] | ++ | ++ | - | + |
| | LDSGs [16] | ++ | ++ | - | + |
| **Standards** | ISO/IEC-27000-Series [23] | + | ++ | ++ | - |
| | Common Criteria [21] | + | ++ | ++ | - |
| | BSI 200-1 & -2 [24, 25] | + | ++ | ++ | + |
| | BSI GSK [28] | + | ++ | ++ | + |
| **Scientific Literature** | Ringmann 2018 [29] | - | - | ++ | + |

**Table 3.1:** Main requirements sources considered for each category and how well they conform to the desired quality standards.

# Chapter 4

# Requirement Extraction and Compilation

Following the selection of appropriate sources, we move on to our second research question (see Section 1.3.2). For this, we have to consider a suitable method of extracting relevant requirements from each source and organize them in a usable format [30]. This process, called requirements engineering, constitutes an entire scientific and practical discipline in and of itself with a wide variety of perspectives and methodologies depending on the application context. In the following chapter, we briefly introduce some aspects and issues from this field that are related to our application case. Afterwards we describe and discuss our approach to extracting requirements from our main sources, as well as the structure of our Legal Requirements Repository [31].

## 4.1 General Concepts

The general process of requirements engineering comprises four iterative stages: Elicitation, analysis, specification and validation (see Figure 4.1) [32, 33, 34]. The first stage encompasses the tasks of establishing a common vision of the system to be developed and defining its functionalities, characteristics and boundaries as precisely as possible. This process is usually conducted by trained business analysts in cooperation with representatives of all relevant stakeholder groups (i.e. product owners, users, etc.). The goal is to establish a preliminary set of free-form requirements that serves as a basis for the further process. During the analysis-stage these initial requirements are then screened for errors, imprecisions and omissions, with any issues found related back to the stakeholders to clarify and correct. Requirements specification comprises organizing the initial requirements and record them in various, standardized formats that facilitate and harmonize comprehension in order to remove any ambiguities. The last phase is the validation stage, which aims to review the outputs of the specification process. The main tasks are to ensure that the resulting requirements repositories are correct and complete to enable developers to design and implement the final product, as well as to propose tests and metrics against which correctness and completeness of the implementation can be verified.

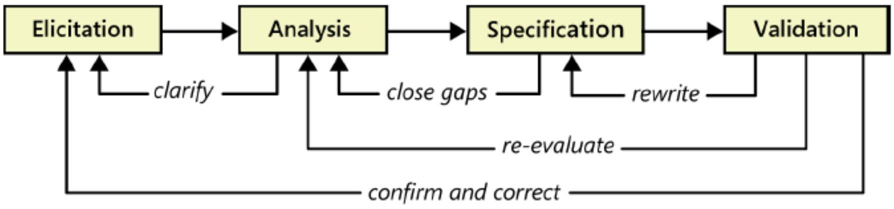


**Figure 4.1:** Iterative process of requirements engineering [34].

### 4.1.1 Defining Data Protection Requirements

The notion of Data Protection Requirements is not well defined in the field of requirements engineering as the term Data Protection itself lacks a universal definition which we discussed in Chapter 1. However, following the two concepts of Mandatory Data Protection and Supplementary Data Protection established in the same Chapter, we can analogously define the concepts of Mandatory Data Protection Requirements and Supplementary Data Protection Requirements. The former category comprises all requirements that are necessary in order to ensure Mandatory Data Protection, while the latter subsumes all requirements that must be fulfilled to comply with any pursued Supplementary Data Protection policy.

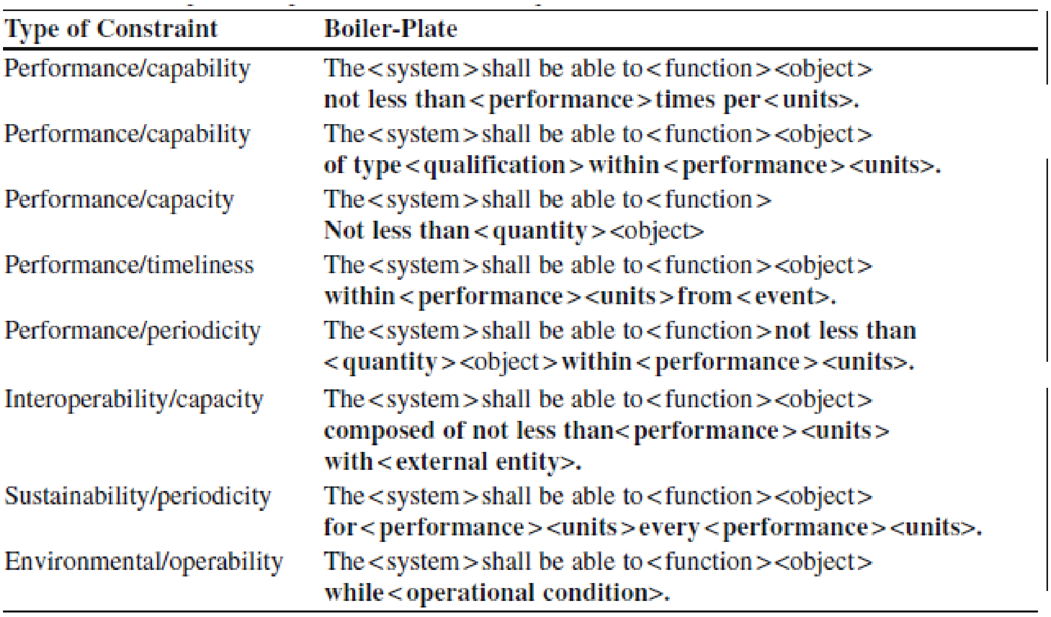
### 4.1.2 Ambiguity and Requirement Templates

Ambiguity in the formulation of system requirements is a prominent issue in the practice of requirements engineering that has been covered extensively by researchers and professional practitioners [35, 36, 37]. In general, ambiguity means the possibility of the reader misinterpreting the original meaning of a sentence that was intended by the writer. Avoiding ambiguity is crucial to requirements engineering, as it can lead to developers gaining a different vision of the planned system than the various stakeholders, at worst resulting in the implementation of a system that doesn't conform with its original vision.

In a literature review on ambiguity in system requirements, Bano identifies five different types of ambiguities that can appear during the requirements engineering process [38]:

- **Lexical Ambiguity**: The ambiguity is caused by a single term having multiple established meanings in a professional or language context.
- **Syntactic Ambiguity**: The ambiguity is caused by the grammatical structure of a sentence allowing for multiple valid interpretations.
- **Semantic Ambiguity**: The ambiguity is caused by the predicate logic of a sentence allowing for multiple valid interpretations.
- **Language Errors**: The ambiguity is caused by grammatical errors that still allow for the sentence to be understood in a valid but potentially erroneous way.
- **Pragmatic Ambiguity**: The ambiguity is caused by differences in contextual knowledge between the writer and the readers.

Lexical ambiguity and pragmatic ambiguity can both be mitigated by providing a glossary along with the requirements repository to clarify any technical terms that might cause this type



| Type of Constraint | Boiler-Plate |
|---|---|
| Performance/capability | The<system>shall be able to<function><object><br>not less than<performance>times per<units>. |
| Performance/capability | The<system>shall be able to<function><object><br>of type<qualification>within<performance><units>. |
| Performance/capacity | The<system>shall be able to<function><br>Not less than<quantity><object> |
| Performance/timeliness | The<system>shall be able to<function><object><br>within<performance><units>from<event>. |
| Performance/periodicity | The<system>shall be able to<function>not less than<br><quantity><object>within<performance><units>. |
| Interoperability/capacity | The<system>shall be able to<function><object><br>composed of not less than<performance><units><br>with<external entity>. |
| Sustainability/periodicity | The<system>shall be able to<function><object><br>for<performance><units>every<performance><units>. |
| Environmental/operability | The<system>shall be able to<function><object><br>while<operational condition>. |

**Figure 4.2:** A set of requirement templates for various types of system constraints [32].

of ambiguity. To address the remaining three types, the most common method is the use of so called requirement templates, sometimes also called patterns or boilerplates [32, 35, 39]. The basic idea is to identify recurring semantic patterns among a set of requirements and using these to define a sentence form for future requirements. The scope of these patterns is arbitrary and can range from extremely specific patterns that are tailored to one particular application context (e.g. heat resistance requirements for different car components) to abstract patterns that can be applied to nearly any scenario (e.g. a general pattern for functional requirements).

Regarding their structure, requirement patterns usually consist of fixed phrases and variables. Fixed phrases encapsulate the common context behind a particular pattern as well as the relation between the various variables. The variables are usually constrained to some degree and only allow for the insertion of a specific category of words or concepts. As an example we consider Figure 4.2, which shows a selection of templates defined by Hull et al. to formalize different types of requirements for system constraints [32]. The first template addresses requirements for the capability of a system to perform certain functionalities without falling below a specific frequency. The bracketed words mark variables that can or must be replaced with a concrete element by the requirements engineer. Let's assume that we want to specify requirements for a printing machine and already elicited the informal requirement that it should be able to produce at least four newspapers per minute. In that case we would take our template and replace each variable with a suitable concept from our informal requirement, resulting in the sentence: *"The <printing machine> shall be able to <print> <a complete newspaper> not less than <four> times per <minute>."*

By providing a detailed explanation of each template and its components, including precise constraints as to what concept or value may be inserted for which variable, the template creator can vastly reduce the likelihood of all of the five types of ambiguity to occur within the requirements repository. It is also useful to assign different priorities to requirements, which is mostly realized through constraints on the template's main operator. The MASTeR-template for example, defines three main operators for requirements, those being *shall*, *should* and *will* [40]. Depending on which of these operators is used at the beginning of a requirement, it is defined to be either mandatory, optional, or intended for an unspecified time in the future.

As stated before, requirement templates can be defined on a wide range of scopes. It is the job of the template creator to identify relevant categories of requirements that suit their particular application case. The more abstract a template is, the more variables have to be entered by the engineer, which reduces the level of conformity among requirements while increasing the risk of lexical or pragmatic ambiguities. A multitude of extremely specific templates on the other hand decreases their usability as engineers have to learn and navigate a larger number of templates and their specific context and constraints, which in turn increases the probability of selecting unsuitable patterns [32, 35].

To improve the quality of our data protection requirements, we also use templates during the specification stage. Identifying a suitable framework for our application in general, as well as selecting the correct template for each individual requirement is a crucial task and we will describe our choice and the rationale behind it further in Section 4.2.

## 4.2 Methodology

In this section, we will present our methodology for the creation of a requirements repository that satisfies the three core principles of legal compliance, ubiquity, and stability. We base our process on the one by Wiegers presented in Figure 4.1, focusing mainly on the first three stages. The validation-stage is excluded from this thesis, as a comprehensive conduction of this task would require cooperation with legal specialists that could properly ensure the correct interpretation of legal texts. Our complete process is shown in Figure 4.3, with the individual stages being described in more detail in the following sections.

### 4.2.1 Elicitation

Considering the lack of stakeholders that could be consulted for business requirements and further details on Health Data Intelligence Systems, our elicitation process has to rely mostly on the legislation, standards and literature presented in Chapter 3. Since Supplementary Data Protection Requirements are not necessary for legal compliance and would violate the principle of ubiquity due to their inherent ties to individual organizations or interest groups, we focus on establishing a comprehensive set of Mandatory Data Protection Requirements. For this we rely mainly on legal sources, as they satisfy all three of our recurring core characteristics of legal relevance, stability, and ubiquity.

As the first part of our process, we traverse the regulations article by article and extract all requirements and restrictions directed towards either the system itself, the data controller, or the data processor. We exclude rights and obligations targeted exclusively towards data subjects, supervising authorities and other external parties as they don’t have any direct relation to the parties involved in the development and operation of the HDIS. The requirements are first documented as free-form texts capturing their main statements, which are further refined during the specification phase.

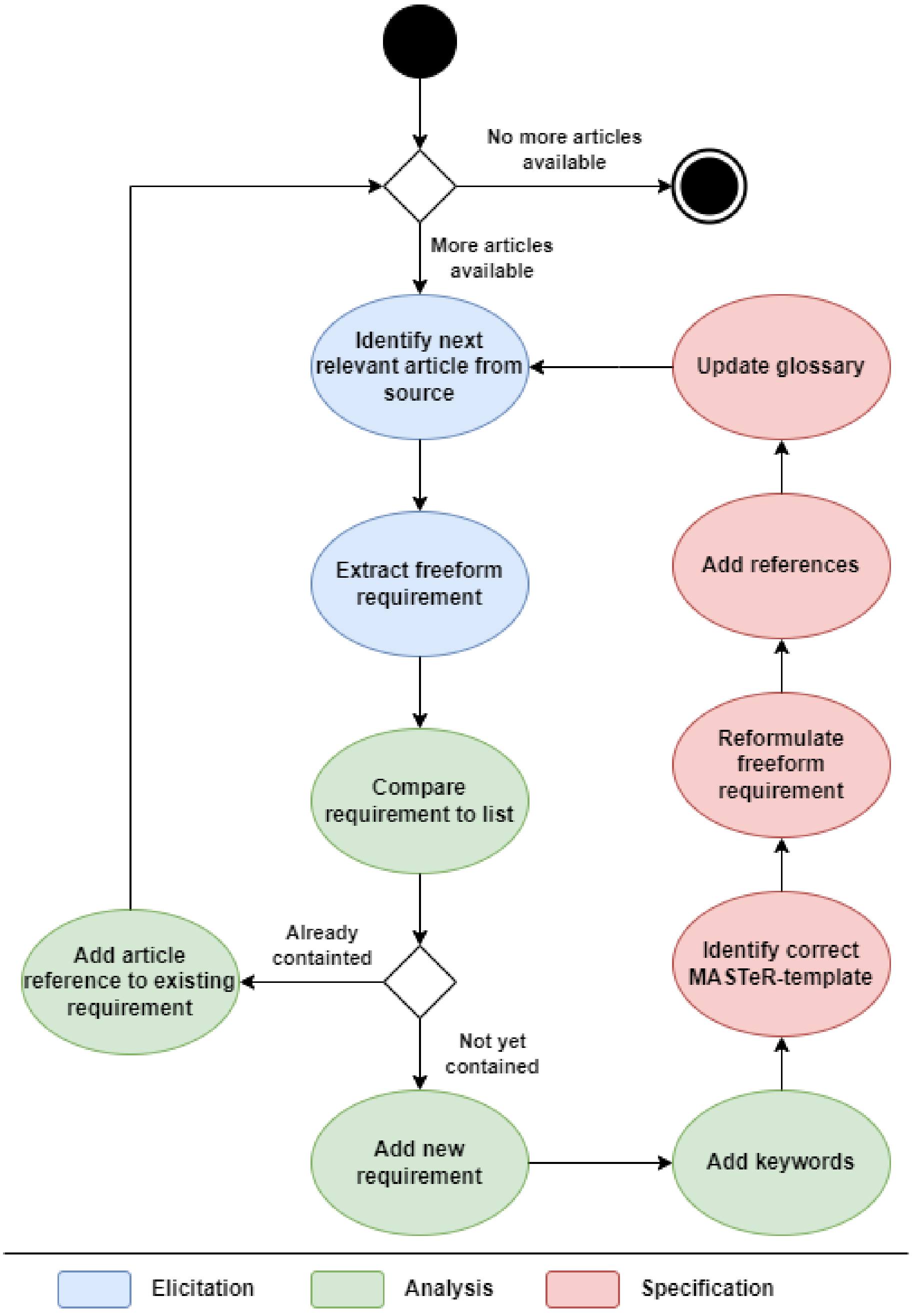

No more articles available
More articles available
Identify next relevant article from source
Update glossary
Extract freeform requirement
Add references
Reformulate freeform requirement
Compare requirement to list
Identify correct MASTeR-template
Already containted
Add article reference to existing requirement
Not yet contained
Add new requirement
Add keywords
Elicitation
Analysis
Specification

**Figure 4.3:** Activity diagram, showing our requirements engineering process and how the individual steps related to the standard phases.

### 4.2.2 Analysis

Once the free-form requirement has been formulated, we compare it to existing requirements in the complete list and look for possible overlaps or analogs. In the case of exact analogs, we simply add the reference to the source article to the existing requirement and continue with the next article. If no exact analog exist, we add a new requirement to the list. In cases where the requirement extends or specifies the statement of an existing one with additional propositions, we add it as a sub-requirement to keep the complexity of each individual requirement as low as possible and facilitate their eventual conversion into development tasks. Otherwise it is simply appended to the end of the list with its own requirement ID.

Finally, we annotate each requirement with one or more keywords summarizing their content into broader categories that are more easily searchable for developers. As a starting point for those categories, we use the 'protection goals' defined by the Standard Data Protection Model (SDM), a publication released by the Conference of Independent Data Protection Supervisory Authorities of the German Federal States which aims to provide a practical guide to the GDPR [41]. For this, it organizes the content of the regulation into seven categories [41]:

- **Data Minimisation**: Limitation of data processing to a necessary and appropriate degree.
- **Availability**: Ensuring that data and processes can be accessed at all times.
- **Integrity**: Continuous conformance of processes to their intended specifications, as well as retention of completeness, correctness, and up-to-dateness of data.
- **Confidentiality**: Prevention of access to data and processes by unauthorized actors.
- **Unlinkability**: Prevention of any merging or linking of data beyond the explicit purposes it was collected for.
- **Transparency**: Ability of data subjects, data processors, and supervisory authorities to access information about collected data and processes to the extent defined by data protection legislation.
- **Intervenability**: Ability of data subjects to enforce their rights to intervening in the processing of data relating to them to the extent defined by data protection legislation.

We start with these categories as keywords and add additional ones if we consider a subset of requirements to warrant a more differentiated categorization.

### 4.2.3 Specification

At last, we finalize the requirements repository in terms of form and presentation. As mentioned at the end of Section 4.1.2, we utilize requirement templates to standardize their structure and mitigate the risk of ambiguities for the user of our repository. We chose the MASTeR framework which is a collection of high-level requirement templates proposed and maintained by the SOPHIST GmbH [40]. The MASTeR-templates have the advantages of being both usable for any thematic context as well as sufficiently abstract to model requirements that are not inherently technical as is the case for many of the data protection requirements extracted from the BDSG and GDPR. Figure 4.4 shows one of the seven templates offered by the MASTeR framework. While it is structured very similarly to the examples presented in Figure 4.2, it also features an optional variable with *condition* which is signaled through square brackets and

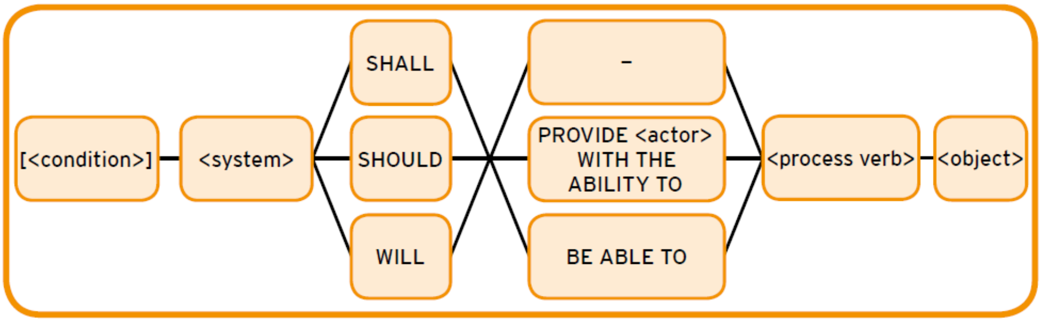


**Figure 4.4:** The Functional MASTeR-template [40].

allows for the variable to be left empty when using the template. We also see that compared to the boilerplates-example, the Functional MASTeR only has a small number of fixed components and instead leaves more room for variables. While this increases the amount of flexibility to accommodate a wider thematic range of requirements, it also increases the necessity to mitigate lexical and pragmatic ambiguity through the use of a consistent and well documented technical vocabulary. The definition of the *shall*-operator as a mark that the requirement is mandatory for the operation of a system is important to represent the inherently mandatory nature of legal requirements.

Returning to our process, to standardize each new requirement, we identify the appropriate template from the MASTeR-catalog and reformulate the free-form description text accordingly. Afterwards we add the reference to the source article to provide traceability for each requirement in case of uncertainty on part of the developers or updates to the sources requiring an update to the requirements as well. The references will also facilitate the validation by legal experts to mitigate the risk of misinterpretations which could lead to either too restrictive requirements or the involuntary omission of important details. Lastly we consider any new technical terms that need to be added to the glossary. This is another important aspect of mitigating ambiguity, especially in cases where the requirement source doesn't provide a comprehensive glossary itself.

## 4.3 Results

In the end, the process as described in Section 4.2 proved feasible for application to both main sources. The majority of requirements were specified using the Functional MASTeR-template presented in Section 4.2.3, as nearly all articles were formulated as obligations for the data controllers and processors instead of requirements to the characteristics of a system. This is primarily due to the GDPR and BDSG being aimed at all sorts of data processing, even if it doesn't involve any information system or automation in general. While we expect the majority of processing in Health Data Intelligence Systems to be fully or at least partially automated, we decided against focusing our requirements solely on technical systems as doing so would omit the perspective of the human personnel involved in their operation, even though they are a crucial aspect of ensuring data protection for the operation as a whole. The finished Legal Requirements Repository can be found in the thesis repository [31].

Overall, we extracted a total of 159 data protection requirements from the BDSG and GDPR. As shown in Figure 4.5, 47 of these were sourced exclusively from the GDPR and 39 from the BDSG, with the remaining 73 originating from both regulations. Figure 4.6 shows the distribution of keywords over all requirements. Over the process of requirements extraction, we added four additional keywords:

- **Distribution**: Restrictions to and requirements for lawful distribution of collected data among third parties.
- **Documentation**: Requirements defining the mandated, continuous documentation of data collection and processing during operation.
- **Legality**: Requirements defining the circumstances under which any purpose for collecting and processing data can be considered legal.
- **Risk Analysis**: Requirements relating to the continuous analysis of risks to personal data posed by external threats or the processing itself.

An interesting discovery is the extremely uneven number of requirements per keyword. *Transparency* stands out with a total of 75 requirements, which is close to half of our total requirements and three times the number annotated with the second most frequent keyword, which is *Documentation*. At the same time, protection goals traditionally valued highly by developers like the CIA-triad only feature with a comparatively small number of requirements.

The main reason for this imbalance is a strong variance in specificity of both regulations with regards to different aspects of data protection. *Availability* for example is only considered in Article 32 of the GDPR, which mandates ongoing availability of data and processing systems without further specifying what the notion of availability entails or how it should be achieved [2]. In contrast, there are eleven different articles partially or completely concerned with *Transparency* with detailed descriptions of the information that should be provided to the various stakeholders.

This lack of specificity for many of the requirements from less represented categories, combined with the absence of any reference to other documents that could provide supplementary information results in a high degree of uncertainty as to the measures that need to be taken in order to fully comply with both regulations. On one hand, this could be interpreted as beneficial to the developers as it leaves them free to apply their own expertise in finding the optimal and most cost-efficient solutions for each requirement without undue restrictions. On the other hand, this could also be seen as a problem as it implies that in case of a legal dispute over any of these aspects, the decision about the adequacy of the developers' measures will be made on the basis of either case-law resulting from previous disputes or subsequent legislation which hadn't been enacted at the time of development yet, meaning they might have to refactor major parts of their system.

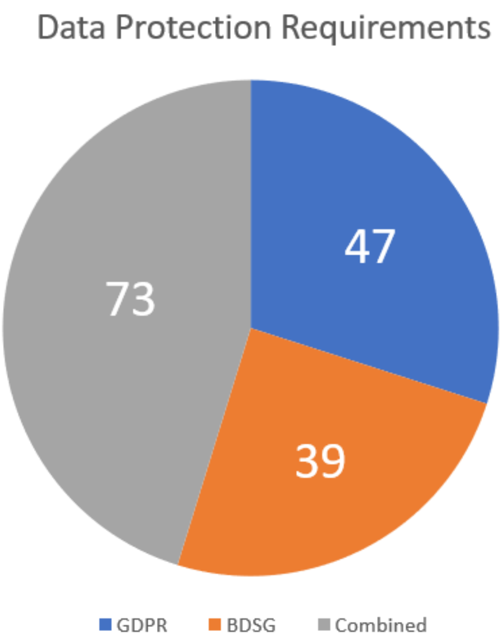


**Figure 4.5:** Distribution of requirements sourced from the BDSG, GDPR or both.

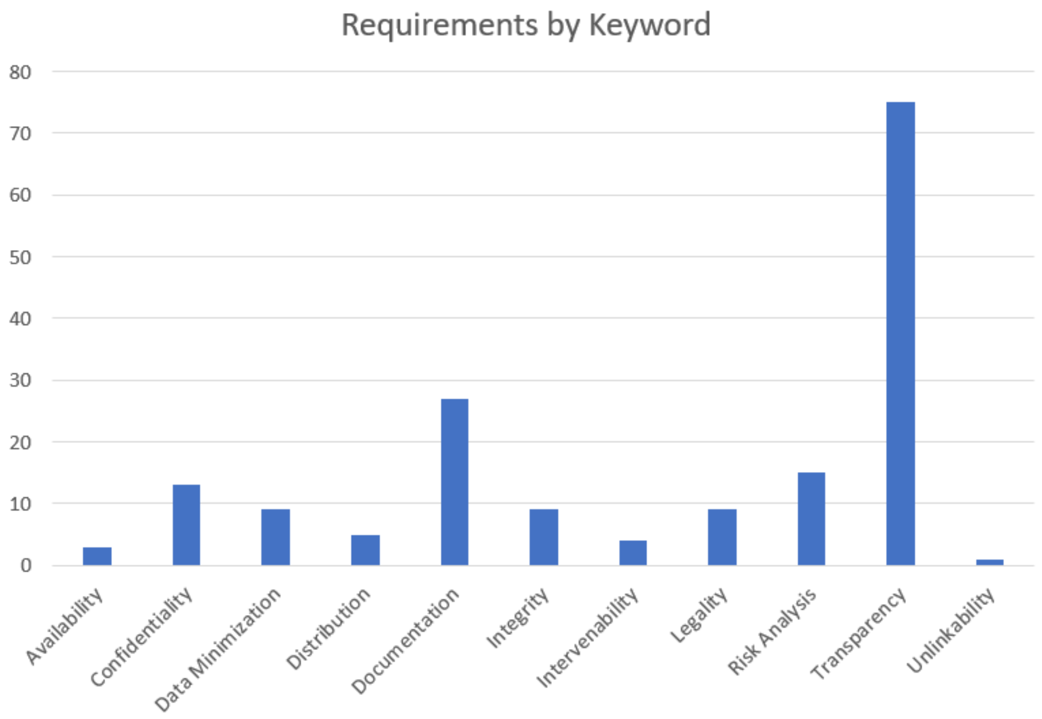


**Figure 4.6:** Distribution of requirements among all keywords.

# Chapter 5

# Threat- and Risk Analysis

As discussed in Section 1.3.3, determining necessary and applicable measures to ensure data protection for any given system first requires the identification of the potential threats and risks that shall be mitigated by their adoption. In this chapter, we address RQ.3 by discussing various kinds of threats that might compromise data protection in an HDIS and presenting a selection of concrete threats that we deem to be the most urgent.

In Section 5.1, we establish the basic terminology used in the practice of threat analysis. This is important as many of the technical terms used in this field, such as *weakness* and *vulnerability*, are often treated as synonyms in common parlance. Afterwards we discuss the sources used for compiling our list of threats and our methodology of extracting them in Section 5.2. Beginning with Section 5.3, we present the actual threats themselves by category.

## 5.1 Terminology

The terms defined in this section were sourced from various large communities of practice as well as scientific articles [28, 42, 43, 44, 45, 46, 47, 48, 49]. All of them are commonly used by the IT-security community for several decades now and are usually defined very similarly, thus facilitating the transfer of concepts between works.

**Weakness.** Describes a flaw within a system or process that might result in the introduction of a vulnerability. These flaws can exist in the system's architecture, design, coding, deployment, hardware or operation procedures.

**Vulnerability.** An existing weakness is considered a vulnerability if it can be exploited by an adversary to cause a system to fail or act in an unintended manner. This usually results in a negative effect on confidentiality, integrity, or availability of an impacted system and its associated data.

**Exploit.** A tool or technique consisting of a sequence of actions, aimed at taking advantage of one or more vulnerabilities.

**Attack.** An intentional action or sequence of actions with the goal of gaining unauthorized access to a system's processes or resources. Attacks are commonly partitioned into five basic stages. During the first stage, the attacker gathers information on the target and its protective measures through passive or active reconnaissance. In stage two, an exploit is used to gain initial access to the target via one or more vulnerabilities. Afterwards, if the attacker intends to maintain access for a longer period of time, they introduce a backdoor into the system that allows for easier access in subsequent intrusions. Stage four is comprised of lateral movement to other systems that might be connected to the initial point of entry through an internal network. Lastly, during the action phase, the attacker executes the actions necessary for his original goal which can vary greatly but will invariably cause a violation of confidentiality, integrity, and/or availability.

**Threat.** Describes any set of circumstances or events that might lead to a negative impact on a system. Threats are potential by default but can become realized threats by being used to actively impact a system, usually through the direct exploit of a vulnerability. Threats can be force majeure (e.g. earthquakes, fire), human error (e.g. phishing, lost items, deleted files), technical failures (e.g. hardware malfunctions, lack of storage or processing power) or intentional misuse (e.g. hacking, theft).

**Risk.** A value which is calculated through the combination of various metrics and functions as a priority ranking to decide if and in which orders additional measures need to be taken to address each specific vulnerability. Common metrics are the value of an affected asset, the likelihood and impact of a potential threat as well as the severity (i.e. ease of exploitation) and exposure (i.e. number of system components affected) of a vulnerability. A risk is usually assigned to either a threat or a vulnerability depending on the perspective taken by the analyst. This way, designers and developers can decide to either focus on single threats by eliminating their corresponding vulnerabilities or take a broad approach by prioritizing specific vulnerabilities that are part of many threats to mitigate their risk across the board.

## 5.2 Methodology

Conducting a dedicated analysis of potential threats and risks is common practice among system developers, and many concepts attempting to standardize this process have been proposed. On an international level, the ISO/IEC 27005 is probably the most important example [50]. It is part of the ISO/IEC 27000-series which attempts to cover the entire information security lifecycle with a holistic system called an *Information Security Management System (ISMS)* [23]. On a national level, many countries have published individual documents describing standardized approaches to threat and risk assessment, some notable examples being the *Harmonized Threat and Risk Assessment (TRA) Methodology* by the Canadian Communications Security Establishment [44] and the Guide for Conducting Risk Assessments by the US National Institute of Standards and Technology [45].

There also exist more specialized guides focused on specific aspects of threat analysis like the *Standard Data Protection Model* (SDM) which was developed by the Data Protection Conference of the German federal states and focuses on the required degree of protection for data used in an organization's operations with a focus on legal compliance [41]. Its counterpart is the IT-Grundschutz-Kompendium (GSK), which focuses on the more technical aspects of system security and is meant to be used complementary to the SDM [28]. It is published and maintained by the Federal Office for Information Security.

While most of these approaches are comprehensive and fairly similar with regards to their basic structure, they are very extensive due to being designed with large, corporate-scale organizations in mind, which are able to dedicate anything from several employees to entire departments to this process, including domain experts on the concerned system and professional system analysts. Furthermore, it is usually assumed that the system in question is either in an advanced stage of design or already in deployment, allowing for the use of automated vulnerability analysis tools like OpenVAS [51] or Nessus [52].

For Health Data Intelligence Systems, we have neither of these common prerequisites and thus need to adapt our process accordingly. In the following sections, we present our custom methodology, which is based on concepts of common threat and risk analysis that can be feasibly and meaningfully applied to Health Data Intelligence Systems, i.e. those based on the initial specifications outlined in Chapter 2.

### 5.2.1 Setting the Scope

Our first step is to set the scope for our threat and risk assessment, meaning we have to decide what kinds of threats we want to take into account.

Like most aspects of system development, threats can be defined on various levels of abstraction. As an example, the CAPEC-knowledge base presents a list of adversary attack techniques which differentiates between three different levels of abstraction [47]. On the highest level we have *Meta Attack Patterns*, which provide a general characterization of common attributes expressed by a larger group of attack strategies (e.g. Parameter Injection). On this level, patterns are still mostly independent of any specific technology. *Standard Attack Patterns* describe a specific technique on a more detailed level (e.g. Abuse of Command Delimiters). Such techniques are usually aimed towards a certain group of technologies that share common weaknesses that can be exploited in a similar way. Lastly, CAPEC considers *Detailed Attack Patterns* which are aimed at one specific technology using one specific technique (e.g. HTTP Parameter Pollution). The same applies when switching perspectives and analyzing potential weaknesses that might be abused to compromise the system. Comprehensive knowledge bases like the CWE employ similar classification hierarchies with entries being categorized from highest to lowest abstraction level as *Pillar Weakness*, *Class Weakness*, *Base Weakness* or *Variant Weakness* [49].

Since this thesis focuses on the conceptual planning phase of HDIS, we forego low-level threats in this analysis. Not only are they much too extensive in number to provide any form of utilitarian value for a development team if presented in full, but are also mostly tied to the utilization of very specific technologies within the system, which might not be included at all in the final implementation. Therefore, we mostly consider high-level threats and weaknesses as they will hopefully provide more utility for future development teams by offering a set of general threats and weaknesses which they can then build upon depending on the concrete technologies employed.

To achieve the best possible protection efficiency for our reference architecture presented in Chapter 6, we also want to optimize the ratio between number of mitigation measures and the combined level of protection they provide. For this reason, we prioritize threats based on their likelihood and impact. The former means the likelihood of a specific attack technique being used against the HDIS or a particular weakness in the system to be exploited. Severity, on the other hand, describes the impact that a successful attack or exploit could have on system protection.

Last but not least, we want to provide the broadest coverage of the potential threat surface while considering only threats that we believe to be relevant for all future implementations of HDIS, regardless of their individual choices of technologies. To account for all these factors and requirements we decided to focus our research on two major aspects that satisfy these requirements.

**Artificial Intelligence Threats.** AI is a broad field of many different techniques and approaches. However, at a basic level many common AI-methods share functional characteristics that can be vulnerable to manipulation and disruption from external adversaries [53, 54, 55]. Apart from that there are also concerns with regards to possible detrimental effects that the use of Artificial Intelligence could introduce to applications handling personal data [56, 57].

**Threats to General System Security.** The consideration of individual system characteristics is an important factor due to the potential of specific threats that might be overlooked otherwise. However, overall data protection cannot be ensured if the system as a whole is not protected against malicious influence as most other security measures can be evaded or penetrated by compromising the system itself. Since general system security is an extensive topic that is also highly dependent on the choice of technologies by the development team, we focus on threats with particularly high relevance due to being extremely dangerous in terms of impact and very common over many types of technologies.

### 5.2.2 Identifying Assets

The second preparatory step comprises the identification of crucial assets that might be affected by different threats. This also includes valuating each one with regards to the potential negative impact that could be caused by it being damaged or otherwise compromised. The term *asset* is very broad and can mean anything and anyone that might be subject to threats or exhibit weaknesses that could endanger the overall operation of a specific system or an organization in general. Since this thesis is concerned with the design phase of HDIS, specifying precise information about the assets is challenging. Therefore, we resort to the limited information from our system outline and define broader categories of assets.

**Processing Systems.** As defined by Article 4(2) of the GDPR, any system coming into contact with private data – be it for storage, computation or transmission – is considered a processing system from the perspective of data protection legislation. Since this classification is too broad for our purposes, we treat data storage and data flows as separate asset types and constrain our notion of processing systems to a number of components performing specific tasks. From the system outline established in Chapter 2 we derive three principal types of processing components. Data management systems comprise all systems responsible for data ingest, organization and distribution among data holders, controller, processors and recipients. Preprocessing systems are systems responsible for preparing raw data for further use. Since we assume a decentralized organization of data processors, there will be a multitude of different preprocessing systems as each AI- and mathematical simulation method requires differently formatted inputs. Lastly we consider AI- and Mathematical Processing Systems which produce the various outputs that will be either used for future runs or aggregated to form the results returned to the data recipients.

**Data Stores.** Data stores are a among the most important assets for AI-driven systems as they contain not only the raw data used as an input for the system but also all outputs, including models that will be updated and reused for further processing runs. They are also one of the most common targets for attackers that either want to profit or damage an organization, as data often holds significant value and is easier to steal or compromise than physical assets. Any disruption of data stores not only affects an organization's internal operations but can also cause legal consequences and damage to public perception. All those aspects are crucial to an HDIS as its operation will involve high volumes of personal data and is entirely dependent on a governmental mandate, which in turn is strongly connected to continued public trust in the system's security.

**Data Flows.** Data-intensive, distributed systems necessarily involve a large number of data flows between data holders, controllers, and processors (see Figure 2.4). These flows are subject to a number of potential threats depending on their implementation and the data they are carrying. Specific attention has to be given to any data flow carrying private data as they are likely to be subjected to more threats than their counterparts only used for transferring public data.

**Human Staff.** While not directly included in Figure 2.4, human interaction will be a big part of HDIS as data-collection, system operation and administration are unlikely to be fully automated. On one hand, humans are one of the most common targets for adversarial attacks against organizations and their IT-infrastructure, due to the comparative simplicity of the techniques used against them [58]. On the other hand, humans also pose a threat in and of themselves as we are inherently vulnerable to negligence and other non-malicious threats that can nonetheless gravely endanger privacy in other assets. The risk of such threats depends on the specific system in question as well as the number of staff involved in the operation of critical systems.

**Data.** Data in general and private data in particular has to be considered as a security-relevant asset, as getting some degree of access to it is among the main goals of adversarial attacks. In addition, there are comprehensive legal restrictions, whose violation could lead to a breach of its privacy without any outside interference simply due to faulty internal processes. While concern for public data as defined in Section 2.4 is mostly aimed towards its integrity, any private and personal data has to also be considered from a confidentiality perspective, as its contents may not be disclosed to any unauthorized party.

**Physical Systems.** When considering data storage and processing systems, we also have to consider possible threats to the hardware necessary to operate them. These can vary widely and range from natural disasters to fires or simple theft and are usually less specific to the context of the processing itself.

## 5.3 Artificial Intelligence Threats

In this section we will discuss threats that are specific to systems and applications that utilize artificial intelligence and which of them might be applicable to Health Data Intelligence Systems in general. We start with a discussion of possible sources for researching this particular category of threats in Section 5.3.1. We then proceed with a general overview of the common processing pipeline as well as a description of our methodology to extract a subset of the most relevant AI-related threats. Lastly, we will present our findings and consider potential mitigation strategies.

### 5.3.1 Sources

Even though artificial intelligence is used extensively by large scale corporations and in commercial applications for many years now, the discourse about its security during training and deployment has been largely confined to the scientific sphere until recently [59, 60]. Unlike general system security, which has been in the scope of the industrial sector and governmental authorities for a long time, their consideration of AI is relatively recent and the amount of regulations, standards, and other publications regarding its security is still sparse. While the BSI for example has released several extensive publications with the goal of sensitizing organizations for cyber threats and security, their releases towards threats to and concerns regarding artificial intelligence are limited to two white papers in 2021 [61, 62]. However, awareness is steadily increasing, which is also indicated by the growing number of ongoing initiatives to introduce regulations and standards to this important field of computing, like the EU Artificial Intelligence Act and Ethics Guidelines for Trustworthy AI or the publications by the ISO/IEC-committee on Artificial Intelligence [18, 63, 64].

This means that the majority of available sources, especially regarding external threats to systems using AI, has been contributed by the scientific community, whose number of publications in this area has grown exponentially since the mid 2010s [60]. As a starting point, we consulted both BSI publications regarding security, robustness, and transparency, as well as multiple scientific surveys, which provide decent overviews of the topic of secure AI [54, 55, 59, 61, 62, 65]. It's important to note that the terms artificial intelligence and machine learning are often used synonymously by many sources, both scientific and non-scientific, which is technically incorrect as the later is usually considered a subset of the former. This imprecision can be attributed to the overwhelming prevalence of machine learning over other techniques in current commercial applications of artificial intelligence and its strong representation in the media. This is also reflected in the scientific sources for this research, which almost exclusively focus on machine learning techniques in their discussion of threats and countermeasures.

When discussing more specific threats, the common pattern followed by researchers generally is to propose a novel way of attacking existing algorithms as well as possible approaches to counter these attacks, meaning that the arms race between attacks and security mechanisms is driven on both sides by the scientific community itself. Zhao et al. for example first introduce a technique for attacking learning models under black-box conditions and then consider possible defenses against their approach [66]. Inherent problems of artificial intelligence have also been discussed by researchers, specifically focusing on ethical concerns with its application and transparency [56, 57, 67].

While scientific papers constitute a good source for technical information about threats, they provide little to no quantitative data about the extent to which those threats occur in the wild [60, 68]. Further research outside of scientific literature didn't yield any clear statistical data

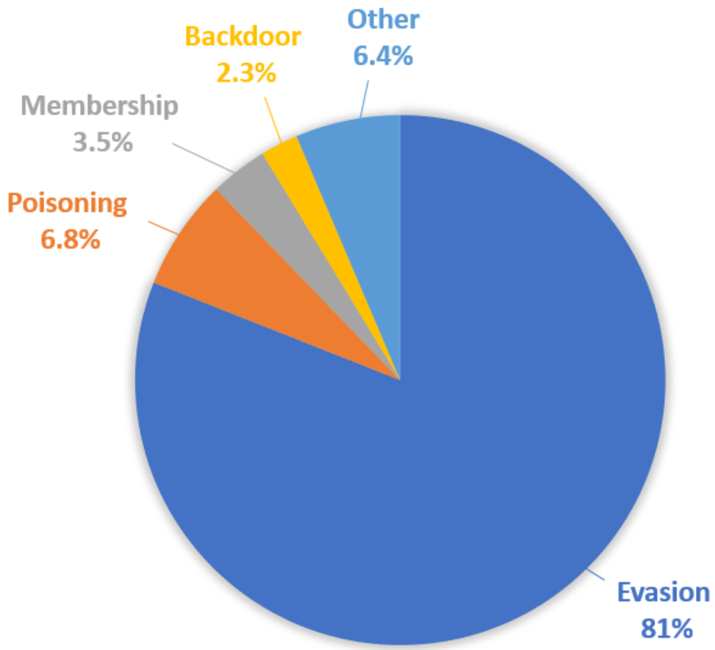


**Figure 5.1:** Average distribution of different categories of attacks on artificial intelligence as the principal topic of scientific papers between 2010-2020 [60].

either, implying a lack of publication of data on the frequency and types of attacks on their systems by targeted organizations. A study by AdversaAI in 2020 examined the representation of different types of attacks on artificial intelligence in scientifc papers after 2010 [60]. It discovered a strong focus of scientific literature towards so called evasion attacks (see Figure 5.1), but this can be attributed to their practical relevance to AI-driven image classification, which is currently the most important field of commercial application [60]. McGregor introduced the *AI Incident Database* in 2020, which contains information on 1400+ incidents, involving ethical issues, security incidents and privacy breaks and is maintained and extended by an open community[1] [69]. However, each incident entry is usually documented in the form of quoted news articles or similar, unstructured sources, only uses limited keywording, and doesn't provide much statistical information.

### 5.3.2 General Overview

Due to the lack of sources covering security issues in other variants of artificial intelligence techniques, our overview focuses mostly on machine learning as it is the most widespread form used in practical application. Artificial intelligence threats can be divided into two groups, these being external and inherent threats. The former comprises all forms of threats posed by external adversaries that actively intend to cause damage to the system or impede its performance, while the latter encompasses all issues that are inherent to the application of artificial intelligence and could negatively influence its operation. In this section, we provide some contextual and methodological background before moving on to discuss both groups successively and finish with an estimation of their threat potential with regards to Health Data Intelligence Systems.

[1] https://incidentdatabase.ai/

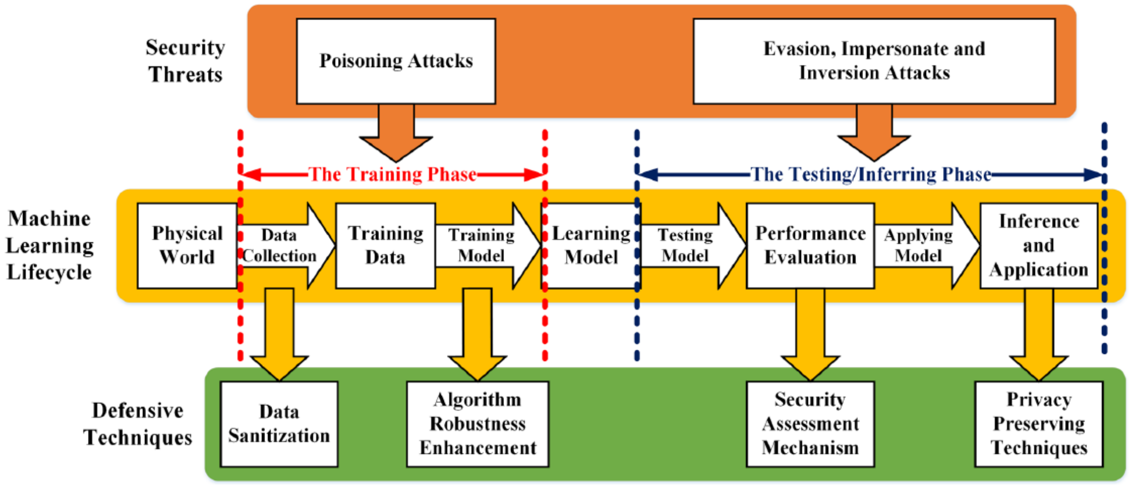

Security Threats
Poisoning Attacks
Evasion, Impersonate and Inversion Attacks
The Training Phase
The Testing/Inferring Phase
Machine Learning Lifecycle
Physical World
Data Collection
Training Data
Training Model
Learning Model
Testing Model
Performance Evaluation
Applying Model
Inference and Application
Defensive Techniques
Data Sanitization
Algorithm Robustness Enhancement
Security Assessment Mechanism
Privacy Preserving Techniques

**Figure 5.2:** Common security threats and defensive techniques and where they relate to the common machine learning life cycle [55].

**Machine Learning Pipeline.** The base life cycle of a machine learning system consists of two phases: The *training phase* and the *inference phase* [55]. Figure 5.2 provides a concise overview of these two phases as well as potential security threats and defensive techniques that can be applied at different points of the pipeline. During the training phase, the system collects data from the physical domain (e.g. images, audio, textual data) to which the system is applied. This training data is then converted into input data which is used to train and optimize the actual learning model. After the training the process, the model can then be tested for its performance within the actual application context and lastly applied to its intended use case (e.g. mobile music recognition-app, automated driving, etc.). Threats are often categorized according to the phase within the machine learning life cycle during which they impact the system.

**Adversarial Model.** An important factor in the consideration of the applicability of external threats to an AI system is the adversary controlling the threat. For this, researchers commonly employ adversarial models consisting of various characteristics that influence the types of attacks that can be conducted as well as their severity. These characteristics are made up of three aspects, which we extracted from scientific literature and visualized in Figure 5.3. In the following, we describe these three aspects and their respective attributes in greater detail.

First, we consider the attacker's goals. Among them, the intended impact is the most straightforward and is mostly differentiated into privacy/confidentiality attacks, integrity attacks, and availability attacks [55, 65, 68]. The first category comprises all forms of information disclosure, with common targets being the model parameters (i.e. model parameters that are optimized during the learning process) and hyper-parameters (i.e model parameters that are set beforehand and remain fixed during the learning process), as well as the data used during the training process. Attacks that decrease the performance of the model in its assigned task by causing it to produce erroneous outputs during the inference phase are called integrity attacks. Due to the common scenario of machine learning being used to categorize input data and assign labels to it, those outputs are also generally called misclassifications [65]. Lastly we have attacks on availability which means the general ability of the system to perform its designated task when required. This is mostly relevant for live services where the model is made available to an arbitrary number of users on demand. The implications of such an attack are similar to other contexts, meaning that an attack on availability would prevent users from accessing the AI-component of the service.

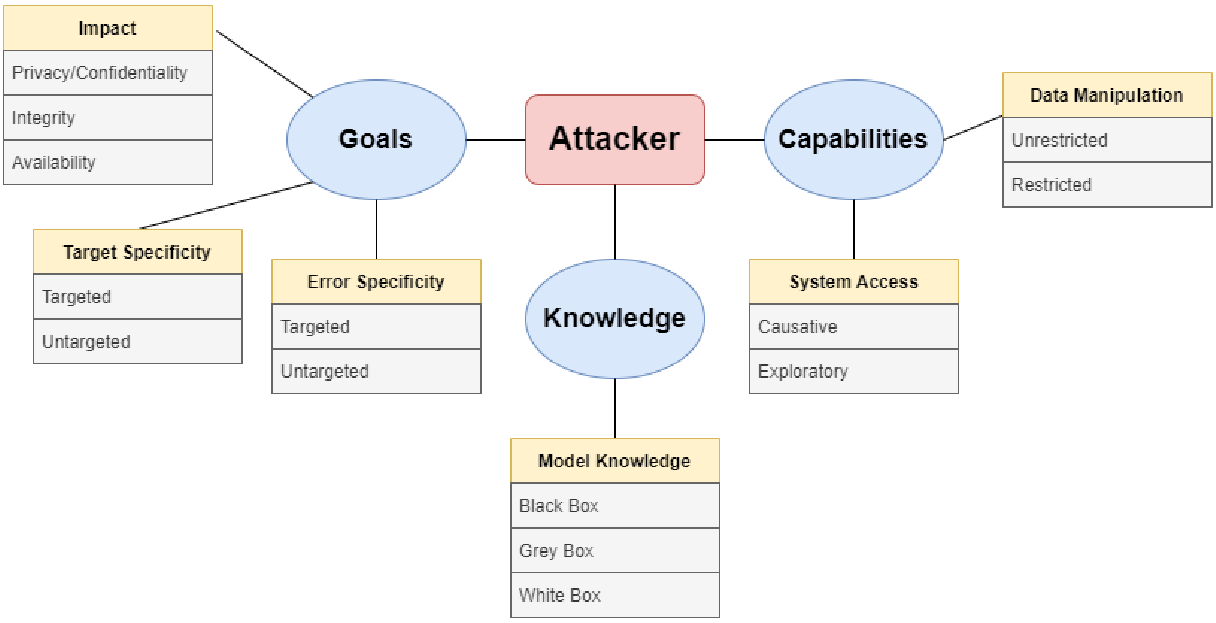


**Figure 5.3:** Common attributes of the adversarial model for machine learning extracted from scientific literature [54, 55, 59].

Apart from the impact, we also consider two notions of specificity regarding the attacker's goals [59]. First, we have the target specificity, which describes the range of inputs that the attacker intends to manipulate. If their attack is focused on a specific subset of input data, we speak of a targeted attack, while the manipulation of any input-set of sufficient size is called an untargeted attack. This concept can be transferred seamlessly to the error that shall be produced: In one case the attacker wants to produce a misclassification of a specific output, while in an untargeted scenario the goal is to cause any output other than the ground truth. Error specificity in general is mostly important for attacks aimed at the model integrity and the concept therefore is not equally relevant for all types of attacks.

The second factor of note is the attacker's knowledge about the targeted system, particularly concerning information about the learning model, the training data used and the parameters and hyperparameters. Due to the large variety of possible scenarios with varying degrees of knowledge to consider, AI-researchers usually distill these down to three general scenarios [54, 55, 70]. In a black box-scenario the attacker is assumed to have no insight into the internal workings of the model, with their only potential source of information, depending on the application scenario of the AI, being the outputs of the API that is presented to them during model deployment. Opposite to that, an attacker in a white box-setting has perfect information about the model either due to being part of the development team, having gained previous access to the development environment or having been able to reverse engineer the model outside of the official system. Lastly, we have the notion of a grey-box setting, which covers any degree of information between the two previous scenarios and is only rarely considered by researchers [59]. This is mainly due to new discoveries in the area of model-reverse engineering in recent years which have made clear that it is possible for attackers in many common use cases to create accurate substitute models from extremely limited information by using large numbers of input/output-pairs to approximate the original training data [71, 72].

The third and final factor in the adversarial model are the attacker's capabilities, meaning the ways in which they can access and manipulate the model and the resources it uses. Here researchers commonly discern between the attacker's access to the system itself and their ability to manipulate the underlying data [55, 59, 66, 68]. For the former view, the two most common scenarios are causative and exploratory access. Causative access is given if the attacker is able to influence the model's parameters or hyperparameters directly or indirectly. Direct manipulation requires administrative access to the operator's system itself, while indirect manipulation can be caused by supplying the system with large amounts of malicious training data which is often possible through the misuse of official APIs. Considering the vastly greater access required for direct manipulation, gaining indirect access to the model is much more common in practice [70]. On the other hand, a scenario in which the attacker cannot influence the model's performance, but only its input/output behavior, is called exploratory. In this case, the attacker cannot introduce weaknesses into the model but is instead forced to seek existing weaknesses in the current model. This makes effecting their intended impact more difficult, but at the same time requires less expertise and access to the backend.

The second aspect of an attacker's capabilities is their ability to manipulate data. Most attacks, be they causative or exploratory, require the introduction of very specific manipulations into either the training or test data. However, the degree to which that can be done might be limited by other factors. Images for example are commonly represented as a feature vectors consisting of individual pixel values, which allows the insertion of much more precise and subtle alterations as opposed to categorical data, where each feature has to conform to a very limited set of values, severely limiting the range of permutations [70].

### 5.3.3 Training Attacks

After establishing the different characteristics that attacks on AI can exhibit in general, we now discuss individual types of attacks and their relevance to HDIS. A common goal for attackers of AI systems is to manipulate the function that is learned by the model in a way that is either specifically beneficial to them or detrimental to the targeted company. This in turn means that the attacker has to find a way to interfere with the model's training process. The most prevalent method to achieve this is through so called *Poisoning Attacks* which we discuss further in this section.

**Poisoning Attacks.** The general idea of such an attack is illustrated in Figure 5.4. Under ideal circumstances, the model developers use training data which perfectly represents the structure of the physical domain which the AI is supposed to learn [55, 70, 73]. From a mathematical perspective, the data contains a certain distribution which ideally closely approximates an intended real world domain. The learning algorithm's task then is to teach that distribution to the model. This means that the easiest way for an attacker to influence the distribution learned by the model is to manipulate the training data from which it is learned. This manipulation is called *poisoning*, the intention being to substitute a legitimate model with a corrupted version for the testing and inference phases.

Applying the adversarial model, the intended impact of a poisoning attack is always the integrity of the targeted model, as it aims to either reduce the general performance of the model (e.g. reduce classification accuracy or confidence) or skew it towards specific types of misclassifications that can be abused during deployment. This further implies that target and error specificity of poisoning attacks can be both targeted or untargeted depending on the intended outcome. The required knowledge for such an attack also depends on this, as a mere reduction of model performance could theoretically be affected through the simple injection of random noise into the training data. A specific misclassification, on the other hand, requires far more insight into the training data to find the correct permutations to make. Regarding the necessary

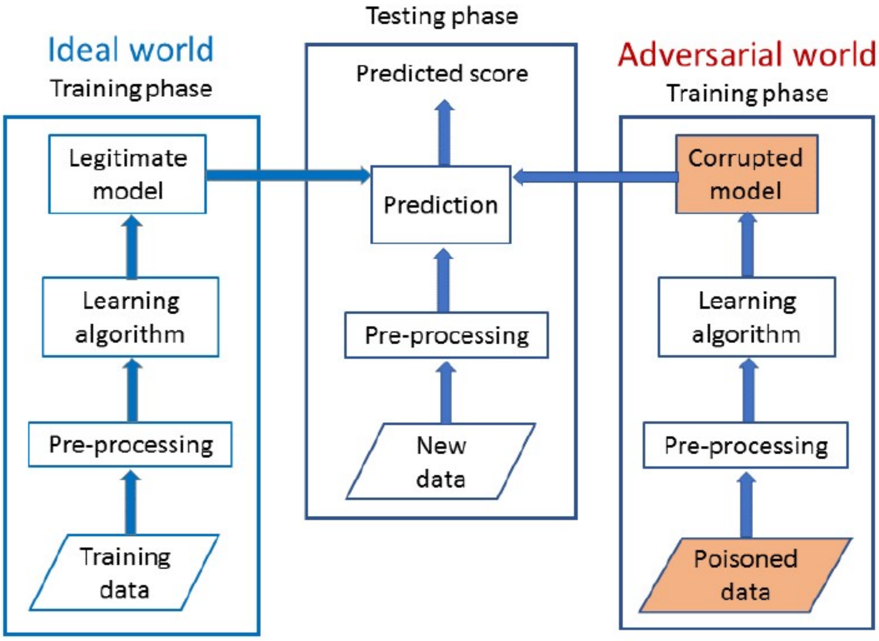


**Figure 5.4:** Comparison of a learning algorithm's training phase under ideal circumstances and under the influence of data poisoning [73].

capabilities to conduct a poisoning attack, the manipulation of training data is always considered causative access. This access however can have different forms, depending on the system in question. Many modern AI-applications rely on crowdsourcing and online learning to continuously improve the performance of their models utilizing the data provided by the users of their application. In such a case it would be relatively easy for attackers to use mass automated queries to the application to affect the learning algorithm on the go. For comparison, in an offline-learning scenario, where the model is only trained in fixed intervals and independent from the live data feed, affecting the training data becomes much more difficult as an adversary either needs access to the data storage or intercept the data before or during preprocessing.

### 5.3.4 Inference Attacks

In cases where the adversary has no ability or intent to manipulate the model itself, but rather abuse vulnerabilities of the model during deployment, we speak of an *Inference Attack* [55, 65, 71]. Since this type of attack usually requires much less prerequisite information and less specialized access to the training environment, the variety of attacks is lager but can be generally boiled down into two major categories which we discuss in this section.

**Inversion Attacks.** Inversion with regards to artificial intelligence refers to the extraction of sensitive information from models through analysis of input/output pairs [55, 65, 71, 74]. These kinds of attacks are typically aimed at apprehending either singular data points from the training data (see Figure 5.5), the entire set of training data, or the model parameters itself. Beyond the obvious incentive of potentially discovering private and valuable information, they are also commonly used as a preparatory step for other attacks as the extraction of training data and analysis of input/output-behavior allows an adversary to create a substitute model that simulates the original model and can be used to improve the precision of subsequent attacks [72].

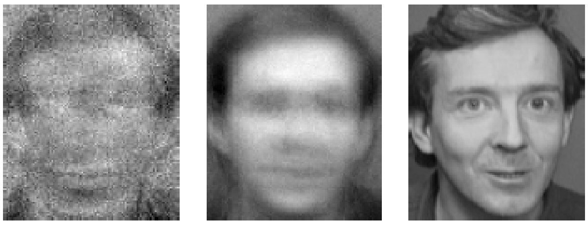


**Figure 5.5:** Two results (left, middle) of reconstructing-attempts of an original image (right) from the training data of a facial recognition model through different model-inversion techniques [71].

Considering the goals in terms of our adversarial model, inversion attacks always have an impact on privacy and/or confidentiality as the trained model and often also the training data are proprietary, with the latter also potentially containing personal or other private information. The target specificity of inversion attacks can be both targeted and untargeted depending on whether the adversary wants to extract information about one specific point or subset of data or the entire training data. Even though previous knowledge about the model can be beneficial to decrease the number of queries required to achieve the intended impact, it is not necessary and inversion attacks can therefore be conducted in a black box environment. The main limiting factor of not only inversion attacks but most inference attacks are the necessary capabilities. Exploratory access is generally much easier to achieve than causative, as many applications using artificial intelligence require some degree of model exposure to the user. On the other hand, this also means that the format in which an adversary can freely interact with the input and output of the model are determined and potentially restricted by the developers. Unrestricted access to the model therefore requires further efforts to circumvent the public API.

**Evasion Attacks.** The act of intentionally generating or manipulating input data so that it is misclassified by a learning model is called *evasion* and is one of the earliest forms of attacks on artificial intelligence, first being used to allow the circumvention of modern email filters by spam mails [55, 65, 59, 68]. The range of use cases for evasion attacks is broad and particularly relevant to common modern applications of machine learning like image classification.

Evasion attacks generally jeopardize the integrity of affected models. Their targets for this are usually specific with error specificity being targeted or untargeted depending on the particular goal. For example, an attempt to make a facial recognition system not identify someone as a person would have an untargeted error specificity as the goal is for that person to be classified as anything else. In the opposite scenario, trying to have a person classified as another particular person would require to cause a targeted error. Evasion attacks can require different levels of knowledge depending on the required level of precision and obfuscation. While a misclassification can be caused by simply introducing high amounts of noise into the sample data, a higher degree of information on the classification behavior of the model enables an adversary to minimize the amount of manipulation necessary to achieve the intended impact, making discovery and mitigation of the attack more difficult. It has been shown, for example, that well performing image classification models can be caused to fail by introducing targeted noise imperceptible to the human eye (see Figure into the images. 5.6) [75].

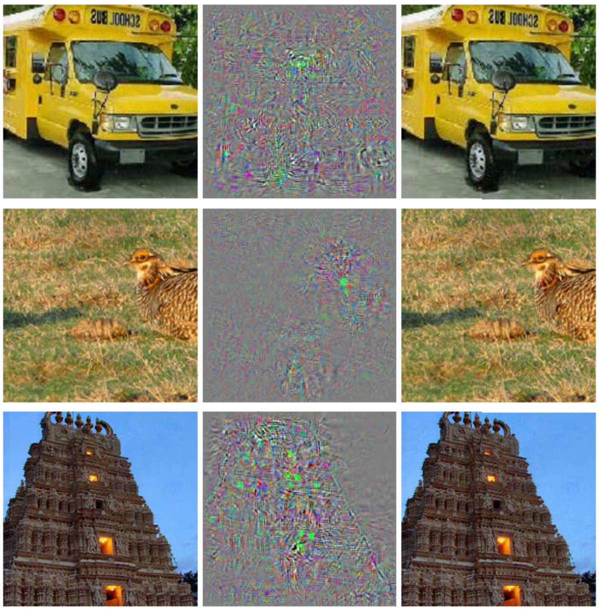


**Figure 5.6:** Three examples of correctly classified images (left), falsly classified images and their difference (middle) from experiments on the AlexNet-classifier [75].

### 5.3.5 Inherent Threats

There are several common issues with the application of artificial intelligence that are often not directly related to a potential breach of security but rather towards its ethicacy and reliability in performing the tasks it was designed for. Therefore their consideration is essential to avoid potential legal liabilities and guarantee user trust.

**Ethical Problems.** There are four main ethical issues commonly discussed by the scientific community and increasingly also the public which are accountability, transparency, responsibility, and fairness [62, 67]. The notion of *accountability* in relation to artificial intelligence is important in all cases where the output of a model has direct influence on the real world, e.g. an automated driving system. Since no learning model is completely free of errors, it has to be clear in advance to all parties involved who is accountable for any potential negative effects caused by it. Second we consider *transparency*, which is probably the most discussed ethical concern of the four presented here. Even though the application of AI has shown positive results in many use cases like the analysis of medical scans, the question of how the model came to one specific conclusion is still an open question in many areas [67]. While the majority of techniques used in the 20th century relied on decision processes that were potentially interpretable by humans - so called symbolic AI - many popular methods used today like neural networks utilize a form of abstraction that prevents any immediate traceability of their decision processes by humans. However, with the ever growing popularity of artificial intelligence as a key part of business concepts and consumer products, new regulations like Article 13 and 14 of the GDPR have emerged that raise the issue of providing transparency for AI-based decisions from an optional to a legally mandated matter [2]. The notion of *responsibility* in the context of artificial intelligence means the question what tasks an AI-driven system should be allowed to perform. While from a legal perspective this is determined by law, this also influences the public acceptance of any such system as the public opinion on a specific use case might not be in accordance with its legality in front of the law [67]. Lastly we consider the concept of *fairness*, which has become another important topic in recent years due to the discovery of internal bias in AI systems used in the context of healthcare and medical devices towards specific population groups [76]. All learning models used in artificial intelligence extract their knowledge from a set of training data. This implies that any biases represented within this data are also likely to be learned and reproduced by the model [77]. There are multiple types of biases that originate mostly from general patterns of human behavior and are often unconscious (see Table 5.1) [77, 78, 79, 80].

| Bias | Description |
|---|---|
| Reporting Bias | Introduced through over-/underreporting of certain real-world data due to various factors. |
| Selection Bias | Introduced through faulty methods of data collection, e.g. missing randomization, limited sample-population, etc. |
| Automation Bias | Introduced through human tendency to favor information generated by automated systems over non-automated systems. |
| Overgeneralization Bias | Introduced through faulty assumption that patterns from one data set apply to any other data set of the same domain. |
| Group Attribution Bias | Introduced by applying stereotypes to groups that are based on the characteristics of individual members (e.g. In-/Out-group bias). |
| Implicit Bias | Introduced through faulty assumption of group characteristics based upon generalization of personal characteristic. |

**Table 5.1:** Types of biases that can be found in data [78, 79]

**Improper Algorithms.** A seemingly obvious but rarely discussed pitfall is the selection of the correct algorithm for a specific task. Different AI techniques tend to perform differently when confronted with the same problem [53]. This can have different reasons from one method utilizing a form of abstraction that is closer to the real world domain than another technique up to wrongly chosen hyperparameters which are not adjusted during the training process but impede the performance of a system. This problem is related to transparency as it also emphasizes the importance of being able to understand and interpret the decision process of a given algorithm, not only to increase trust but also to understand its suitability to a given task.

**Improper Training Data.** The importance of training data for the success of an AI-based system has been emphasized multiple times throughout this chapter. However, while up until now we were mostly concerned with external dangers to training data through adversarial manipulation, we also have to consider the threat of training data being of inherently poor quality, either due to errors in their collection or selection of unsuitable data by the developers [53, 62]. Many machine learning techniques are highly sensitive to any shift in distribution between training and test data, meaning that a model that performs perfectly during training might suffer from drastically reduced performance during deployment because the training data did not accurately represent the real world environment [53].

| ID | Threat |
|---|---|
| AI.1 | Data Poisoning |
| AI.2 | Lack of Accountability |
| AI.3 | Lack of Transparency |
| AI.4 | Lack of Responsibility |
| AI.5 | Data Bias |
| AI.5.1 | Reporting Bias |
| AI.5.2 | Selection Bias |
| AI.5.3 | Automation Bias |
| AI.5.4 | Overgeneralization Bias |
| AI.6 | Improper Algorithms |
| AI.7 | Improper Traning Data |

**Table 5.2:** List of AI-specific threats which we consider relevant to HDIS. The comprehensive list can be found in our Threat Repository [81].

### 5.3.6 Relevance to Health Data Intelligence Systems

Considering the previously mentioned lack of quantitative data on attacks on AI systems, conducting a meaningful ranking of AI threats is difficult. Therefore we discuss the relevance of the presented threats to Health Data Intelligence Systems based on their likely applicability to our use case. A list of relevant AI-specific threats to HDIS discussed in this section is shown in Table 5.2. The comprehensive list with all corresponding information can be found in the Threat Repository [81].

**Training Attacks.** There are multiple factors that would suggest that training attacks might be highly relevant to Health Data Intelligence Systems. First we have to consider data collection. According to the system characteristics established in Chapter 2 and the information extracted from the project proposal, we assume the HDIS to gather its data from a multitude of sources both open and confidential. As discussed in section 5.3.3, the main vector of attack against the training phase of a learning algorithm is through poisoning, i.e. manipulation of the training data. Considering the high demands to transparency that come along with being legitimized and funded by public means, it can be expected that information about the data sources will be for the most part freely accessible, making those sources vulnerable to interference from potential adversaries. However, while training attacks could be an important issue, the attack surface will likely be somewhat limited as we believe the HDIS to be unlikely to involve any form of public user application that would allow for continuous and difficult to control push-based data ingest from a large number of devices.

**Inference Attacks.** We consider the risk of inference attacks on Health Data Intelligence Systems to be rather low. The nature of these attacks generally requires some form of access to the model. The most common scenario is for an attacker to have semi-supervisised access to the system, which means utilizing a publicly available API provided by the developers to interact with the system. While this API restricts access by limiting and controlling the possible interactions between attacker and system, it can contain technical weaknesses that can be exploited. The other variant would be unsupervised access, which is gained by hacking into the application backend, allowing for potentially unlimited access to information. Since the HDIS in its current form is not intended to feature semi-supervised interaction with users through an API, there is no convenient route of attack for adversaries short of breaking into the system itself via conventional means of compromising a system. However, the output of the HDIS will most likely be made public in some form as it is supposed to be used by governmental and healthcare authorities. Since the outputs have been produced through the processing of personal data, there is a potential threat of the outputs being used to deduce valuable information about the underlying training data or extract individual points from them. Apart from that, the biggest risk of exposing training data to unauthorized parties will be on the side of the data providers and during the transfer process from them to the controller. However, safeguarding the data during transfer and processing is not specific to artificial intelligence and has to be addressed in the scope of general system security.

**Inherent Threats.** The question of accountability is somewhat vague with regards to Health Data Intelligence Systems, as it doesn't have any immediate influence on the real world, but is rather intended as a decision support system for various authorities. However, even though the decision if and to which degree the HDIS's outputs will affect the real world is ultimately up to its addressees, the responsibility for maintaining the trustworthiness of the system in the eyes of the public and said authorities lies with the developers and operators. Therefore, a clear chain of authority among these two groups is required, as well as a comprehensive system of roles and responsibilities to establish individual and group accountability for the legality and quality of each processing step.

Transparency is a major issue for HDIS due to the utilization of personal data for its processes as well as its reliance on public opinion and trust for funding and legitimization. Of the four ethical concerns discussed before, transparency is maybe the most thoroughly defined through its significance in data protection law. Thus, developers and operators must be highly aware of their individual responsibilities with regards to legal transparency requirements concerning their particular field of competence to avoid an erosion of trust that might ultimately lead to a withdrawal of the governmental mandate.

The question of responsibility is essential to any application of AI where humans are directly or indirectly affected. In case of HDIS, this aspect needs to be firmly established ahead of the development process by determining what kinds of data may be used and by which means they may be processed. On the other hand, the impact of its outputs is mostly determined by the authorities themselves, as its main task is only to provide additional information and not to make any binding decisions by itself.

The notion of fairness is again highly important for the very reasons mentioned in its description in the previous section. The degree to which pandemic events affect a population may vary between different social groups and have many different and seemingly unrelated factors playing into it. These dynamics will then often also be reflected in the data documenting such events and have to be accounted for in order to prevent any underlying systemic bias towards or against any group from affecting the systems output. This is particularly important with regards to the selection and collection of training data, but can also affect how the results produced by the HDIS are handled by the recipients. From the six types of biases presented in Table 5.1, we should be concerned mainly with the first four. Reporting and selection bias can both occur during the data collection process for the operation of HDIS. A common example for the former is the regular drop of reported COVID-19-cases each weekend, which has been attributed to the fact that most doctors in Germany, who are one of the primary sources for data about the number of new infections, don't work on Saturdays and Sundays [82]. Selection bias on the other hand could be introduced through negligence in the selection of data sources for the models employed by HDIS. Automation bias is more important for the HDIS outputs, as their significance could be overestimated or misinterpreted by recipients with no previous experience with artificial intelligence. Due to the possibility of overgeneralization, all results also

| ID | Mitigation |
|---|---|
| M.AI.1 | Quality Metrics for Data Sources |
| M.AI.1.1 | Assessment of Reputation & Independence |
| M.AI.1.2 | Assessment of Source Integrity |
| M.AI.2 | Quality Metrics for Data Sets |
| M.AI.2.1 | Automation of Metric Application |
| M.AI.2.2 | Metrics for Relevance |
| M.AI.2.3 | Metrics for Up-to-Dateness |
| M.AI.2.4 | Metrics for Completeness |
| M.AI.2.5 | Metrics for Consistency |
| M.AI.2.6 | Metrics for Accuracy |
| M.AI.2.7 | Evaluation of Collection Methodology |
| M.AI.2.8 | Metrics for Outlier Detection |
| M.AI.3 | Robust Learning |
| M.AI.3.1 | Reduction of Features |
| M.AI.3.2 | Reduction of Value Space |
| M.AI.4 | Assignment of Roles and Accountability |
| M.AI.5 | Provide Explainable Models |
| M.AI.6 | Definition of Modalities for Model Application |
| M.AI.7 | Performance Evaluation for Learning Algorithms |
| M.AI.8 | Metrics for Predictive Performance |

**Table 5.3:** List of potential mitigation strategies for AI-specific threats which we consider relevant to HDIS. The comprehensive list can be found in our Threat Repository [81].

have to be validated in order to prevent the extrapolation of knowledge gained from subsets of data to similar domains where they might not apply.

Lastly, the threats of improper algorithms and training data are general issues that apply to all applications of artificial intelligence and therefore also to Health Data Intelligence Systems. While the extent of this issue is impossible to determine precisely before the actual development process, it stands to reason that the planned combination of various different techniques and data sources by the research group necessitates particular attention towards both these aspects as any increase in complexity of the overall processing also increases the risk of utilizing non-ideal methods for particular tasks or incorporating training data of inferior quality.

### 5.3.7 Mitigation Strategies

In Section 5.3.6, we identified poisoning attacks, the inherent ethical problems of AI, as well as the risk of improper algorithms and training data as the most relevant AI-specific threats to Health Data Intelligence Systems. This now poses the question as to how these threats can be effectively mitigated. In this section we briefly present some common mitigation approaches that will be considered for the design of our reference architecture. A list of relevant mitigation strategies discussed in this section is shown in Table 5.3. The comprehensive list with all corresponding information can be found in the Threat Repository [81].

**Mitigating Training Attacks.** The scientific community has proposed various mitigation strategies for training attacks which act on different steps of the learning process. The first step at which traning attacks can be prevented is during data collection by detecting adversarial data points before they can influence the learning model [54, 83]. To achieve this, a commonly suggested technique is the use of data sanitization, using different metrics of purity extracted from trustworthy sets of training data to identify and remove manipulated data points from other data sets [61, 66, 84]. This should also be applied to HDIS, specifically for data collected from public sources like Wikidata that can be easily accessed and edited by potential adversaries.

Another frequently mentioned approached is the employment of so called *Robust Learning*, which is an umbrella term for a number of techniques aimed at making learning algorithms more resilient against poisoned training data [55, 59, 66, 84, 85]. Some of these techniques are universally applicable like a reduction of the number of analyzed features as well as the respective value space of each feature to a necessary minimum [55, 84, 85]. While this weakens the capability of attackers to create potent adversarial data points as they have a smaller range of features and values to chose from, this may also negatively impact the performance of the learning model for more complex domains [70]. However, there also exists a large variety of techniques that are bound to specific algorithms or application cases and can therefore not be generalized to HDIS, since both are not sufficiently specified yet to allow for consideration of these specialized approaches [55, 65, 73].

**Mitigating Ethical Concerns.** The first ethical issue we need to consider is accountability, which has to be addressed mainly on an organizational level. It requires a clear definition of organizational structures which precisely describes individual roles as well as their respective competencies and responsibilities for organizations as well as individual staff involved in the operation of AI-based systems. A rough outline of some of these roles is provided by data protection legislation, namely the roles of data subject, controller, processor and supervisory authority as well as the staff role of Data Protection Officer which are all defined in Article 4 of the GDPR [2].

Regarding transparency, we are fortunate to have a set of extensive legal requirements provided by the GDPR and BDSG which facilitates the selection of proper measures and provides a rough outline of the information that has to be provided to the data subjects. While most of this information can be provided fairly easily, Article 13 and 14 of the GDPR also require the controller to provide the data subject with meaningful information about the logic used for automated decision-making. Not only is this requirement somewhat vague, it also touches on the problem of explainability of artificial intelligence, i.e. the question how an AI-system arrived at a specific conclusion given the information it was provided [86, 87, 88]. Providing this information is particularly complex for subsymbolic artificial intelligence like neural networks, for which suitable methods are mostly specific to certain algorithms and application cases [86, 89].

As already mentioned in Section 5.3.6, the suggestions and analysis generated by HDIS don't directly impact humans, which is why the responsibility of operating such systems is determined mostly by the degree to which it will impact political and industrial decisions which in turn affect the wider population. We propose a number of modalities which we consider to be important and should therefore be determined, recorded, and approved beforehand by the supervising authorities in cooperation with the controllers and processors.

1. Should the results be publicly available or limited to select recipients?
2. Should there be limits to the possible recommendations made by HDIS?
3. How should the results be formatted?
4. What supplementary information should be provided with the results?

Our first question regards the recipients of the results generated by the HDIS. While the outputs will always be disclosed to at least the supervisory authority and representatives of institutions responsible for their evaluation and potential implementation, it should be considered if and how they should also be made publicly available. While the idea of a public release is beneficial in terms of transparency, its legality depends on the exact content of the results and if their disclosure would entail privacy concerns or other detrimental side effects. Question two basically considers the necessity of limiting the solution space that can be navigated by the HDIS. This might be necessary to prevent suggestions whose implementation would be infeasible, come into conflict with the law or would otherwise be considered unethical. Thirdly, we have to determine the format in which the results will be presented. This is important as the raw outputs of artificial intelligence algorithms are usually highly abstract and require some degree of treatment to be interpretable, especially for technical laypersons. To prevent the introduction of unnoticed or unconscious bias into the results during this process, the ways in which they are treated should be clearly defined as to be traceable for the recipients. The last question extends the previous point by asking what additional information should be provided to the recipients to improve their understanding of the results. This is crucial since there might be contextual aspects to the results which seem obvious to the technical specialists operating the HDIS, but not so to the recipients. The extent of additional information might vary depending on the degree of available technical knowledge for each recipient.

In Section 5.3.6 we established that fairness is a significant concern for Health Data Intelligence Systems with the main issues being reporting, selection, automation and overgeneralization biases. Mitigating the detrimental effects of biased data on an algorithmic level is a heavily studied problem in the scientific community and many algorithmic approaches have been proposed to improve the quality of biased data during pre- and post-processing [77, 80]. However, the ideal solution would be to prevent such biases from being introduced into our data in the first place. To achieve this, the processes of collecting training data has to be designed with bias mitigation in mind. To achieve this, it is important to implement comprehensive quality criteria that can be applied during selection of data sources and validation of individual data sets provided by them.

**Data Quality.** Of the issues mentioned in Section 5.3.6, training attacks and fairness-concerns in particular can be directly addressed by ensuring the quality of data used for artificial intelligence processing. To achieve this, a number of metrics has been proposed and established over the years that allow for the assessment of data quality based on certain criteria. Table 5.4 presents a list of popular categories of metrics that are commonly applied for data validation in database management systems and machine learning and can be fully or at least partially automated [83, 79]. Applying such quality metrics to the various input data contributes to some degree to the mitigation of nearly all AI-related issues we consider relevant to HDIS.

We already mentioned robustness as a popular approach to reduce the impact of training attacks. However, classical database management metrics also reduce the risk of such attacks by detecting very simple forms of data manipulation like the duplication of entries or an excessive amount of missing values that might be filled in with default values. Regarding accountability

| Domain | Quality Metric | Description |
|---|---|---|
| Database Management | Data Correctness | Identifies schema violations within the data, such as discrepancies between data types and values. |
| | Data Consistency | Identifies non-syntactic defects, e.g. duplicate entries, invalid values. |
| | Data Completeness | Determines the ratio of missing values with a data set. |
| | Statistical Properties | Calculates basic descriptive statistics that can serve as indicators individual and more complex quality metrics, e.g. mean, mode. |
| Machine Learning | Predictive Performance | Serves as indicator for model performance and are often used for optimization, e.g. accuracy, precision, F-score. |
| | Robustness | Evaluates the impact of data sets on the model performance to identify potentially harmful data sets. |
| | Generalization | Evaluates the generalization performance of a model on other data sets from the same domain, e.g. cross-validation. |
| | Fairness | Evaluates the performance of a model w.r.t. (dis-)advantaging specific groups represented within the training data. |
| | Privacy | Evaluates if trained model allows for the identification of individual data points used in the training set, e.g. differential privacy. |

**Table 5.4:** Common metrics for evaluating data quality in general and for machine learning applications specifically [79, 83]

and transparency, objective metrics provide contextual information for the documentation of operation and performance of an AI-system that can be particularly useful when reporting to technical laypersons. Their relation to responsibility is simpler, as maintaining a good quality of data is a necessary prerequisite of producing models of sufficient quality to be responsibly applied to real world-applications. While fairness in itself has already been mentioned as its own category in Table 5.4, it can also be supplemented by database management metrics as they allow the operators of an AI-system to ensure that different groups from any sample population are not underrepresented due to incompleteness or other basic deficiencies in data sets focused on them.

## 5.4 General System Security Threats

In this section we consider threats to general system security. We'll first examine available sources for threat intelligence in this area and discuss their usability for our research. Afterwards, we present our methodology for compiling a list of the most important high-level security threats and discuss our findings. The results of this section are also contained within the Threat Repository [81].

### 5.4.1 Sources for Threat Intelligence

There is a large number of sources dedicated to general weakness and threat intelligence, some as written catalogs, but most in the form of cloud based knowledge bases. Due to the wide range of aspects and perspectives that need to be considered in threat and risk analysis, the focus of these knowledge bases also varies considerably. *OpenCTI*[2] and the EU-funded *MISP Threat Sharing*[3] focus on collecting and publishing concrete threat events and descriptions of threat actors in cloud databases that can be updated and extended by the community [90, 91]. The *Common Attack Pattern Enumeration and Classification (CAPEC)*[4] and *Adversarial Tactics, Techniques & Common Knowledge (ATT&CK)*[5], which are both operated by the MITRE corporation, aim to provide a more conceptual overview of threats and attacks, forgoing the description of specific events for a more compact list of techniques commonly employed by attackers [46, 47]. MITRE additionally operates the *Common Weakness Enumeration (CWE)*[6] and *Common Vulnerability Enumeration (CVE)*[7] which are structured similarly to CAPEC and ATT&CK but focus on the eponymous concepts as they were defined in Section 5.1 [48, 49]. For all four knowledge bases, new entries can be proposed by anyone but are vetted by professional employees before being added to the database to ensure a consistent level of quality.

Many of the works about methodology that we discussed in Section 5.2 also contain lists of weaknesses, vulnerabilities, and threats, mostly from a highly abstract perspective [44, 45]. An expansive collection of potential threats is contained within the *IT-Grundschutz-Kompendium*, which not only has been updated during the making of this thesis, but whose inclusion might also prove important with regards to our goal of providing legal compliance with German data protection regulations at it is supposed to be used as a standard by authorities to evaluate system security [28].

As mentioned before in Chapter 3, a general problem with static catalogs is the danger of them quickly becoming outdated and therefore either containing entries that might be obsolete nowadays or potentially missing recent and more common threats. Therefore, if static resources are to be considered, it should be ensured that they are intended to be updated regularly. One good example for this is the *ENISA Threat Landscape (TLS)*, which is a yearly publication by the European Union Agency for Cybersecurity and presents a regularly updated list of the most relevant threats to interconnected information systems [58]. The TLS is also interesting since the fact that it is an official document released by the European Commission implies that it could serve as a reference for the implementation of threat mitigation-requirements in EU legislation. A similar case with regards to German regulations can be made for the *IT-Grundschutz-Kompendium*, which contains an extensive catalog of recommendations on a wide

[2] https://www.opencti.io/en/

[3] https://www.misp-project.org/

[4] https://capec.mitre.org/

[5] https://attack.mitre.org/

[6] https://cwe.mitre.org/

[7] https://cve.mitre.org/

range of aspects regarding information and system security, released by the Federal Office for Information Security [28]. These recommendations are based on a list of 47 high-level threats that should be considered during development of all information systems.

Considering the planned prioritization of threats outlined in Section 5.2.1, we also want sources that provide us with some kind of metric to rank them based on their likelihood and severity. Some sources like CAPEC support these limitations in scope by providing the user with various metrics like the risk of an attack or its potential impact severity. Other sources instead opt for a more accessible but less differentiated approach by providing top-lists that present a selection of threats deemed the most relevant by their authors. The CWE for example doesn't contain the aforementioned risk- and severity-metrics but instead offers a yearly list of the 25 most important software weaknesses which is curated by experts from MITRE [49]. The same approach is used for the TLS which contains extensive descriptions and background information about the ten threats considered most crucial by the ENISA.

### 5.4.2 Methodology

After reviewing the available sources, we decided to use CAPEC and CWE as our primary sources for external and internal threats respectively. Among their multiple advantages, the most important to mention are their professional curation, ongoing support, comprehensive description, and interconnection of individual entries as well as their offered functionality for customized filtering. Beyond that, we also use ATT&CK as a supplementary source for CAPEC, as it provides additional practical information as well as additional mitigation strategies to various entries from CAPEC. Lastly, we include the ENISA Threat Landscape and the GSK to confirm all existing threats and ensure some measure of completeness from the perspective of the relevant legislators. Both contain internal as well as external threats and can therefore be applied to both perspectives.

The general process of compiling threats into our repository is visualized in Figure 5.7 and shows some similarities to the process used for requirement extraction in Chapter 4.2. First we select the next source and apply any available filtering options to reduce the number of threats to a feasible level. Afterwards, we extract the next threat in free text form and compare it to the ones extracted before. In the case of duplicates, we simply add a new reference to the existing entry and continue. Otherwise, we add a new threat entry to the repository and research potential mitigation strategies to this specific threat. If any new strategies are discovered, we add those to our list of mitigations contained within the repository. Lastly, we add references to the threat source as well as to relevant mitigation strategies to the threat entry.

For internal weaknesses we use the CWE as our primary source. To focus on the most relevant threats, we considered the 2021 CWE Top 25 list of the most dangerous software weaknesses [49]. The list is then trimmed by removing all technology-specific weaknesses as they would go against our principle of ubiquity. Afterwards we compare the resulting set of weaknesses to the TLS and GSK and add all weaknesses not yet mentioned.

For CAPEC, we can't rely on a curated list of the most significant attack strategies and therefore have to use the manual filtering function to narrow down the selection of threats. The repository allows to filter for the severity of an attack pattern, which means the potential damage that one particular strategy can cause to a system, as well as the likelihood of that pattern being used by attackers. We use these two options to only keep those threats that have a likelihood of at least *high* and a *very high* severity. We then follow the same procedure as the CWE by further removing all technology-specific attack patterns and finally completing the list using the TLS and GSK. The complete documents, including descriptions and linked references can be found on the thesis repository [81].

### 5.4.3 Weaknesses

We begin the discussion of our results with the weaknesses extracted from the CWE. Excluding all but one of the technology-specific weaknesses from the 2021 CWE Top 25 resulted in an initial set of 15 entries. After cross referencing this list with the TLS and GSK and adding missing weaknesses from those sources, we ended up with a total of 18 high-level weaknesses to consider (see Table 5.5) [81]. We decided to keep *W.14* despite its relation to a specific technology due to the particular relevance and ubiquity of XML in data-intensive systems in general. Although the weaknesses vary greatly in terms of scope and which part of a system they affect, we identified four main security issues that are related to multiple weaknesses and can help us getting an initial overview over the most important areas of concern resulting from this list.

**Access Management.** Multiple weaknesses in our list are concerned with missing or inadequate mechanisms for accessing critical services or resources. *W.1*, for example, is concerned with faulty authentication mechanisms that allow for the impersonation of users with higher levels of access by malicious users. *W.2* simply regards the complete absence of authorization mechanisms for important resources and services, leaving them open to manipulation and unintended disclosure. Our list also contains some more specific weaknesses that have to be tackled on an implementation-level, like *W.4* which describes the lack of technical restrictions on memory buffers, leaving the system vulnerable to common threats like buffer overflow-attacks.

**Information Disclosure.** One of the most common consequences of a majority of weaknesses in our list is the accidental or unlawful disclosure of sensitive information to unauthorized users. This can be caused by a variety of issues like the aforementioned lack of authorization checks or an erroneous assignment of permissions to a specific resource, allowing it to be accessed by the wrong user group (e.g. see *W.2*, *W.8*). Information disclosure can also be caused on an implementation level, for example through the use of hard-coded credentials (see *W.9*) like passwords or cryptographic keys, which are used routinely for authentication or encryption.

**Missing Data Validation and Sanitization.** Another recurring topic among our list is a lack of validation and sanitization on various forms of data supplied by external entities and then used within the system. Systems managing large amount of data commonly feature the automated construction of commands and database queries using data provided by external sources (see *W.11*). If this data is not properly sanitized, this automation can open a path of attack for malicious users by injecting custom commands into the data they provide to the system. All weaknesses from *W.12* to *W.16* cover similar scenarios where incoming data from known and unknown sources alike is not sufficiently sanitized or validated, allowing for various kinds of impacts ranging from compromising sensitive data to the misuse of computing resources.

**Negligence.** The last recurring theme is human negligence, particularly by users and administrators. *W.10* describes how otherwise adequate security mechanisms can be compromised by misconfiguration, for example through the use of default accounts and passwords which are openly available to potential attackers, while *W.18* regards the risk posed by outdated or

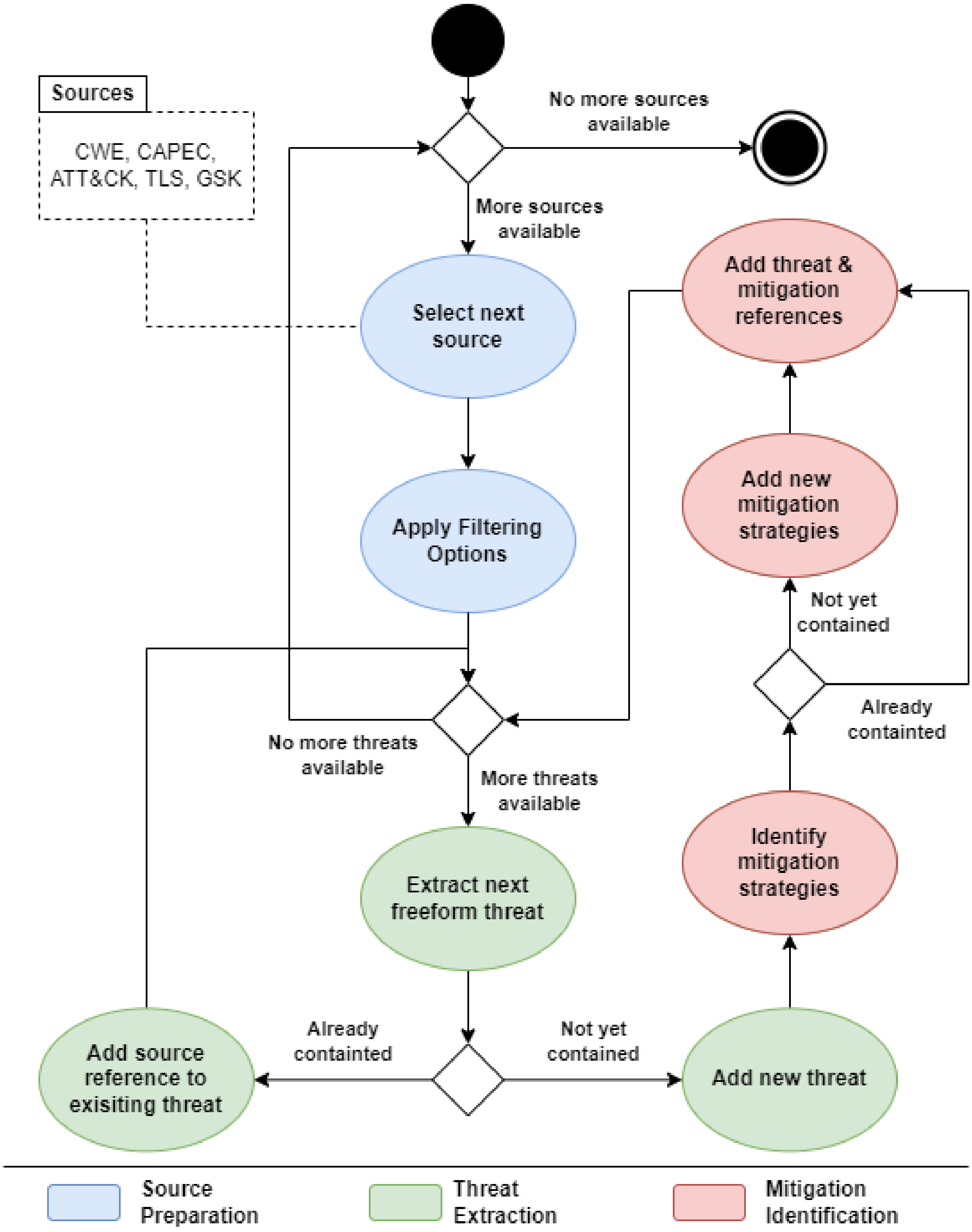

Sources
CWE, CAPEC, ATT&CK, TLS, GSK
No more sources available
More sources available
Select next source
Apply Filtering Options
Add threat & mitigation references
Add new mitigation strategies
Not yet contained
Already containted
No more threats available
More threats available
Extract next freeform threat
Identify mitigation strategies
Already containted
Not yet contained
Add source reference to exisiting threat
Add new threat
Source Preparation
Threat Extraction
Mitigation Identification

**Figure 5.7:** Process for compiling the repository of general system threats.

| ID | Weakness |
|---|---|
| W.1 | Inadequate Authentication |
| W.2 | Missing Authorization |
| W.3 | Inadequate Limitation of a Pathname to a Restricted Directory ('Path Traversal') |
| W.4 | Inadequate Restriction of Operations within the Bounds of a Memory Buffer |
| W.5 | Integer Overflow or Wraparound |
| W.6 | Exposure of Sensitive Information to an Unauthorized Actor |
| W.7 | Insufficiently Protected Credentials |
| W.8 | Incorrect Permission Assignment for Critical Resource |
| W.9 | Use of Hard-coded Credentials |
| W.10 | Security Misconfiguration |
| W.11 | Inadequate Neutralization of Special Elements used in a Command ('Command Injection') |
| W.12 | Unrestricted Upload of File with Dangerous Type |
| W.13 | Deserialization of Untrusted Data |
| W.14 | Inadequate Restriction of XML External Entity Reference |
| W.15 | Server-Side Request Forgery (SSRF) |
| W.16 | Inadequate Input Validation |
| W.17 | User Negligence |
| W.18 | Outdated Software |

**Table 5.5:** List of system and software weaknesses from our Threat Repository [81].

obsolete software, particularly the increased susceptibility to zero-days and similar exploits, which are often fixed through dedicated updates by the affected companies. Finally, *W.17* encompasses all types of user negligence. This is probably the broadest weakness of the list as its entry in the TLS encompasses all types of unintentional human misbehavior which can either compromise a system and its operation on their own or be used as vectors by malicious actors. Basic examples would be the loss of hardware containing sensitive data or the introduction of insecure hardware into the system network (e.g. personal USB-sticks).

### 5.4.4 Attack Techniques

Using the filtering options described in Section 5.4.2, we compiled an initial list of 23 attack techniques from the CAPEC knowledge base. After cross referencing and supplementing these, using the ATT&CK knowledge base as well as the TLS and GSK catalogs, we ended up with a final list of 30 entries (see Table5.6). It is theoretically possible to differentiate between intrusion and impact techniques, with the former describing strategies to invade a target system and the latter meaning ways to affect an already invaded system in a harmful way. However, the lines between both categories are extremely vague and don't offer much additional insight. Instead we can identify certain patterns with regards to the most common targets of each technique, that can help us in identifying aspects of a system or organization that need particular consideration in the design of security measures.

**Authentication Mechanisms.** One common target for multiple attack techniques are authentication mechanisms of a system. *AT.1* presents probably the most basic form of such a technique where the attacker attempts to gain entry by simply trying a large number of possible credentials using randomness, previously gained information and heuristic support in the hope of finding a valid set of credentials at random. A more sophisticated way of circumventing authentication barriers is the abuse of session-mechanisms which are common for web-based systems in order to enable authentication for large numbers of users over multiple message-exchanges, as described in general by *AT.2* and for active programs in particular by *AT.3*. Lastly, *AT.4* and *AT.5* consider different ways of delivering exploits that take advantage of such session mechanisms via scripts embedded into websites or URLs respectively.

**Input Data.** The second recurring target group is data which is provided to the system by external sources and then used as inputs for internal functions. The pervading goal of attackers for this kind of attack is to introduce malicious code into the process-flow of a running program or service. *AT.6*, *AT.7* and *AT.8* describe three well-known variants of this. The first takes advantage of missing boundary restrictions for memory buffers in order to overflow the pre-assigned memory space and write malicious commands to the execution stack. The latter two abuse the insertion of unsanitized string-inputs during the automatic creation of queries for SQL databases and XML documents respectively, which are then executed with the active user's or process' privileges. Even though both entries are tied to specific technologies, we again chose to include them due to SQL and XML being ubiquitous technologies in the design and operation of data intensive systems. *AT.9* considers a different form of input manipulation, where the attacker uses a vulnerable file search-function to escape from intended directories and access other areas of the files system that are supposed to be private. *AT.10* and *AT.11* describe techniques to fit more complex file types like binaries or document files with malicious payloads which are executed upon upload to the system. Lastly, *AT.13* is concerned with user controlled variables that are set before startup and can be abused by attackers to cause the system to be operated in a vulnerable configuration.

**Sensitive Data.** While input data is targeted mostly as a potential intrusion vector, sensitive data can also be the impact target of an attack. Here, the attacker's goal is to either manipulate, steal or destroy the data in order to produce a negative impact. *AT.14* describes a rather specific variant of this, where an attacker abuses their access to build-files to redirect the inclusion of external libraries into a software to an unintended target, allowing them to introduce their own code into the implementation. In a similar manner the manipulation of configuration files, as mentioned in *AT.15* can be used for various means like disabling other security functionalities or inserting credentials that allow unauthorized users to gain access to administrative functionalities. *AT.16* outlines the threat of attackers extracting sensitive data which was embedded in the implementation, for example in the form of hard-coded credentials or inadequately secured encryption keys. Ransomware describes a category of malware which is used to either encrypt or steal sensitive data from a target's system in order to blackmail the owner in exchange for decryption or non-disclosure. Lastly, we have two rather mundane threats with *AT.18* and *AT.19*, that nevertheless cannot be ignored when considering security from a holistic viewpoint as hardware in general and mobile hardware in particular can be subject to physical theft or destruction, resulting in the contained data being either stolen or destroyed as well. While non-mobile hardware is easier to protect from theft or intentional destruction, it is nonetheless susceptible to fires, flooding and other natural events (see *AT.20*) which therefore also has to be accounted for to some degree.

**Communication & Network.** If an attacker is not or not yet able to invade a target system itself, a common tactic is to attack lines of communication or the network in which the system operates. *AT.21* describes so called *Adversary in the Middle*-attacks where the attacker intercepts communication over an insecure channel like an unencrypted FTP-connection. Depending on the attacker's previous knowledge about encryption and communication protocols, the intercepted messages can be used for a variety of means like extracting sensitive information, misinformation or conducting replay attacks. Denial of Service-attacks (DoS) are one of the most common scenarios in today's threat landscape [58]. The basic concept is to disable a network or system by flooding it with requests in order to overwhelm its processing capacities. DoS-attacks usually target either the network in which a system operates aiming to exhaust its communication bandwidth (see *AT.22*) or the system directly by sending their requests to one or more ports used to receive outside messages (see *AT.23*). Lastly, *AT.24* presents a more specific scenario in which an adversary takes advantage of remote control-systems between different computers in an organization network to send malicious commands to more privileged terminals in order to be executed there.

**Users.** The last major category of targets are users themselves as their manipulation often requires little technical know-how and instead relies heavily on various forms of negligence. Pharming, as introduced by *AT.25*, describes the act of mimicking a target's website or service with the goal of luring users into disclosing sensitive data like login credentials via fake login forms or other interfaces that appear legitimate at first glance. Phishing is another form of manipulation that directly targets members of an organization, usually via phone or Email, in order to make them disclose sensitive information or introduce malicious files or programs into the organization network. These payloads are usually delivered as an attached file or URL which the attacker asks the target to open under the guise of a fake identity. The general act of disguising a malicious action as a harmless, e.g. by hiding a download behind a seemingly innocuous button on a website, is called Action Spoofing (see *AT.27*) and is a common component of the previous two techniques. *AT.28*, on the other hand, describes a frequent consequence of Phishing in particular, which is the automatic execution of malicious scripts or download of files on the first visit of an attacker's website by the victim.

Disinformation describes the active spread of false information in order to guide a victim or group of victims into a specific course of action. Apart from its common use in social media to affect public opinion, disinformation can also be used to affect systems or organizations by presenting unaware employees with false warnings or other types of false information in order to provoke an action with negative consequences to the organization's systems or operation in general. Misinformation is similar to the previous entry, with the difference being that the false information is spread unintentionally by users or employees who are themselves unaware of its fallacy.

### 5.4.5 Mitigation Strategies

As mentioned in Section 5.4.2 and similar to our process for AI-related threats, we integrate the search for potential mitigation strategies into our research flow for general security threats. This time, our primary sources facilitate this process as the CWE, CAPEC and ATT&CK all include descriptions of mitigation strategies for each threat as part of the knowledge base entry. While, as a printed source, the TLS is less convenient to navigate, it also features a short list of defensive techniques at the end of each chapter. Considering the strategies offered by these sources, we compile a list of 48 general mitigation strategies (see Table 5.7). The complete list is part of our Threat Repository and contains a short summary of each mitigation strategy as well as a precise reference to its source [81]. In the following, we will discuss these strategies as a whole and what themes and patterns stand out among them.

| ID | Attack Technique |
|---|---|
| AT.1 | Brute-forcing of credentials |
| AT.2 | Session Highjacking |
| AT.3 | Target Programs with Elevated Privileges |
| AT.4 | Cross Site Scripting (XSS) |
| AT.5 | Cross Site Request Forgery (CSRF) |
| AT.6 | Buffer Overflow |
| AT.7 | SQL Injection |
| AT.8 | XQuery Injection |
| AT.9 | Path Traversal |
| AT.10 | Overflow Binary Resource File |
| AT.11 | Leverage Executable Code in Non-Executable Files |
| AT.12 | Using Malicious Files |
| AT.13 | Manipulating User Controlled Variables |
| AT.14 | Library Access Redirection |
| AT.15 | Manipulating Writable Configuration Files |
| AT.16 | Retrieve Embedded Sensitive Data |
| AT.17 | Ransomware |
| AT.18 | Intentional Destruction of Hardware |
| AT.19 | Theft of Hardware |
| AT.20 | Force Majeure |
| AT.21 | Adversary in the Middle (AitM) |
| AT.22 | Network Denial of Service (NDoS) |
| AT.23 | Endpoint Denial of Service (EDoS) |
| AT.24 | Manipulating Writable Terminal Devices |
| AT.25 | Pharming |
| AT.26 | Phishing |
| AT.27 | Action Spoofing |
| AT.28 | Drive-by-Compromise |
| AT.29 | Disinformation |
| AT.30 | Misinformation |

**Table 5.6:** List of attack techniques from our Threat Repository [81].

**Authentication.** The first common theme among the proposed mitigation strategies is a focus on improved authentication techniques and policies. Most of these are common features of cyber-security like the use of multi-factor authentication, the enforcement of password policies to decrease the likelihood of successful brute-force attacks or the minimization of the number of user accounts to a necessary minimum. CAPEC also recommends the use of token-based authentication for requests to the system by external entities thus preventing replay-attacks due to the binding of individual actions to specific sessions.

**Communication.** With regards to cross-network communication, mitigation strategies focus mainly on encryption and use of certificates. The former is self-explanatory as sensitive information should always be encrypted during transmission via public infrastructure. Certificates in the context of communication and the use of trusted third parties for their exchange are adjacent to authentication mechanisms as they facilitate the establishment of communication protocols and decrease the risk of identity spoofing by potential adversaries.

**Network Protection.** As the attack surface of institutional networks is rather extensive, so is the array of possible safeguards to reduce it. The majority of these can be realized with commercially available solutions like antivirus software, network level firewalls that allow for black- and whitelisting of certain traffic patterns, or application level firewalls to prevent malicious messages from reaching listening applications. The use of Email gateways is also recommended as they constitute one of the most common attack paths for phishing and similar social engineering attacks.

**Monitoring** Multiple mitigation strategies focus on the detection of internal weaknesses or suspicious behavior on various system levels. General monitoring of activities and communications at network boundaries and between internal systems can help detect behavioral patterns that might indicate adversarial attacks. The same holds true for computing resources in order to detect DoS-attacks. Regular integrity checks on important resources like configuration files reduce the threat of intentional introduction of vulnerabilities into a system during operation. Lastly, penetration testing by trusted specialists or internal red teams as well as automated vulnerability scanning by dedicated software-products helps with detecting and closing hitherto undetected attack paths.

**Permission Management.** A strict management of permissions to access and interact with critical resources is essential to reducing the risk of users causing harm to the system both intentionally and unintentionally. For this, the system should be compartmentalized into sections requiring different levels of authorization to access with the number of authorized personnel being held at a minimum for each level. These sections can then be used to restrict access to certain resources and functions to privileged users like the aforementioned configuration files or potentially dangerous tools like command lines. It can also be useful to analyse standard software used by a majority of users in daily operations for potentially dangerous features like macros and disable them if they aren't essential to organizational processes.

**System Administration.** Two often mentioned processes that are essential to increasing system security are regular update routines and configuration management. Most major IT-companies provide frequent patches and security updates to their systems and services which address newly found vulnerabilities that could or have been exploited before. The more ubiquitous the product in question, the more important it is to apply those updates as soon as possible as the popularity of a product is generally proportional with its value as a target for adversaries. The process of configuration management is also related to the use of off-the-shelve products as they are often designed to be easy to setup by using a set of default configurations provided by the developer. However, these default settings are often neither secure nor optimized towards the customer's specific use case and therefore need to be customized and maintained in order to prevent common weaknesses like easily guessable default user accounts or advanced security features that need to be activated manually.

**Implementation.** A number of mitigation strategies are directed towards the implementation itself as a lack of understanding of certain programming language and library features is a common source for vulnerabilities in newly developed systems. The two most commonly mentioned strategies in this context are input validation and boundary checking. Input validation in general is important as adversaries could otherwise take advantage of vulnerable functions for command injection or DoS-attacks. Boundary checking is a subset of input validation and particularly important for programming languages that allow for the assignment of values of arbitrary size to certain variable types. If not checked manually, this can be used by attackers to influence the execution stack and either cause breakdown or insert their own code into the execution flow. One common way of discovering these and other weak points in the code base of a system is the utilization of static code analysis, which is available as a general feature for most popular programming languages and can be extended with more specialized analysis tools that allow for the implementation of custom rules and code smells.

**Resilience & Recovery.** While it is always preferable to entirely prevent negative events for the system in the first place, this can never be guaranteed. It is therefore essential to have a tried and tested plan of action to mitigate the negative impact in case of such incidents and design the system with resilience and recoverability in mind. An important part of this is the setup of comprehensive backup strategies, that prevent the permanent loss of sensitive data. Additionally, potent encryption on stored data can decrease the impact of data theft. It can also be beneficial to implement safe failing mechanisms that enable an organized shutdown of the entire system in the event of a partial breakdown of individual components.

**Facility Maintenance.** Threats to a system are not always on a digital level but can also occur through physical impacts like hardware failure or physical theft. Therefore, proper maintenance of physical facilities is an essential aspect of system security as is proper access control to prevent unauthorized personnel from entering those facilities.

| ID | Mitigation |
|---|---|
| M.Gen.1 | Restrict Resource Permissions |
| M.Gen.2 | Multi-Factor Authentication |
| M.Gen.3 | Account Use Policies |
| M.Gen.4 | Password Policies |
| M.Gen.5 | Restrict Access to Command Lines |
| M.Gen.6 | Account Minimization |
| M.Gen.7 | Comprehensive Backup Strategies |
| M.Gen.8 | Response and Recovery Management |
| M.Gen.9 | Physical Maintenance |
| M.Gen.10 | Protection from Physical Intrusion |
| M.Gen.11 | Check Library Integrity |
| M.Gen.12 | Use Obfuscation |
| M.Gen.13 | Restrict Access to Configuration Files |
| M.Gen.14 | Use Boundary Checking |
| M.Gen.15 | Input Validation |
| M.Gen.16 | Antivirus/Antimalware |
| M.Gen.17 | Network Intrusion Prevention |
| M.Gen.18 | Restrict Content |
| M.Gen.19 | General Activity Monitoring |
| M.Gen.20 | Monitoring of Computing Resources |
| M.Gen.21 | Use of Black/Whitelisting |
| M.Gen.22 | Company-Wide E-Mail Gateways |
| M.Gen.23 | Disable Potentially Dangerous Program Features |
| M.Gen.24 | Filter Network Traffic |
| M.Gen.25 | Monitoring of External Data Sources and Data Quality |
| M.Gen.26 | User Training |
| M.Gen.27 | System Administration & Update Routine |
| M.Gen.28 | System Administration & Configuration Management |
| M.Gen.29 | Vulnerability Scanning |
| M.Gen.30 | Malware Detection |
| M.Gen.31 | Data Loss Prevention (DLP) Tools |
| M.Gen.32 | Application Level Firewall |
| M.Gen.33 | Network Level Firewall |
| M.Gen.34 | Content Delivery Networks (CDN) |
| M.Gen.35 | Restrict Resource Allocation per User |
| M.Gen.36 | Limit Protocol Scalability |
| M.Gen.37 | Exchange Sensitive Data via Secure Connections |
| M.Gen.38 | Penetration Testing |
| M.Gen.39 | Continuous Integrity Checks |
| M.Gen.40 | Static Code and Resource Analysis |
| M.Gen.41 | Up-to-Date Session Management |
| M.Gen.42 | Token-Based Authentication for User Requests |
| M.Gen.43 | Utilize Certificate Authorities |
| M.Gen.44 | Encrypt Communications |
| M.Gen.45 | Safe Failing Mechanisms |
| M.Gen.46 | Duplicate Security Mechanisms |
| M.Gen.47 | System Compartmentalization |
| M.Gen.48 | Secure Storage of Credentials |

**Table 5.7:** List of potential mitigation strategies from our Threat Repository [81].

# Chapter 6

# Reference Architecture

In the following chapter we propose our Data Protection Reference Architecture (DPRA) using the data protection requirements from Chapter 4 as well as the threat analysis compiled in Chapter 5. We start by describing the structure of our architecture and how it is supposed to be used during the development process. Afterwards, we present the various components.

## 6.1 Structure

We base the structure of our architecture on our initial data flow diagram (see Fig.2.4). Since we want to provide data protection and security for the entire system while leaving as much flexibility as possible for future developers, it only includes those components that we believe to be essential to Health Data Intelligence Systems in terms of basic functionality and data protection compliance and therefore expect to be incorporated into each realized system in some shape or form.

We consider each element class (i.e. data store, data flow, process, entity) and individual element in turn and define their respective data protection requirements based on our legal requirements and threat analysis. Following that, we present solutions to all issues that can reasonably be addressed on an architectural level.

## 6.2 Requirement Mapping

To find architectural solutions to the various data protection requirements and threats presented in the previous chapters, we first have to consider to which parts of the system they actually apply. To facilitate this, we have created a mapping between the individual components defined in the initial data flow diagram (see Figure 2.4) and the keywords attached to the legal requirements presented in Chapter 4, showing our personal estimation of the relevancy of different requirement categories to each component. This facilitates the determination of necessary protection measures for each individual system part while also helping us to discover similarities that might enable us to find solutions that cover multiple components and/or concerns at once.

Our mapping for our initial model of HDIS is shown in Tables 6.1 and 6.2. A double-plus marks a strong relevancy of a keyword to a specific component, meaning that the requirements repository contains multiple requirements for which compliance has a direct and necessary impact

| | Component | Availability | Confidentiality | Integrity | Data Minimisation | Documentation | Intervenability |
|---|---|---|---|---|---|---|---|
| **Data Store** | Controller Database | + | ++ | ++ | ++ | ++ | ++ |
| | Shared Knowledge Base | + | + | ++ | o | + | o |
| | Processor Database | + | ++ | ++ | ++ | ++ | ++ |
| **Data Flow** | Public Data Ingest | + | o | + | o | + | o |
| | Private Data Ingest | + | ++ | ++ | o | ++ | o |
| | Aggregated Result Transfer | + | + | ++ | o | ++ | o |
| | C2P-Raw Data Transfer | + | ++ | ++ | o | ++ | o |
| | C2P-Historic Knowledge Transfer | + | ++ | + | o | + | o |
| | P2C-Partial Result Transfer | + | ++ | ++ | o | ++ | o |
| | Preprocessing Input | + | ++ | ++ | o | ++ | o |
| | AI- and Mathematical Processing Input | + | + | ++ | o | ++ | o |
| | AI- and Mathematical Processing Output | + | + | ++ | o | ++ | o |
| **Process** | Preprocessing System | + | ++ | ++ | ++ | ++ | o |
| | AI- and Mathematical Processing | + | + | ++ | ++ | ++ | o |
| **Entity** | Public Data Holder | o | o | ++ | o | + | o |
| | Private Data Holder | o | ++ | ++ | + | + | o |
| | Result Recipient | o | + | ++ | o | + | o |
| | Controllerside Operator | + | ++ | ++ | ++ | ++ | ++ |
| | Processorside Operator | + | ++ | ++ | ++ | ++ | ++ |

**Table 6.1:** First part of the relevancy-mapping between the core components of HDIS (see Figure 2.4) and the basic protection goals from our Legal Requirements Repository. Green indicates a strong relevancy, yellow marks a limited relevancy, and grey no relevancy at all.

| | Component | Unlinkability | Distribution | Legality | Risk analysis | Transparency |
|---|---|---|---|---|---|---|
| **Data Store** | Controller Database | ++ | o | o | o | o |
| | Shared Knowledge Base | ++ | o | o | o | o |
| | Processor Database | ++ | o | o | o | o |
| **Data Flow** | Public Data Ingest | o | o | o | o | o |
| | Private Data Ingest | ++ | o | o | o | o |
| | Aggregated Result Transfer | o | o | o | o | o |
| | C2P-Raw Data Transfer | ++ | o | o | o | o |
| | C2P-Historic Knowledge Transfer | ++ | o | o | o | o |
| | P2C-Partial Result Transfer | + | o | o | o | o |
| | Preprocessing Input | ++ | o | o | o | o |
| | AI- and Mathematical Processing Input | ++ | o | o | o | o |
| | AI- and Mathematical Processing Output | + | o | o | o | o |
| **Process** | Preprocessing System | ++ | o | o | o | o |
| | AI- and Mathematical Processing | ++ | o | o | o | o |
| **Entity** | Public Data Holder | o | o | o | o | o |
| | Private Data Holder | ++ | o | + | o | o |
| | Result Recipient | + | o | + | + | + |
| | Controller side Operator | ++ | ++ | ++ | ++ | ++ |
| | Processor side Operator | ++ | ++ | ++ | ++ | ++ |

**Table 6.2:** Second part of the relevancy-mapping between the core components of HDIS (see Figure 2.4) and the basic protection goals from our Legal Requirements Repository. Green indicates a strong relevancy, yellow marks a limited relevancy, and grey no relevancy at all.

on the design of that component. A keyword is partially relevant to a component if compliance to one or more requirements from that category could have an impact on that component but is either dependent on specific circumstances or not legally mandated to be realized in this specific part of the system. Lastly, we denote a keyword as non-relevant for any component, if compliance with a corresponding requirement is neither legally mandated, nor could it be reasonably supported through any measure relating to that component.

### 6.2.1 General Observations

When considering the mapping as a whole, some patterns can be observed that apply to a wider set of components.

While *Availability* is generally applicable to nearly all human and technical system components, we consider it the least essential of the CIA-triad as HDIS are not intended to perform any time-critical functionality. While decision support is an important task, it is not one that requires constant availability of the system, meaning that occasional downtimes don't necessarily impede its overall performance. Conversely, both *Confidentiality* and *Integrity* are essential to the operation of HDIS. The importance of the former is a consequence of the system's heavy reliance on private data, which is handled to some degree by all but two components. Any disclosure of that data would constitute a serious violation of regulation as well as public trust, which needs to be maintained in light of our requirement for a public mandate to legitimize the use of HDIS in the first place. *Integrity* is also key to public trust, but this time not only in context of responsibility towards data subjects, but also with regards to the system's performance quality. All AI-driven reasoning is highly dependent on accurate and correct data to produce models that are beneficial to real world-scenarios. Therefore, data integrity has to be maintained throughout all components with special emphasis on those that handle private data.

The requirements for *Data Minimisation* can be roughly summarized into two main aspects: The first is the limitation of collection and processing of private data to the degree that is absolutely necessary, the second is the deletion of any private data that is no longer needed or whose deletion has been lawfully requested by the data subject. The former affects mostly both operators, the private data holders who ultimately decide what data is collected and processed as well as the processing systems, which need to ensure that only intended and necessary data is used. The later is mostly a concern for all data stores that handle private data as they need to provide the necessary functionalities to both manually and automatically delete all data qualifying for removal, as well as keep track of all metrics that determine this qualification (e.g. storage time).

*Documentation* is the second most used keyword in our repository and the respectively annotated requirements are the most ubiquitous with regards to their applicability to our system components. The GDPR and BDSG both mandate extensive documentation of all processing involving private data including every transfer of such data as well as all implemented security measures and breaches of security. This means that some form of manual or automated documentation and/or monitoring functionality needs to be implemented for every standard component of HDIS.

The notion of *Intervenability* as defined by the regulations guarantees the right for data subjects to request information about their private data, ensure its correctness, as well as demand its deletion or withdrawal from all processing activities. Therefore intervenability requirements mostly concern the HDIS operators on the organizational side and the relevant data bases on the technical side.

Moving on to Table 6.2, our first concept is *Unlinkability*. While there is only a single requirement annotated with this keyword, it has major implications as it mandates the separate processing of private data which has been collected for separate purposes. Since 'processing' as defined by Article 4 (2) GDPR also includes pervasive activities like storage or transfer, the requirement for unlinkability has to be enforced in all components that handle or store private data throughout the entire system.

The last four concepts, i.e. *Distribution*, *Legality*, *Risk Analysis* and *Transparency* stand out for their extremely limited applicability even though the later is by far the most common keyword throughout our requirement repository. This is due to the requirements marked with those keywords being mostly targeted towards organizational entities while having nearly no explicit mandated impact on any technical system components. The requirements annotated with these keywords describe mandated actions and prerequisites that have to be conducted by the controller and operator of a system before initial and during ongoing operation.

## 6.3 Technical Requirements

Using the requirement mapping presented in Tables 6.1 and 6.2, we create a list of technical requirements for each component type which is based on our legal requirements and threat mitigation strategies for general and AI-specific threats. These requirements can be used directly by the developers to design and implement an individual Health Data Intelligence System. Instead of considering all entities as a whole, we only define technical requirements for the external data holders, since the majority of issues regarding the internal operators and result recipients have to be addressed on an organizational and operational level. The complete list can be found in the Technical Requirements Repository [92]. We use these requirements to improve our initial system architecture by considering them in the creation of the DPRA. They also serve as a fallback to prevent any issues that cannot be addressed by our reference architecture from being omitted entirely. Their actualization has to then be considered in later stages of the development process by the respective development teams.

The short example given in Table 6.3 shows how the Technical Requirements Repository is structured. The first column is reserved for a categorical keyword which aligns with those used in our Legal Requirements Repository and is supposed to improve structure and usability of the document. The *Threat Mitigation*-keyword is not featured in the Legal Requirements Repository and has been added to categorize those technical requirements originating from the mitigation strategies compiled in Chapter 5. The two following columns hold an individual ID for each requirement and the textual requirement itself. The last two columns contain references to the legal requirements and threat mitigations upon which every individual technical requirement is based.

| Keyword | ID | Requirement | Legal Requirements | Threat Mitigation |
|---|---|---|---|---|
| **Availability** | Tech.DS.1 | Data Stores shall provide ongoing availability. | Leg.1 | |
| | Tech.DS.2 | Data Stores shall provide ongoing resilience. | Leg.2 | |
| | Tech.DS.3 | In the event of a physical or technical incident, availability of Data Stores shall be restorable in a timely manner. | Leg.2.1 | |
| **Threat Mitigation** | Tech.DS.32 | Data Stores shall only be usable for authorized users. | | M.Gen.1 |
| | Tech.DS.33 | Data Stores shall use multi-factor authentication to authenticate authorized users. | | M.Gen.2 |
| | Tech.DS.34 | Data Stores shall feature appropriate measures to prevent misuse of authentication functionalities. | | M.Gen.3 |
| | Tech.DS.35 | Data Stores shall enforce appropriate password policies for authentication functionalities | | M.Gen.4 |

**Table 6.3:** Sample of the Technical Requirements Repository [92].

## 6.4 Improved Architecture

In this section, we present our Data Protection Reference Architecture, taking into account the technical requirements presented in Section 6.3, which in turn are based upon the legal requirements and mitigation strategies presented in Chapters 4 and 5 respectively. Not all technical requirements can be considered in this architecture as some of them operate on a level of implementation that can not be properly included in a reference architecture without restricting the developers' individual choice for realization options.

Additionally, we want to note once more that this architecture constitutes what we believe to be the minimum level of data protection measures necessary to ensure regulatory compliance within the scope of European Union and German data protection law. Developers are highly encouraged to consider additional measures that might be applicable or even necessary to their specific implementation or application case.

### 6.4.1 System Overview

We start at the top level overview of our architecture. Some basic aspects of the baseline architecture shown in Figure 2.4 remain unchanged, like the two external data holders, the result recipient as well as the general directions of data flows between external entities, controller, and processor. Apart from that, there are a number of relevant changes. First, we add a database management system-component (DBMS) to the controller and processor network that will handle the majority of cross-network data transfers. For both parties, we also move all their databases into respective data store-components and add an additional data store for creating and managing backups of the Controller Data Store. A smaller, but still important change is applied to the networks which are extended from being limited to the controller's and processor's own respective networks to also include external networks that are provided by trusted third parties. This is done in order to remove unnecessary restrictions on the developers and allow for the use of cloud-based deployment under the condition that the cloud provider is able to meet all legally mandated data protection standards. We include both preprocessing and AI-processing into a dedicated component which will also hold all other necessary features for that part of the system. To improve clarity on the top-level of our architecture, we merge all data flows between the same components into a singular named data flow. This will remain an exception however, since we find it important to also visualize the transferred data in our diagrams. Thus all diagrams of lower-level components will feature the transferred data types as annotations to all data flows instead of names.

Considering the changes with regards to their impact on data protection, a detached backup of the controller's data store is the first major adaption made to the original design that we can observe on this level. Creating backups is a basic way of supporting a multitude of important aspects of data protection, e.g. resilience, recoverability and data integrity. While multiple redundant backups provide additional safety, they also increase mirroring time as well as operation costs. The other addition are the two database management systems, which are common as a way to regulate database access by users without administrative rights. This is important for an HDIS to prevent unauthorized access to private data of any sort which might otherwise be abused to manipulate, delete or insert data into any of the data stores. In addition to that, DBMS can also serve as a gateway where incoming data is sanitized and validated with regards to various quality metrics that can be chosen individually by the developers. We'll go into further detail on the DBMS in particular in Sections 6.4.2 and 6.4.4.

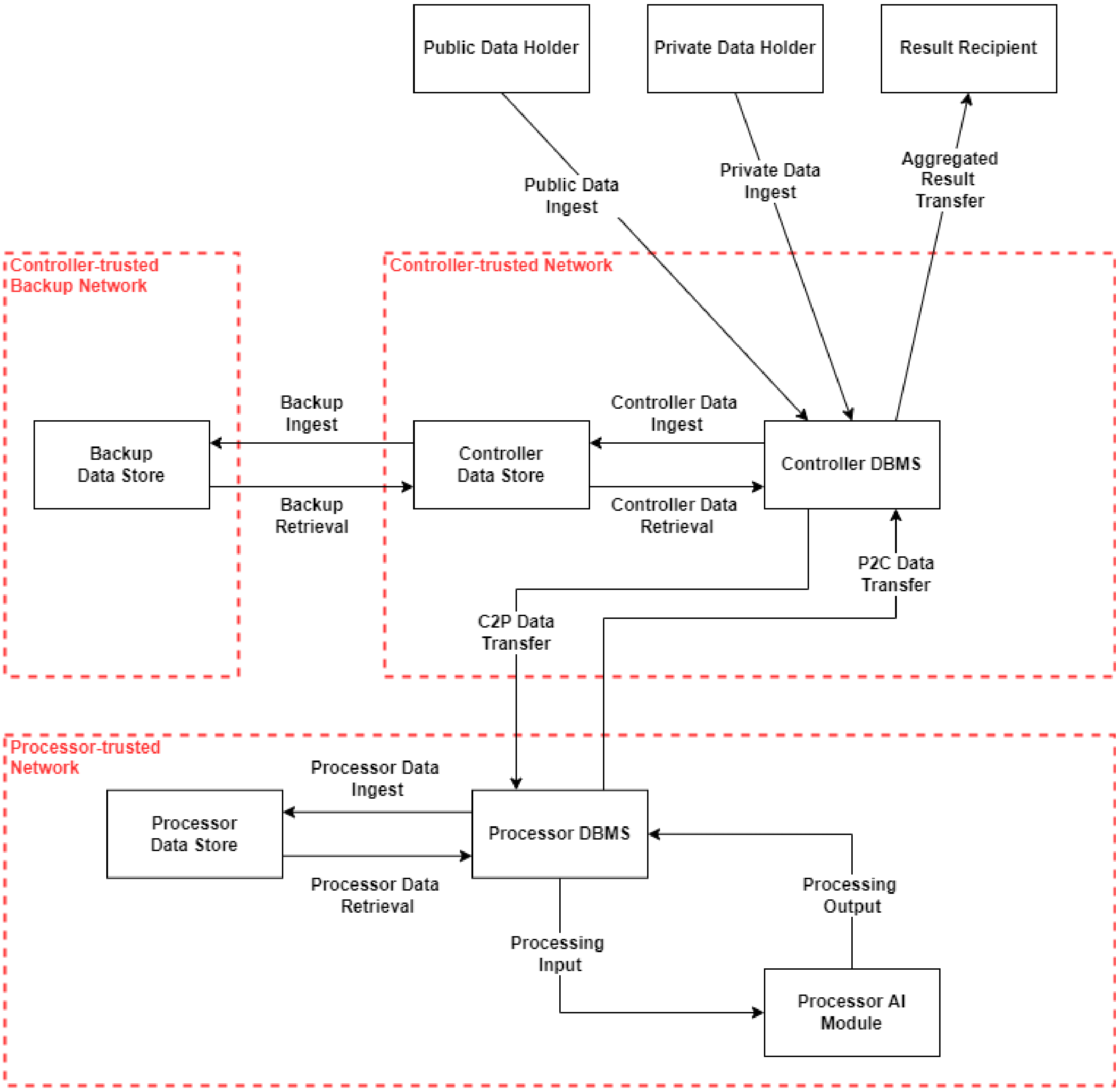


**Figure 6.1:** Data flow diagram of the top level of the improved architecture.

### 6.4.2 Controller DBMS

Beginning our description of the individual components, our first new one is the Controller DBMS, which is depicted in Figure 6.2. It is also probably the most complex due to the large number of functionalities it fulfills. At the top and bottom of the diagram you can see all other external entities from Figure 6.1 which either send data to or receive it from the DBMS. We'll explain each process and other sub-components in order, roughly following the flow of data from external data holders up to the delivery of results to the Result Recipient.

For private data sent via data flows throughout all components of our new architecture, we use three different annotations that provide some contextual information about its status:

- **Annotated (A)**: Certain technical requirements like *Tech.DS.28* and *Tech.DS.29* mandate the annotation of private data with specific information. These annotations are very important, as they include restrictions put on some private data which have to be adhered to by processors by, for example, excluding restricted data entries or sets from AI-processing. Therefore, it has to be ensured that these annotations are preserved when transferring private data, which is symbolized through the A-annotation in the diagrams.

- **Encrypted (E)**: *Tech.DS.7*, *Tech.DF.6*, and a number of further technical requirements mandate the encryption of private data during storage and transfer. While the use of encryption in both scenarios is quite common and might seem obvious, we nonetheless decided to explicitly mark them, since a legal requirement for encryption only exists for private data. Therefore, we wanted to allow developers to consider different standards for implementing those data flows that don't handle private data and could, for example, be operated with a more lightweight communication protocol.

- **Pseudonymized (P)**: Requirement *Tech.DS.9* mandates the pseudonymization of private data during storage. At the same time, *Tech.DS.10* roughly defines circumstances under which pseudonymized data may be re-identified. We use this annotation to mark all stages at which private data has to remain pseudonymized.

The actual processing pipeline begins with the ingest of public and private data from the Public and Private Data Holders, as well as the various processing outputs from the Processors. Private data has to already be encrypted at this stage as it would be illegal to transfer it in readable form. The first point of contact with the HDIS itself is the Data Sanitization & Validation process. It is unfeasible to predefine the exact metrics that incoming data is evaluated by, however *Tech.DS.41* and its subrequirements provide a number of recommendations for consideration.

From here, public data, partial results, and knowledge are sent directly to the Controller Data Store as they require no further processing before storage. Private data however is first transferred to a collection of processes called Controller Data Marshalling (see Figure 6.3). Here, the private data is first decrypted, before receiving any necessary annotations and being pseudonymized. Afterwards, the data is encrypted again and sent to the Controller Data Store as well.

Both the sanitization and validation, and the marshalling component send log data to the Action Log Database. The extensive documentation and logging of all activities involving private data is a key point of the legal requirements and manifests in all technical requirements that fall under the Documentation category. Since there are no legal requirements to the log database itself and under the condition that the log data doesn't contain any private data itself, its implementation is entirely up to the developers.

If any external processor wants to retrieve specific data from the Controller Data Store, they first have to authenticate themselves with the Processor Authentication process. While the exact method of authentication is left open, the threat mitigation requirements resulting from our research into general system security offer some recommendations for a higher level of security, like, for example, the use of multi-factor authentication by *Tech.DS.33* and *Tech.PR.18*. Any authentication attempt is also logged in the Action Log Database. Parallel to the authentication data, the processor can send their query to the Query Manager, which processes them upon receiving a confirmation of authentication by the User Authentication component.

The Query Manager (see Figure 6.4) consists of three different steps. Incoming Queries are first sanitized to prevent the execution of malicious commands or other potential exploits that could be effected through the query interface. Afterwards, the sanitized query is validated against a list of known-good formats and requests. Lastly, if the querying user has been successfully authenticated, the manager executes the query on the Controller Data Store and on success returns the requested data to the processor. To improve general security and be able to detect possible suspicious patterns among user requests, queries are also recorded in the Action Log Database.

The Result Aggregation process is responsible for collecting and organizing all partial results sent by the various processors over a certain amount of time and aggregate them into a reasonable format that can be interpreted and used by the Result Recipient. In Section 5.3.7, we already discussed potential concerns with the presentation of results generated through application of artificial intelligence. With regards to data protection, the most important aspect is to exclude all information from the results that might enable the identification of individuals whose data was used for their generation. Verifying this safe state is the main responsibility of this component.

The last component is called the Operator Interface and serves as a simple stand-in for all user interfaces through which an authorized operator can maintain and interact with the system. Operators might have different levels of access to the other components of the DBMS depending on their individual role within the institution serving as controller. This hierarchy of roles and responsibilities depends on preexisting organizational structures and is difficult to prescribe. However, it is important to regulate access to the system with a robust authentication system preventing administrative access to operators with insufficient authorization.

### 6.4.3 Controller Data Store and Backup

Next, we take a look at the internal design for the Controller Data Store and its dedicated backup (see Figure 6.5). Basically, both data stores consist of three separate databases for shared knowledge and partial results, private data, and public data respectively. In the Controller Data Store, these three get supplied with new data by the Controller DBMS and return it when given a valid query by the Query Manager. The Controller Data Store also contains a process for managing backups which is responsible for scheduling and executing regular backups of data from all three databases to the Backup Data Store. If necessary, the Backup Management process also organizes and executes the retrieval of backup data to restore any lost data on the main data store. The backup is situated within a different network in order to increase security in case the Controller Network is compromised. However, this also means that, depending on the implementation chosen by the developers, backup data has to pass through potentially open channels. Therefore we included a separate Backup DBMS into the Backup Data Store, which is supposed to perform the same functionalities of data sanitization and validation as the Controller DBMS to ensure no manipulation of backup data along the data flow.

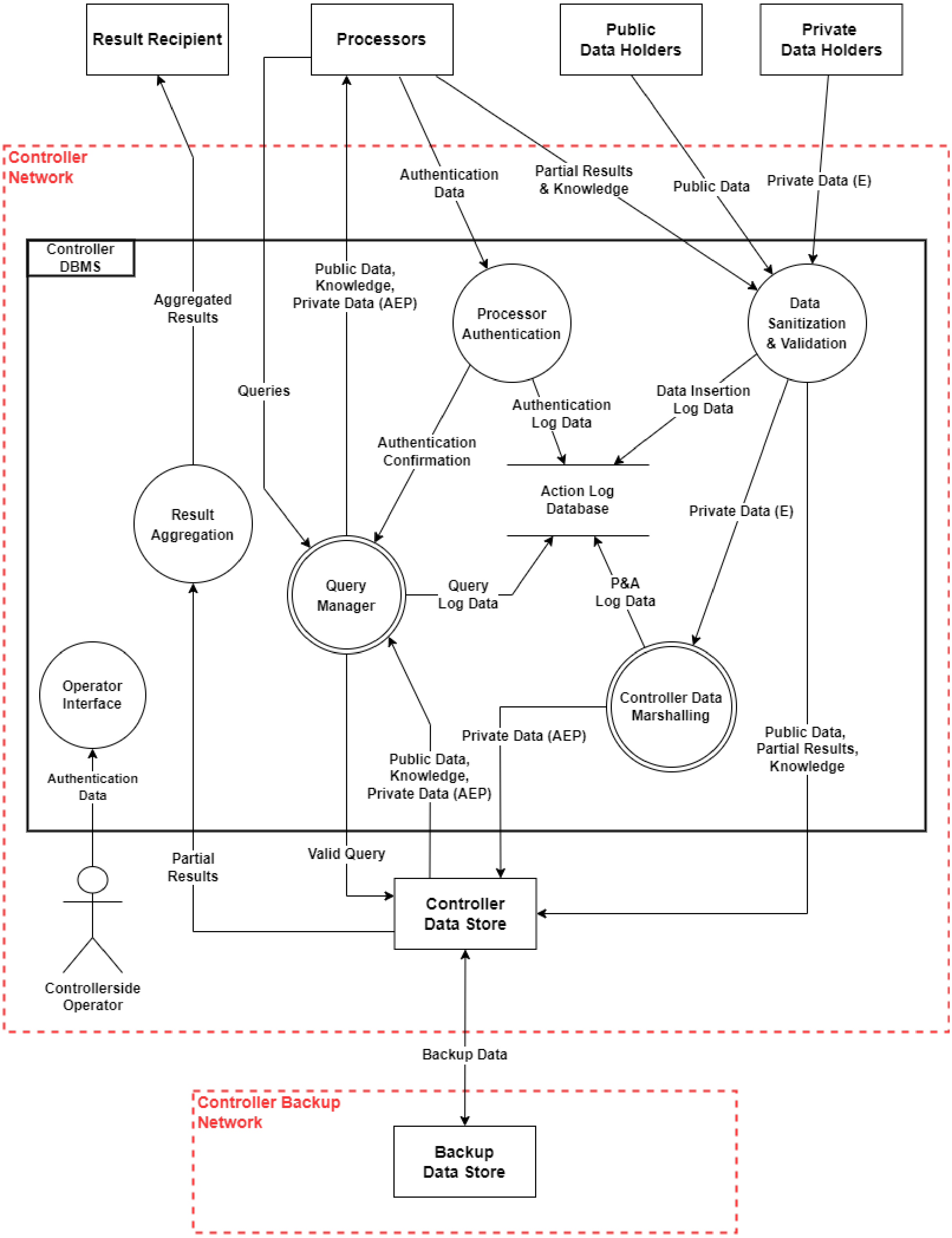


**Figure 6.2:** Data flow diagram of the Controller Database Management System.

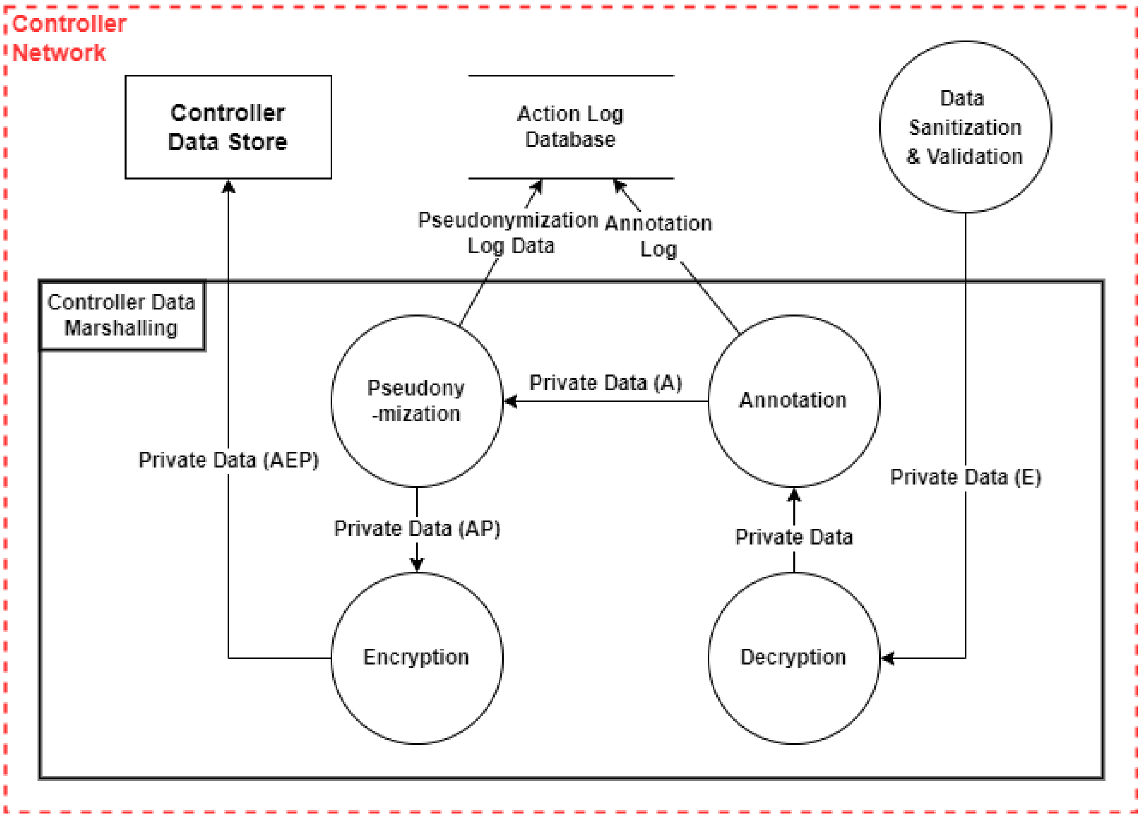


**Figure 6.3:** Data flow diagram of the Data Marshalling-component included within the Controller DBMS (see Figure 6.2).

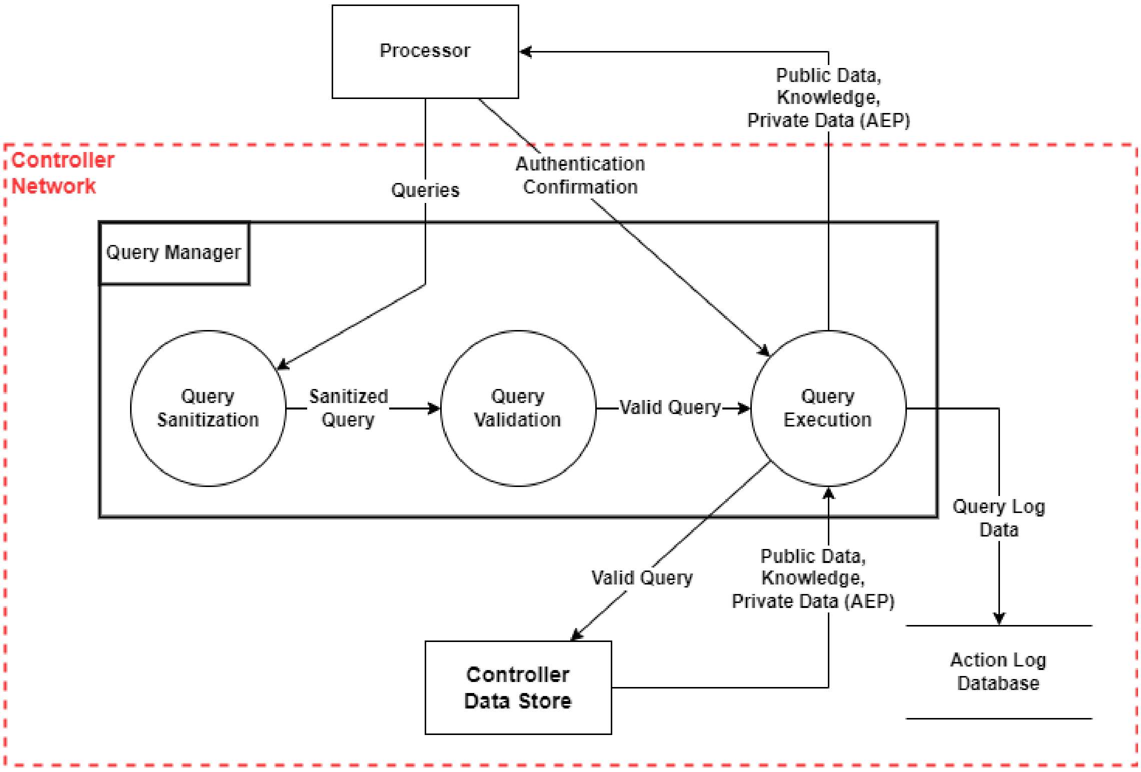


**Figure 6.4:** Data flow diagram of the Query Manager-component included within the Controller DBMS (see Figure 6.2).

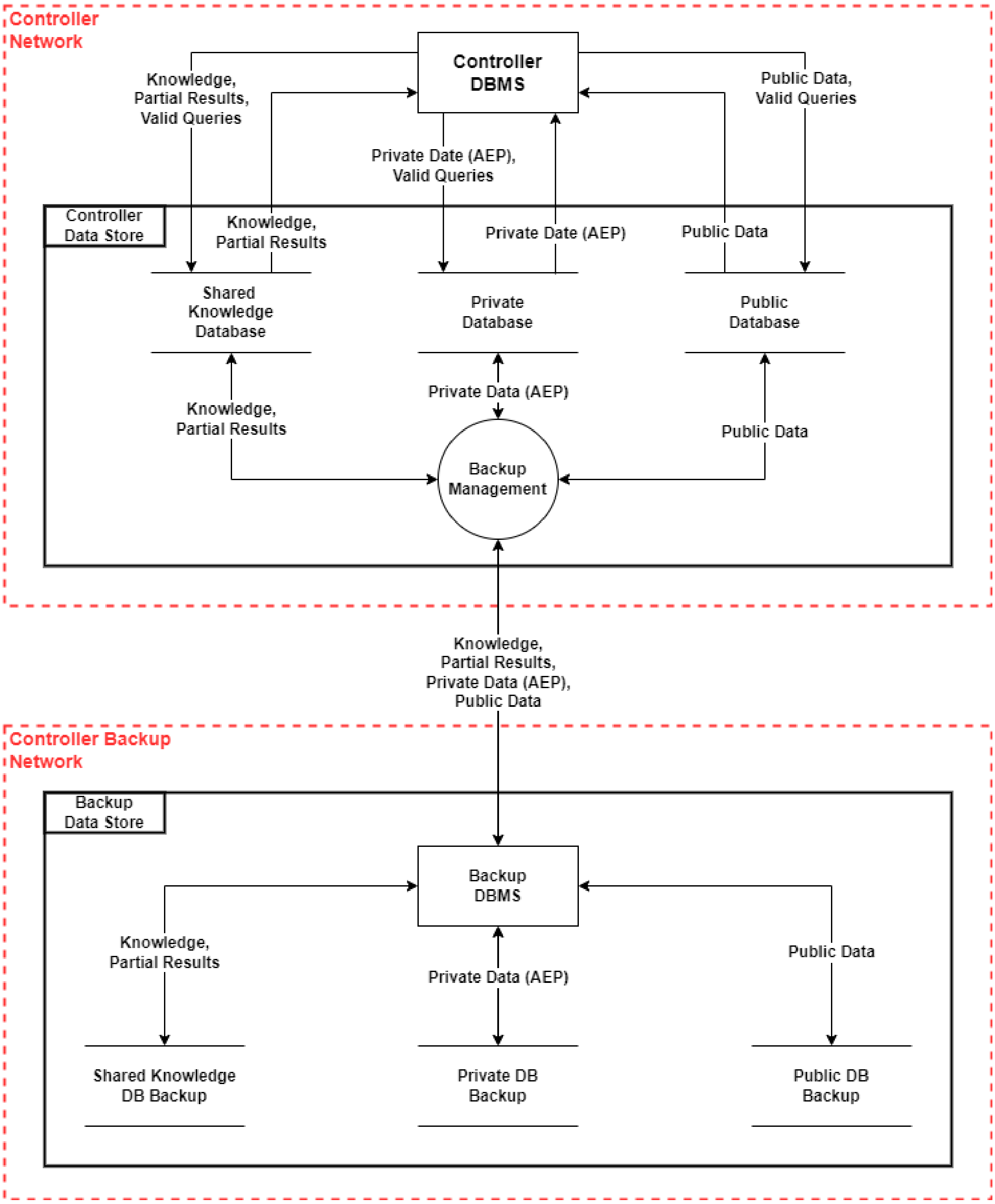

Controller Network
Controller DBMS
Knowledge, Partial Results, Valid Queries
Public Data, Valid Queries
Private Date (AEP), Valid Queries
Controller Data Store
Knowledge, Partial Results
Private Date (AEP)
Public Data
Shared Knowledge Database
Private Database
Public Database
Private Data (AEP)
Knowledge, Partial Results
Public Data
Backup Management
Knowledge, Partial Results, Private Data (AEP), Public Data
Controller Backup Network
Backup Data Store
Backup DBMS
Knowledge, Partial Results
Public Data
Private Data (AEP)
Shared Knowledge DB Backup
Private DB Backup
Public DB Backup

**Figure 6.5:** Data flow diagram of the Controller Data Store-component.

### 6.4.4 Processor DBMS

The Processor DBMS (see Figure 6.6) is built very similarly to the Controller DBMS and contains mostly the same components. Incoming data is again sanitized and validated and then separated into private and non-private data for further processing. Private data is forwarded to another marshalling process, however this time it doesn't need to be decrypted and reencrypted before storage, as this was only necessary for pseudonymizing the data which does not have to be redone by the processor (see Figure 6.7). Instead, the data is only extended with further annotations specific to the processor before being sent to the Processor Data Store. Like the Controller DBMS, the Processor DBMS also records all actions performed on private data in an Action Log Database. Beyond that, this component is also important as a first step when Private Data is retrieved for AI-processing, for which it usually needs to be decrypted and depseudonymized. There are methods to use encrypted data for certain AI-techniques by using homomorphic encryption, however we can't assume this to be possible for all methods and data used in HDIS [55].

The next sub-component of the DBMS is the Query Manager (see Figure 6.8). It is built similarly to its controller-side counterpart, however we forego query sanitization, as all queries are expected to come from the Processor AI Module, which is a trusted source that lies within the same network as the DBMS. Valid queries are again forwarded to the Controller Data Store which directly returns public data and historic knowledge on request. Private data is first sent to the marshalling process to be decrypted and depseudonymized. The collected data is finally returned to the Processor AI Module. As usual, all queries are also logged to the Action Log Database.

Next in line is the Controller Interface, which handles communication with the controller. On the one hand, it is responsible for sending queries for new data as well as the necessary authentication data to the controller. On the other hand, it sends all partial results and new knowledge gained during the AI processing back to the controller to be aggregated and distributed. The last sub-component is the Operator Interface, which functions in the same way as its twin in the Controller DBMS with its main function being the regulation of access to the DBMS by operators with different roles.

### 6.4.5 Processor Data Store

Again, the Processor Data Store functions very similarly to the Controller Data Store. It receives validated queries and newly ingested data from the Processor DBMS and returns it on demand to be distributed to AI processes or sent to the controller. The only outlier here is the ingest of partial results and new knowledge produced by the Processor AI Module which is returned directly from there to the store as it originates from a trusted source within the network and doesn't require any further formatting.

### 6.4.6 Processor AI Module

Lastly, we consider the Processor AI Module, which is responsible for the training of models and the generation of partial results. The pipeline begins with the Processing Manager sending queries for training data to the Processor DBMS which in turn provides it with public data and private data as well as historic knowledge. The training data is then passed on to the Data Validation process where it's quality and format are verified by appropriate metrics to ensure that it is suitable for use in model training. Afterwards, the data is put through the

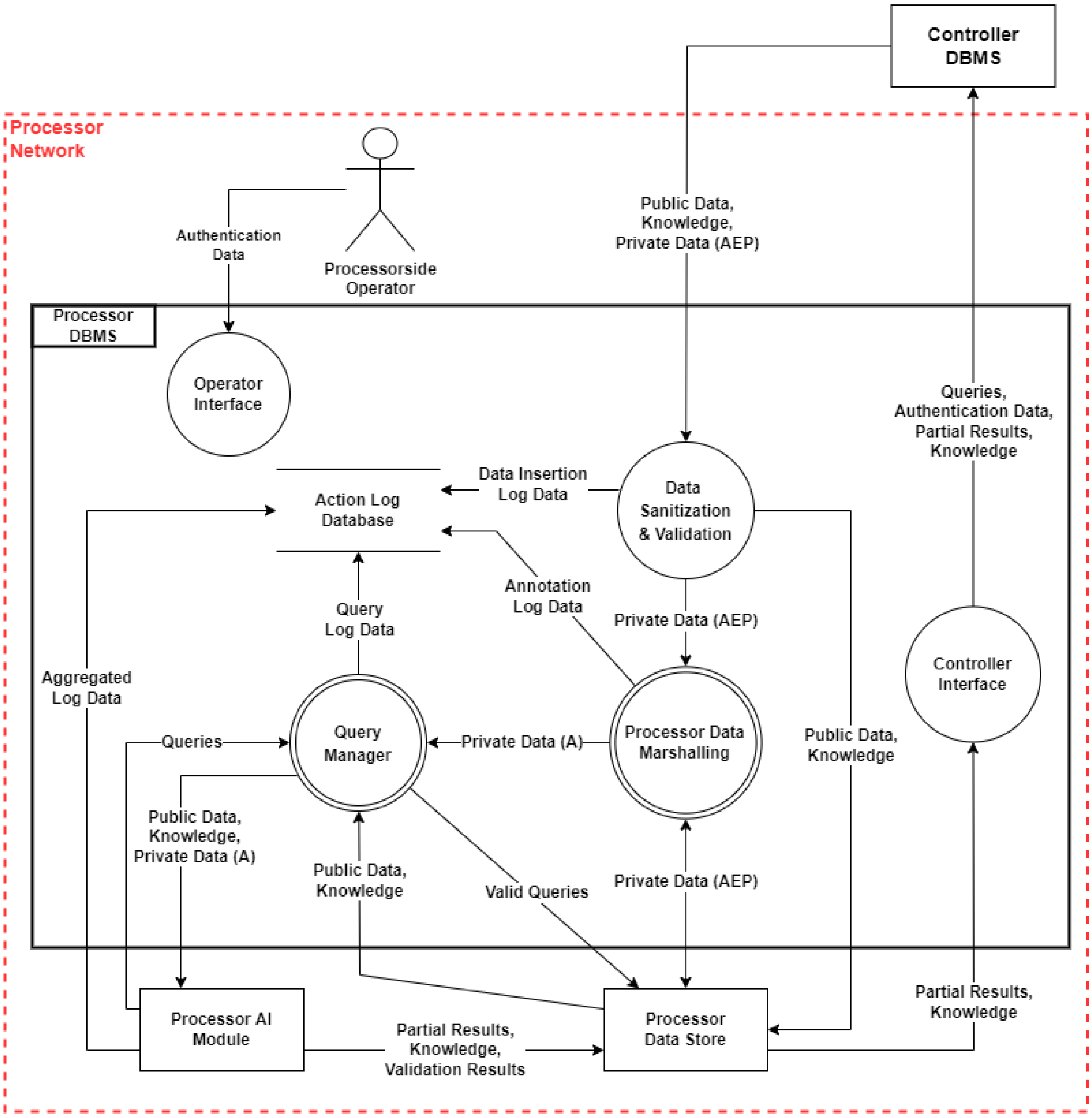


**Figure 6.6:** Data flow diagram of the Processor Database Management System.

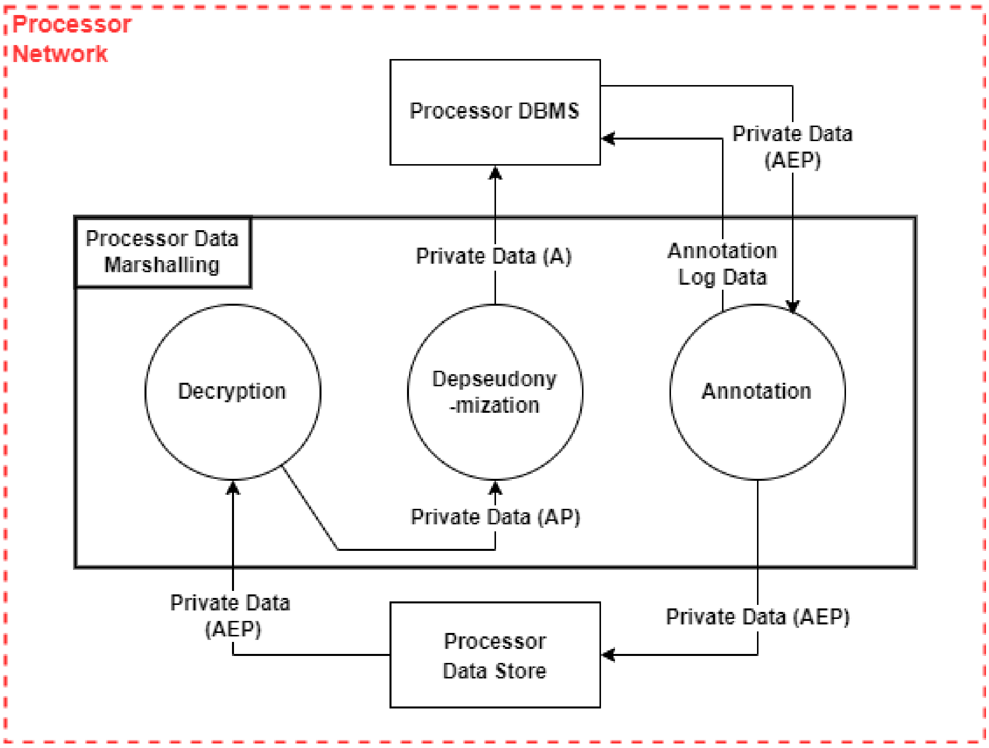


**Figure 6.7:** Data flow diagram of the Processor Data Marshalling-component included within the Processor DBMS (see Figure 6.6).

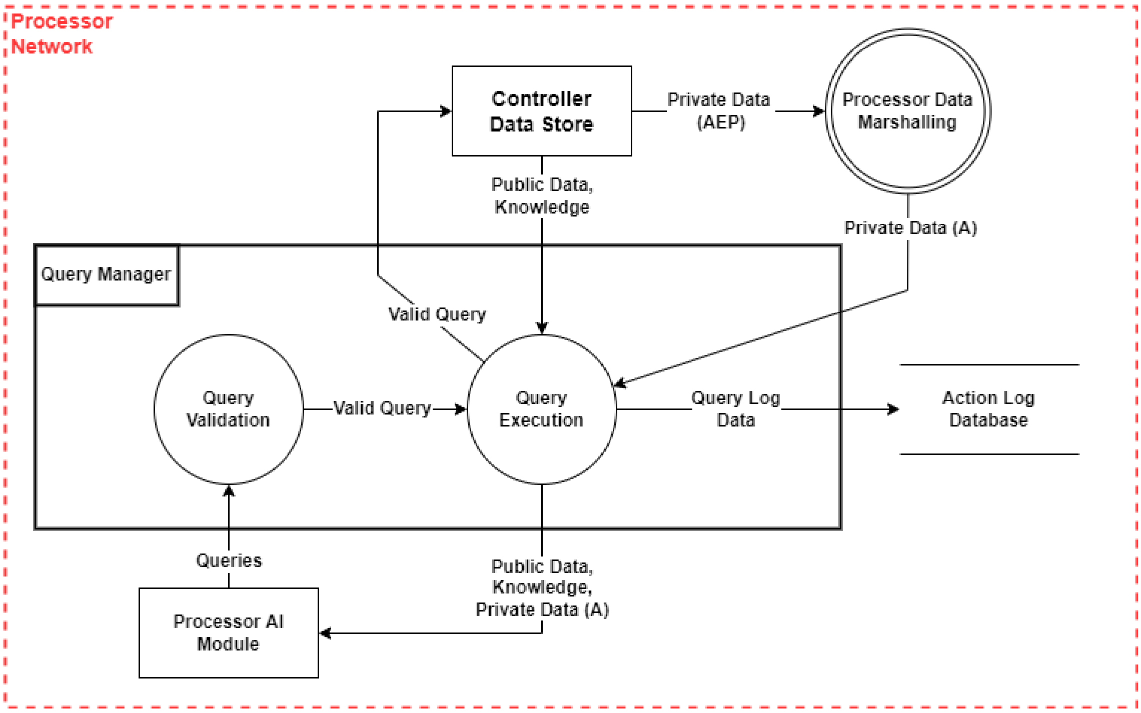


**Figure 6.8:** Data flow diagram of the Processor Query Manager-component included within the Processor DBMS (see Figure 6.6).

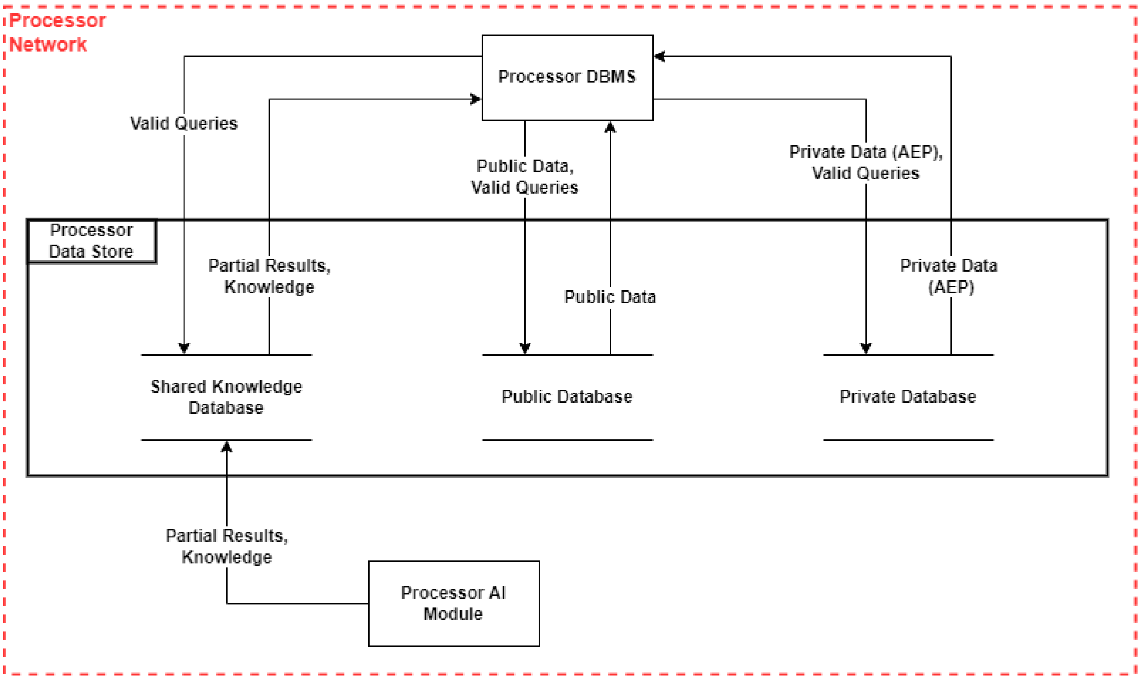


**Figure 6.9:** Data flow diagram of the Processor Data Store-component.

Preprocessing component, where it is transformed into an appropriate representation for use as input data for the AI-Processing. Both preprocessing and AI-processing activities are recorded and sent back to the DBMS and Action Log Database via the Processing Manager. Finally, the results and new knowledge produced by AI-Processing are once more validated in order to assess the impact made by the new data to the existing models. If that impact is deemed to have been positive, the results are passed on to the Processor Data Store. The AI Module also has an Operator Interface as we consider it to be controlled separately from the Processor DBMS.

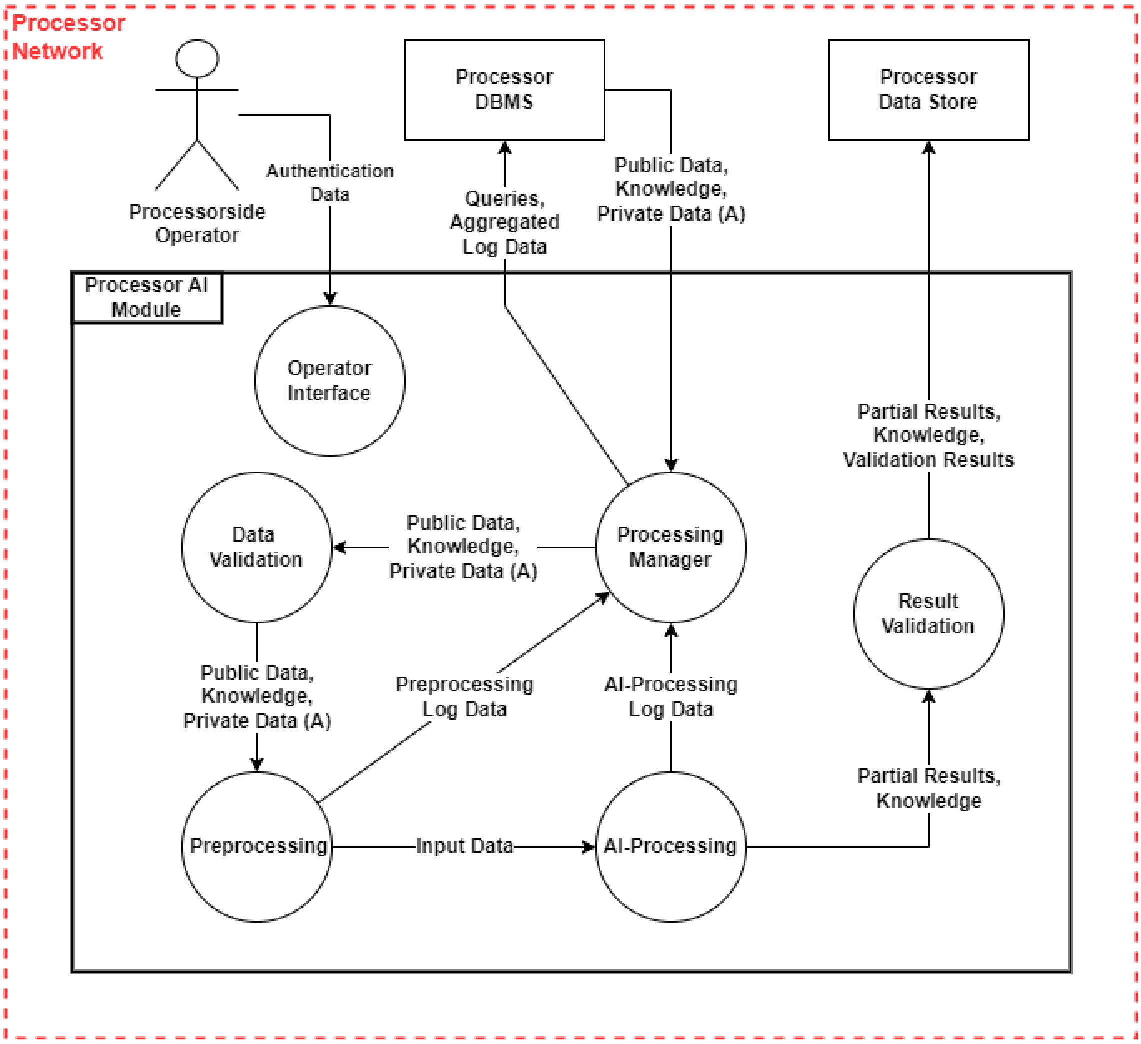


**Figure 6.10:** Data flow diagram of the Processor AI-Module.

## 6.5 Architectural Data Protection

After presenting our improved architecture in Section 6.4, we will now discuss how it supports the realization of data protection requirements and which areas couldn't be covered properly on this level of abstraction. For this, we consider our list of technical requirements for the three main component-types (data stores, data flows and processing systems), and assess to what degree they have been incorporated into the architecture. Tables 6.4 and 6.5 together show a mapping of every component of our Data Protection Reference Architecture to the categories of our technical requirements and to what degree these requirements have been considered in the design of individual components. These tables are supposed to act as an overview while we discuss this relationship in further detail in the following sections.

We chose to forego data flows in this mapping as the most security sensitive data transfers are those crossing network borders, at which point there is usually no way to influence the public communication infrastructure used by the internet. Instead, security of transferred data is reached through the formatting of messages according to certain security protocols which only happens at the communication endpoints. Thus, for example, the requirement for encrypting private data during transfer is not handled by any data flow itself, but rather by a dedicated system component like our Encryption process in the Controller Data Marshalling (see Figure 6.3).

| Component | Sub-Component | Availability | Confidentitality | Integrity | Data Minimization | Documentation | Intervenability | Unlinkability | Threat Mitigation |
|---|---|---|---|---|---|---|---|---|---|
| **Controller DBMS** | Data Sanitization & Validation | o | + | ++ | + | + | o | o | ++ |
| | Processor Authentication | o | ++ | o | o | + | o | o | ++ |
| | Result Aggregation | o | o | o | o | o | o | o | o |
| | Operator Interface | o | ++ | o | ++ | o | ++ | o | ++ |
| | Action Log Database | o | o | + | o | ++ | o | o | ++ |
| **Controller Data Marshalling** | Decryption | o | ++ | o | o | o | o | o | o |
| | Annotation | o | o | o | ++ | + | ++ | ++ | o |
| | Pseudonymization | o | ++ | o | o | + | o | o | o |
| | Encryption | o | ++ | o | o | o | o | o | ++ |
| **Controller Query Manager** | Query Sanitization | o | ++ | + | o | o | o | o | o |
| | Query Validation | o | ++ | + | o | o | o | o | o |
| | Query Execution | o | ++ | o | ++ | + | ++ | ++ | o |
| **Controller Data Store** | Shared Knowledge Database | o | o | + | o | o | o | o | o |
| | Private Database | o | o | + | ++ | o | ++ | o | o |
| | Public Database | o | o | + | o | o | o | o | o |
| | Backup Management | ++ | o | ++ | o | o | o | o | o |
| **Backup Data Store** | Backup DBMS | ++ | o | ++ | + | o | o | o | ++ |
| | Shared Knowledge DB Backup | ++ | o | ++ | o | o | o | o | ++ |
| | Private DB Backup | ++ | o | ++ | ++ | o | o | o | ++ |
| | Public DB Backup | ++ | o | ++ | o | o | o | o | ++ |

**Table 6.4:** Mapping between the controller-side components of our Data Protection Reference Architecture and the main categories of the technical data protection requirements. The colors indicate he degree of consideration of a category in the design of each component. Green indicates a primary consideration, yellow a secondary consideration and grey no particular consideration.

### 6.5.1 Availability

The first three requirements for all three component-types are concerned with availability and resilience. These aspects are mostly reflected in the inclusion of the controller-side Backup Data Store, which ensures that data is not permanently lost in the event of a data loss in the main Controller Data Store. The Backup Management process ensures that backups are created regularly and autonomously and handles the recovery process if necessary. As mentioned in Section

| Component | Sub-Component | Availability | Confidentitality | Integrity | Data Minimization | Documentation | Intervenability | Unlinkability | Threat Mitigation |
|---|---|---|---|---|---|---|---|---|---|
| **Processor DBMS** | Data Sanitization & Validation | o | o | ++ | o | + | o | o | ++ |
| | Controller Interface | o | ++ | o | o | o | o | o | ++ |
| | Operator Interface | o | ++ | o | ++ | o | ++ | o | ++ |
| | Action Log Database | o | o | + | o | ++ | o | o | ++ |
| **Processor Data Marshalling** | Decryption | o | ++ | o | o | o | o | o | o |
| | Depseudonymization | o | ++ | o | o | o | o | o | o |
| | Annotation | o | o | o | + | + | + | ++ | o |
| **Processor Query Manager** | Query Validation | o | ++ | + | o | o | o | o | o |
| | Query Execution | o | ++ | o | ++ | + | ++ | ++ | o |
| **Processor Data Store** | Shared Knowledge Database | o | o | + | o | o | o | o | o |
| | Private Database | o | o | + | ++ | o | ++ | o | o |
| | Public Database | o | o | + | o | o | o | o | o |
| **Processor AI Module** | Processing Manager | o | ++ | o | o | + | + | o | o |
| | Data Validation | o | ++ | ++ | ++ | o | ++ | o | ++ |
| | Preprocessing | o | o | o | o | + | o | o | o |
| | AI-Processing | o | o | o | o | + | o | o | o |
| | Result Validation | o | o | o | o | o | o | o | ++ |
| | Operator Interface | o | o | o | o | o | o | o | o |

**Table 6.5:** Mapping between the controller-side components of our Data Protection Reference Architecture and the main categories of the Technical Data Protection Requirements. The colors indicate he degree of consideration of a category in the design of each component. Green indicates a primary consideration, yellow a secondary consideration and grey no particular consideration.

6.4.3, the backup can also be implemented with multiple redundancies to further improve security against accidental data loss. Our architecture doesn't mandate a similar backup system for the Processor Data Store, as the processors source their entirely from the controller which therefore also acts as a backup for them in case of data loss on their end. However, in practice, processors might produce certain types of data which they don't intend to relay back to the controller and might therefore want to backup locally. In such cases, a backup system similar to the controller-side Backup Data Store should be added and connected to the Processor Data Store.

The non-time critical nature of HDIS means that temporary unavailability of individual data flows is no major issue as long as such events don't lead to any permanent loss of data. The same holds true for our processing systems, whose results might be useful but which will cause no immediate danger in case of a breakdown in operation. Since all data is held in our data stores and only copied from there for transfer or processing reasons, a loss through breakdown of one of these tasks is highly unlikely.

### 6.5.2 Confidentiality

The requirements relating to confidentiality are addressed throughout various parts of the two database management systems. The protection against unauthorized insertion, deletion and manipulation of stored data is handled by each data store's respective Query Manager. On the controller's side, it sanitizes and validates each operation on the data store in order to recognize and prevent illegal operations on the data stores before execution. It further ensures that the querying entity is actually authorized to perform the intended action. On the processor's side, we don't expect any queries from outside the Processor Network and therefore don't require sanitization as mandatory feature. Validation, however, is still important as it also serves as an automatic check for specific annotations on queried data that might restrict its use for AI-processing or deny it entirely. Data stores, data flows and processing systems are protected from unauthorized access through the application of authentication mechanisms in their respective Operator Interface and the processor's Controller Interface, which regulate the interaction of users with the database management systems and the AI-Module. It is up to the institution acting as controller to implement a suitable system of roles which minimizes the number of operators with different levels of authorization to a necessary minimum.

The encryption of private data during storage and transfer is ensured by the general data flow throughout our architecture, which doesn't allow unencrypted private data to reach any of the two principal data stores. Data is only ever decrypted for specific actions like annotation or use in AI-processing. Apart from that, we expect developers to consider secure communication during the implementation of data flows, including end-to-end-encryption of sensitive communication, session management and secure key-generation.

Lastly, the pseudonymization of private data is handled as part of the Controller Data Marshalling immediately after data ingest and ensures that data remains pseudonymized until re-identification is required for the purpose of AI-processing.

### 6.5.3 Integrity

Similarly to availability-related concerns, the protection of data against accidental loss is mostly addressed by the inclusion of database backups into our architecture. The separation of the backup network from the main Controller Network also prevents the backup system from being automatically compromised in case the main network is attacked or experiences internal problems. The main data stores and query managers also contribute to safeguarding data integrity by preventing the execution of unsafe or disruptive queries as well as logging all activity on the database to the respective Action Log Database. This allows retracing of all legitimate actions performed on private data, as well as any illegal actions that weren't able to circumvent the logging mechanisms.

Further integrity requirements mandate accuracy, correctness and up-to-dateness of data during its complete lifecycle. The Controller and Processor DBMS supports this through their respective Data Sanitization & Validation process, which should include suitable mechanisms to ensure integrity of data during transfer. Integrity checks are also part of the Processor AI Module in the form of the Data Validation component and some common metrics have already been suggested as part of the Threat Mitigation category of the Technical Requirements Repository.

### 6.5.4 Data Minimization

Data minimization only concerns data stores and processing systems to varying degrees. The requirement to only store private data that is still necessary for a processing purpose is difficult to consider on an architectural level. However, the Annotation component allows developers to assign various conditional triggers like timestamps to specific entries, which can be used for automatic deletion of data, as well as to restrict or prevent access to data entries with specific annotations. The Query Execution processes in both Query Managers are mainly responsible for automatically catching such annotations within query results and regulate the use of marked data if necessary.

Another requirement is to provide the ability to manually access specific data entries. While these have not been considered as part of the standard operational flow, they are nevertheless included via the Operator Interface, given sufficient authorization of the acting operator.

### 6.5.5 Documentation

The group of requirements dealing with the documentation of specific actions have been considered en bloc through the inclusion of the Action Log Database for the Controller and Processor DBMS respectively. All regular actions performed on the data stores, be they data ingest, annotation, transmission or use in AI-processing are handled by their respective components which in turn log all those activities to said database. The automated deletion of logged actions after a year is part of the implementation and has to be handled by the developers during that phase.

### 6.5.6 Intervenability

This category is closely linked to data minimization and realized through the same mechanisms. The modification of private data in order to correct false information is made possible for authorized operators through the Operator Interface while the prevention of processing of restricted data can be effected by the Query Execution components which can be set to not return restricted entries as part of a query result, as described in section 6.5.4. The Data Validation component of the AI Module can act as a secondary safeguard in that regard as well, depending on the rules programmed during implementation.

### 6.5.7 Unlinkability

The requirement for unlinkability demands the separate handling of private data sets within the HDIS based on their respective purposes. The annotation of data sets with specific processing purposes can be done in the Annotation process of the Data Marshalling component while the Query Manager can prevent the combination and simultaneous querying of differently purposed data sets through suitable validation rules. On the processor's side, it is again up to the Data Validation component of the AI Module to ensure that only data sets with the same purpose-annotation are used together.

### 6.5.8 Threat Mitigation

A common feature among threat mitigation requirements for all three component types is the need for robust authentication mechanisms to regulate their use. We did include authentication as a general notion into our architecture with the Processor Authentication, Operator Interfaces as well as the Controller Interface on the processor's side. However, the specific mechanisms used to this end are a matter of implementation, while the definition of safe authentication policies is mostly an organizational responsibility. For data stores and processing systems, enforcing authentication is comparatively easy as they have a limited number of access interfaces which need to be safeguarded. For data flows, on the other hand, this is more difficult as specifically connections between networks mostly use public physical infrastructure like the internet. This creates the additional challenge for the developers to choose appropriate communication protocols that ensure the security of communications.

Similarly to authentication, physical resilience of components is also a shared concern between all three types. Our technical requirements specifically note the importance of facility maintenance and the protection of those facilities from unauthorized intrusion. These requirements also fall mostly under the umbrella of organizational measures and are not directly included in the architecture.

The issue of data validation and sanitization is represented in multiple requirements, specifically for data stores and processing systems. We include the necessary components for both tasks into our architecture and recommend a number of useful metrics as part of the technical requirements. The selection of specific metrics, however, can not be done properly before the managed data and implemented algorithms have been determined, which is why we cannot go into further detail on this on an architectural level.

The list of technical requirements for data flows features a number of specific entries regarding network security. Those mostly concern usage policies for users interacting with the network like limitations of individual bandwidth or the use of firewalls with black- and whitelisting strategies for incoming traffic. Such defensive mechanisms can only be partially realized by a general firewall, but instead have to be considered for every possible attack-surface, be it general internet traffic from HDIS-adjacent devices, email-traffic or other communication technologies. These surfaces are in constant flux through the operational life-cycle of the system and need to be constantly reevaluated.

# Chapter 7

# Applying the Architecture

In the penultimate chapter, we discuss the artifacts resulting from our previous work and how we intend for them to be used by developers in practice. For this, we consider each artifact and the process by which it is applied before discussing how the application could be executed for an imaginary HDIS-development process. Figure 7.1 shows a flowchart of the first high-level steps of a generic software development process and how the artifacts originating from this thesis can be applied to include data protection by design into the development process. We will use this process flow as a guideline for this chapter.

## 7.1 Artifact Access and Format

The artifacts generated by this project are available on a dedicated GitHub-repository[1] where they can be accessed and freely downloaded. The majority of artifacts comes in the form of Excel workbooks using the *.xlsx*-file format. We chose this file type as it provides good readability through convenient formatting options like URL-masking and table formatting. Although it is a proprietary format owned by Microsoft, it can be read and edited with open source tools like *LibreOffice*[2]. Additionally, there are a number of open source libraries for different programming languages like *Apache POI*[3] or *Pandas*[4] that facilitate ingestion of *.xlsx*-files into other systems and databases. This can be helpful in cases where a team of developers prefers to use custom tools in their design process and wants to import the different repositories into their systems.

All diagrams presented in Section 6.4 are also featured in the GitHub-Repository. They were created using *Draw.io*[5], an open source software for creating a large variety of standard UML and custom diagrams. We provide all diagrams in a combined source file in the *.drawio*- and *.xml*-formats, which can both be viewed and edited using the browser-based or local version of Draw.io. The original versions of the individual diagrams are also provided as *.png*- and *.svg*-files.

[1] https://github.com/DeadPenguin/Engineering-Data-Protection-Aware-Health-Data-Intelligence-Systems

[2] https://www.libreoffice.org/

[3] https://poi.apache.org/

[4] https://pandas.pydata.org/

[5] https://www.draw.io

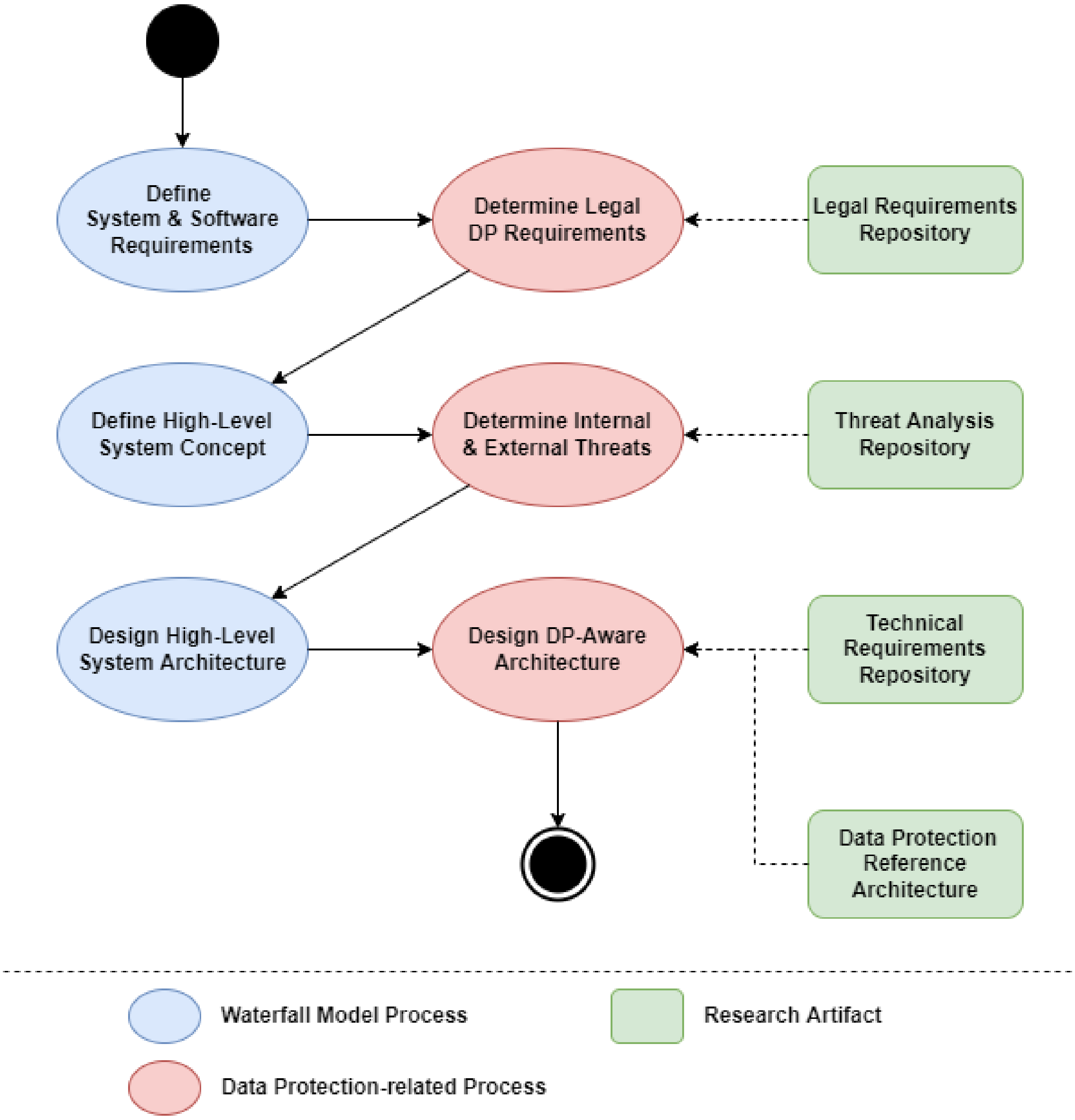


**Figure 7.1:** Flowchart showing an exemplary way of using our research artifacts to include data protection by design into the design phases of a generic software development process based on the waterfall model.

## 7.2 Determining Legal Requirements

The first step of development is the assignment of the legal roles filled by each stakeholder as they will determine the individual obligations of each party involved (see Figure 7.2). Next, the development team needs to verify the content of the repository by comparing the dates in the version in the *Sources*-sheet of the repository to the current versions of the GDPR and BDSG at the time of development. This is necessary to ensure that the requirements are up-to-date. If a newer version of any regulation exists, the development team then needs to match the changelog with the existing requirements to identify any substantial differences. This can be done by searching the *Legal Requirements*-sheet for any article ID that appears in the changelog and then verify if the corresponding requirement still aligns with the changed article. If not, the requirement should be updated accordingly. After finishing this process, the regulation's version in the *Sources*-tab can be updated to the date of the current revision.

Afterwards, all involved parties can use the Legal Requirements Repository to identify those requirements that apply to them specifically. Since the project is based within German jurisdiction, both the GDPR and the BDSG apply to all stakeholders. However, as mentioned in Chapter 4, these requirements are likely to be incomplete, as each stakeholder may be subject to additional regulations based on their federal state and the data sources they are planning to use. Therefore it is up to all involved parties to identify those additional regulations, match them with our existing list and add any missing requirements.

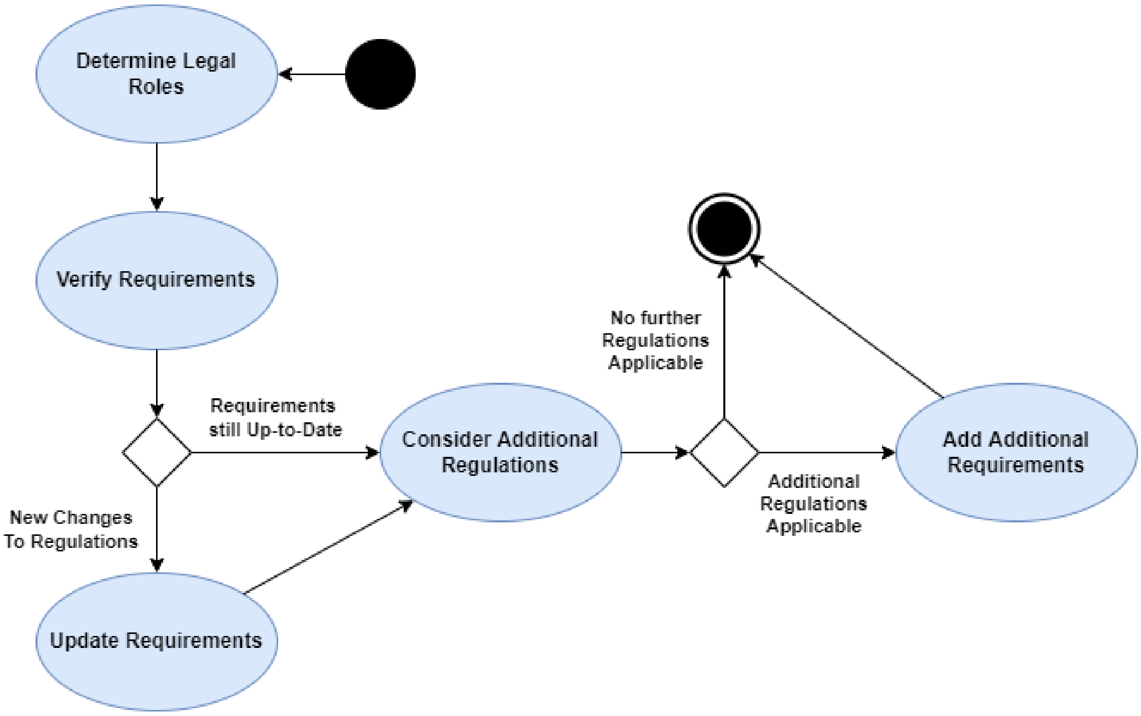

Determine Legal Roles
Verify Requirements
Requirements still Up-to-Date
New Changes To Regulations
Update Requirements
Consider Additional Regulations
No further Regulations Applicable
Additional Regulations Applicable
Add Additional Requirements

**Figure 7.2:** Flowchart showing the process of applying the Legal Requirements Repository to a HDIS-development process.

## 7.3 Identifying Threats

The second step in the application of our data protection architecture consists of identifying the external and internal security threats to the HDIS in general and its individual components in particular. For this, we assume that the development team already created a high-level concept of the system which allows them to realistically assess the relevance of individual threats. However, application of the Threat Repository doesn't require a specific set of design documents as we intend it to be usable in two ways: On one hand, it can be applied in a similar way to the Legal Requirements Repository where the developers identify components in their current design that are susceptible to individual threats. On the other hand, it can also serve as a support for future design decisions by creating awareness for threats that might be avoided completely by making or avoiding certain design decisions later on.

The process itself is more straight forward than determining the legal requirements (see Figure 7.3). First, the development team considers the three threat-categories covered by our repository and identifies any weaknesses, attack techniques and AI-related threats that are applicable to their system design. If a threat is only relevant for later stages of the system lifecycle like implementation or operation and can therefore not be comprehensively assessed at this point in time, it should be marked and forwarded to the individual teams responsible for those stages to be considered at that point. Afterwards, it is again advised to verify the compiled entries using the provided sources, all of which are freely available, either as browser-based web services or as PDF-documents. The most important aspects to consider here are the up-to-dateness of source URLs and possible obsolescence of threats due to technological advances. For sources that rely on a regular publishing schedule like the TLS or the GSK, it should also be checked if an update has been released since the last use of the repository. If the update is intended to replace the previous release, the references in the repository should be updated as well. If it is instead meant as a supplement or extension, the document should be analyzed for relevant additions to the Threat Repository.

For AI-related threats, the verification process is somewhat more difficult as they rarely have a singular source, but have instead been compiled by analyzing various scientific research papers for which there is neither a specific update schedule nor a clear indicator of obsolescence. Therefore, we have to rely on the initiative and professional expertise of the departments responsible for designing and implementing the AI-related components of the HDIS to consider the sources used in Section 5.3 and update the list of threats based on the current general state of knowledge in the scientific community. After verifying the threats suggested by the repository, the developers should then research further threats that are specific to their vision of an HDIS and might therefore not be covered by the original Threat Repository. Those additional threats, as well as their sources should be added to the repository as new entries to the respective sheets.

Next, the developers should use the references to specific mitigation strategies on the *General Mitigation Strategies*- and *AI Mitigation Strategies*-sheets in each threat entry to identify recommended ways of addressing them. Again, those strategies apply on different stages of the system lifecycle and should be forwarded to the appropriate stages of the development process if they can't be implemented immediately. Afterwards, the mitigation strategies should be verified as well, which can be done in the same structured way as described above for general weaknesses and attack techniques. For AI-related mitigation strategies, the problem remains the same as with AI-related threats, which is why we once again have to defer to the sources discussed in Section 5.3.7 and the responsible department's collective expertise. The development team should again research further mitigation strategies that might help alleviate the risks posed by their additional threats and add them to the list of general and AI-specific mitigation strategies. For the sake of prioritizing individual approaches for implementation, it might also be helpful to consider if those new mitigations can also help with any of the original recommended threats. A list of relevant and feasible mitigation strategies is compiled and distributed among developers to enable team members at each stage of the development process to consider and decide on the most promising approaches to include into the design and implement them into the finished system.

Since mitigation strategies are also an important source for technical requirements, the development team should consider their own additional mitigation strategies and if they could be reasonably applied to any component type. If so, they can then devise new requirements to implement those strategies and add them to the Technical Requirements Repository under the *Threat Mitigation*-keyword.

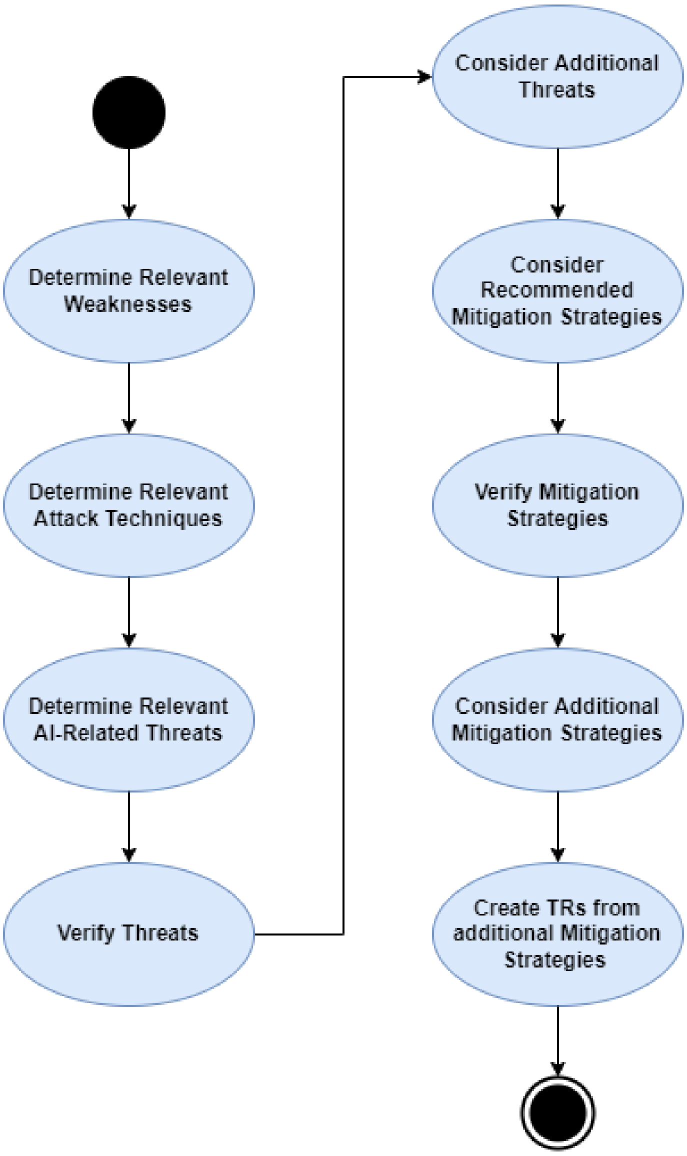

Determine Relevant Weaknesses
Determine Relevant Attack Techniques
Determine Relevant AI-Related Threats
Verify Threats
Consider Additional Threats
Consider Recommended Mitigation Strategies
Verify Mitigation Strategies
Consider Additional Mitigation Strategies
Create TRs from additional Mitigation Strategies

**Figure 7.3:** Flowchart showing the process of conducting a Threat Analysis for a HDIS-development process using the Threat Repository.

## 7.4 Applying the DPRA

The last two remaining artifacts are the Technical Requirements Repository, which holds the requirements that have to be fulfilled by different types of components, and the collection of DFDs showing each individual component, the data flows between them and their internal processes. Both artifacts, combined with the description of the improved architecture in Chapter 6.4 form the actual Data Protection Reference Architecture.

To apply the reference architecture, the development team first needs to define and document a general high-level system architecture of their HDIS, containing the components and processes that are essential for the system to operate, as well as the planned network structure for distributed systems. This also constitutes the first step of our application process outlined in Figure 7.4. Once this initial architecture is complete, the team then needs to map its components to the DPRA, using the description of each individual component provided in Section 6.4 to compare intended and required functionalities. Following the mapping process, they can then determine the technical requirements for each of their components based on the standard requirements provided by the Technical Requirements Repository.

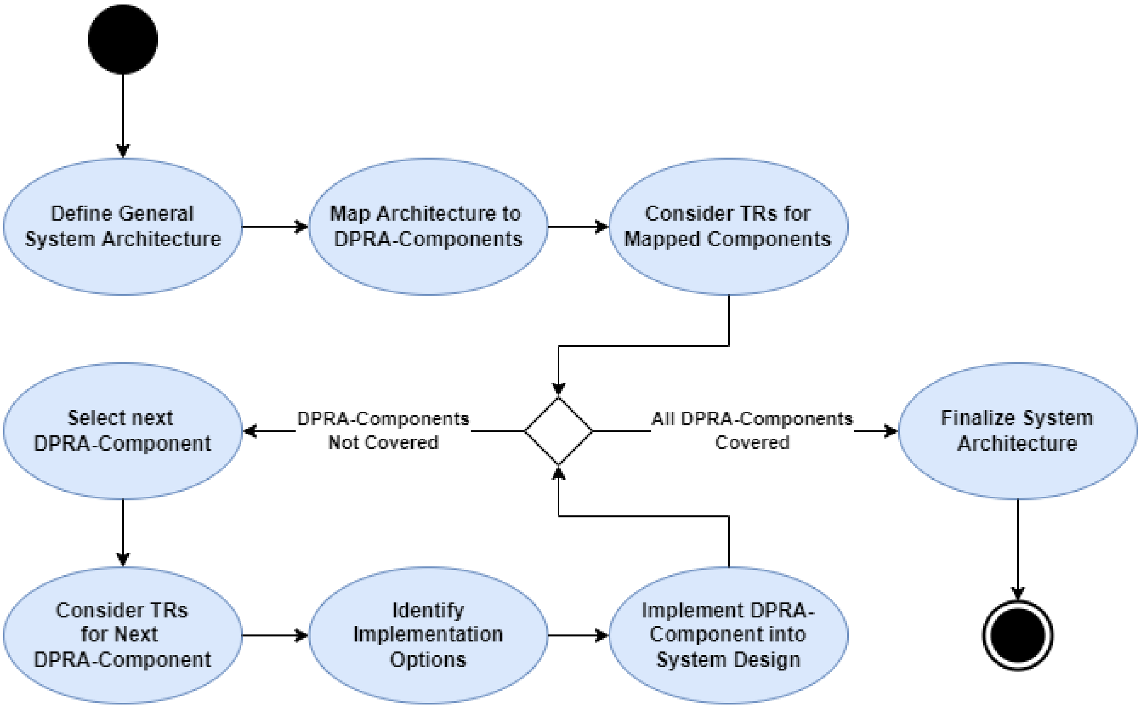

Define General System Architecture
Map Architecture to DPRA-Components
Consider TRs for Mapped Components
Select next DPRA-Component
DPRA-Components Not Covered
All DPRA-Components Covered
Finalize System Architecture
Consider TRs for Next DPRA-Component
Identify Implementation Options
Implement DPRA-Component into System Design

**Figure 7.4:** Flowchart showing the process of integrating the Data Protection Reference Architecture (DPRA) into a general system architecture.

If the mapping does not cover all components and functionalities of the DPRA, the team then should select the next uncovered DPRA-component and consider its description and what requirements from the Technical Requirements Repository apply to it. From this, they can determine the missing functionalities that still have to be included into their design and begin researching possible architectural and technological options to include them into their existing design. If a feasible solution is found, it is then incorporated into the HDIS' architecture. This process is repeated until all components from DPRA have been considered at which point the system architecture can then be finalized before being moved to the implementation stage.

It is important to note once more that the final mapping between the DPRA and the actual system architecture does not necessarily have to be a one-to-one alignment in terms of components. The DPRA is intended to provide a high-level guideline with the focus being placed mainly on the required functionalities of the components and the data flows between them. If, for example, the development team decides to add additional security measures to the Controller Data Marshalling-component (see Figure 6.3) by placing them between the Pseudonymization- and Encryption-processes, this is highly encouraged as long as the requirements for the existing processes and data flows are still met. The same holds true if the developers decide to merge multiple sub-components, for instance because they intend to employ a commercial tool that combines the functionalities of two components which are separate in the original DPRA.

## 7.5 Exemplary Development Process

Lastly, we want to show how the processes described in this chapter could be applied in the context of a realistic HDIS development project. For this, we first establish our premise under which the project will be conducted before going over each step of the previously described application process and describing the actions taken by the hypothetical development team in order to implement it.

### 7.5.1 Premise

The German Federal Ministry of Health funds the development and operation of a central, federal Health Data Intelligence System with the goal of obtaining accurate prediction models on the development and spread of future epidemic events, as well as the projected effects of different containment measures. The system shall be managed by a new department of the federal Robert Koch Institute, which shall be responsible for coordination during development, operational management, as well as data collection and distribution. Several independent research groups throughout the country have been approached and over the next two years a number of AI- and simulation-based models are proposed and tested on historic data. After a thorough review-process, three models are approved by the RKI and the Federal Ministry of Health, which then greenlights the development process.

To utilize existing expertise, it is decided that the individual models shall be operated and improved by the original research groups at Koblenz University, TU Berlin and RWTH Aachen, which are provided funding in order to establish a stable team of developers and acquire the necessary processing capacities. A central group comprising members of all stakeholder institutions is formed in order to coordinate development between individual groups. That group is also responsible for harmonizing security across all planned subsystems.

### 7.5.2 Legal Requirements

To identify the relevant legal requirements, the project team first has to assign the required legal roles to suitable parties among the stakeholders. The Federal Ministry of Health assumes the role of Supervisory Authority as it is the initiator of the program and natural surveillance authority of the RKI. It shares this role with the Federal Commissioner for Data Protection and Freedom of Information (dt. BfDI) who by default is responsible for ensuring compliance with privacy and data protection regulation among all German governmental institutions, as required by Chapter 4 of the BDSG [1]. Additionally, the federal states in which the various stakeholders are situated also have individual offices for data protection to which the respective stakeholders are accountable. The RKI itself will act as the Data Controller while the three research groups take on the roles of processors with regards to data protection law.

After those roles have been determined and documented, the collective development team can begin analyzing the legal requirements. During the verification process, they find that the GDPR has received a small amendment the previous year which has not yet been incorporated into the requirements repository. They identify the changed articles and find four requirements referring to said articles. While two of them are not affected by the changes, the other two no longer align with the articles' statements and are therefore updated as well. Afterwards the *Version*-field of the GDPR-entry on the *Sources*-sheet is updated with the date of the amendment. Following the verification, the team continues by researching additional regulations that might apply to individual parties only. Since all of them are situated in different federal states, they each have to consider their respective state's data protection law. Each stakeholder subsequently analyzes their respective state law, generating additional requirements where necessary or adding references to the repository if an existing requirement is implied by the new laws. They also mark each requirement, that has an effect on the other stakeholders in order to harmonize requirements and ensure legality of their processing activities across federal state borders.

### 7.5.3 Threat Analysis

For the threat analysis process, we assume that the joined development team has by now established a basic outline of what processes and functionalities should be included, for example through a comprehensive list of business and user requirements. Using this outline, they now consider the *Weaknesses*- and *Attack Techniques*-sheets of the Threat Repository and check the applicability of each individual threat to the desired components and processes. As an example, we consider *W.12* which concerns the unrestricted upload of dangerous files. The research group from Koblenz University marks this weakness as relevant, as they intend to cooperate with the Ministry of Science and Health Rheinland-Pfalz with the intention of acquiring data from them through a push-based system. After identifying this weakness as relevant, the team verifies its up-to-dateness by the URL leading to *CWE-434*, which served as the primary source for this entry. They find no significant changes and therefore continue with analyzing further threats until every entry has been considered. Following the suggested threats from the repository, the developers then continue by conducting further, more precise research on specialized aspects of their system that are not yet covered by the Threat Repository and add those threats and their respective sources to the *Weaknesses*- and *Attack Techniques*-sheets.

Afterwards, the team begins compiling the potential mitigation strategies recommended by the repository. For *W.12* specifically, the repository suggests five different mitigation strategies for consideration, those being *M.Gen.15*, *M.Gen.30*, *M.Gen.32*, *M.Gen.33* and *M.Gen.46*. The development team then once more verifies all five entries against their respective sources, before continuing with researching additional mitigation strategies for the threats added by them in previous steps. Following the compilation of mitigation strategies, the team sorts them by the number of individual threats that are addressed by them. While this is not a definitive metric of the impact of each solution on the overall system security, it can still serve as an indicator as to which mitigations should be prioritized during further development. So if the metric shows that *M.Gen.15* and *M.Gen.30* are considered possible solutions to the highest number of individual threats in our exemplary case, this is pointed out to the rest of developers by their ranking in the sorted list. The final decision about which mitigations are integrated, however, is then made in a concerted effort with the team members who possess the highest expertise in their respective application contexts, e.g. network or database development. Lastly, they consider the selected strategies with regards to how and at which level they can be implemented and create new technical requirements based on their conclusions. The requirements are then added to the appropriate sheets of the Technical Requirements Repository.

### 7.5.4 Application of DPRA

After finishing the threat assessment process, the team turns towards designing their system architecture. As a first step, they include all necessary components and processes to realize the core functionalities of the HDIS, resulting in a number of models on different levels of granularity, of which we consider the top-level system architecture as an example (see Figure 7.5). In order to ensure that their system offers a sufficient degree of data protection and security, the team now proceeds with mapping their architecture to the corresponding level of the DPRA which, in our example, would be the top-level depicted in Figure 6.1. Table 7.1 shows how each component of the initial system architecture maps to one of the standardized components of the DPRA. Since there are multiple processors and private data holders, those standard components are covered more than once. At the same time, individual elements of the system architecture cover multiple roles of the DPRA, like the *RKI Data Exchange*, which can act as both the *Controller Data Ingest* as well as the *Controller Data Retrieval*. This mapping now allows the team to consult the Technical Requirements Repository to find the concrete data protection requirements for each component. For each requirement, the developers can then decide if it is applicable for this specific component, is already covered by the current architecture, or will be covered in a later stage of the design process.

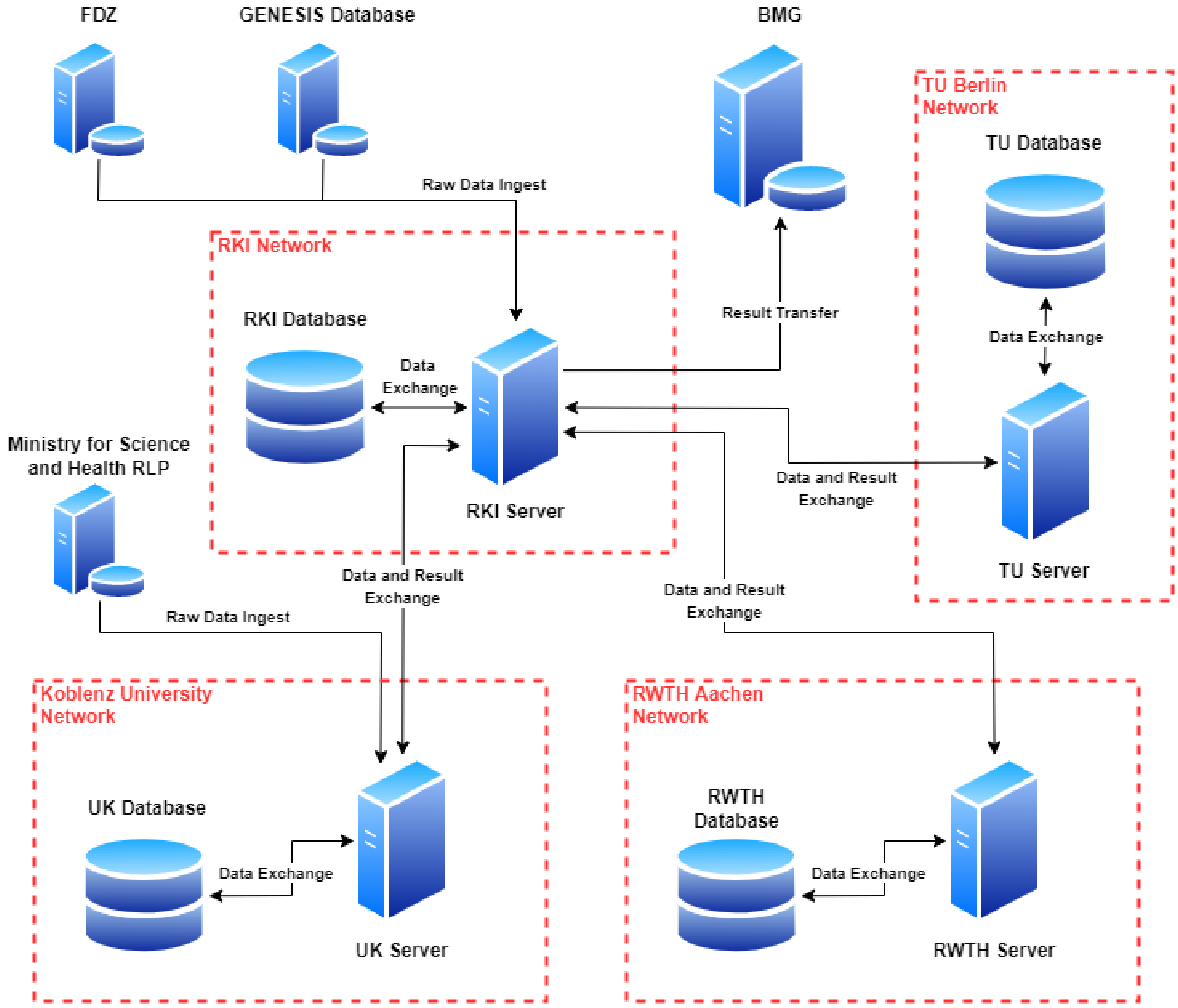


**Figure 7.5:** Top-level diagram of the planned HDIS before the application of the DPRA.

| DPRA Component | System Architecture Component |
|---|---|
| Public Data Holder | GENESIS Database |
| Private Data Holder | FDZ, Ministry for Science and Health RLP |
| Result Recipient | BMG |
| Public Data Ingest | Raw Data Ingest |
| Private Data Ingest | Raw Data Ingest |
| Aggregated Result Transfer | Result Transfer |
| Controller-trusted Network | RKI Network |
| Controller DBMS | RKI Server |
| Controller Data Store | RKI Database |
| Controller Data Ingest | RKI Data Exchange |
| Controller Data Retrieval | RKI Data Exchange |
| Controller-trusted Backup Network | - |
| Backup Data Store | - |
| Backup Ingest | - |
| Backup Retrieval | - |
| Processor-trusted Network | Koblenz University Network, RWTH Aachen Network, TU Berlin Network |
| Processor DBMS | UK Server, RWTH Server, TU Server |
| Processor AI Module | UK Server, RWTH Server, TU Server |
| Processor Data Store | UK Database, RWTH Database, TU Database |
| Processor Data Ingest | UK Data Exchange, RWTH Data Exchange, TU Data Exchange |
| Processor Data Retrieval | UK Data Exchange, RWTH Data Exchange, TU Data Exchange |

**Table 7.1:** Mapping between the components of the original system architecture (see Figure 7.5) and the corresponding DPRA diagram (see Figure 6.1).

As an example, we consider the *RKI Server* which has been mapped to the role of *Controller DBMS*. The DBMS counts as a processing system and therefore needs to fulfill *Tech.PR.4* which states that processing systems shall prevent unauthorized processing of private data. Since data transfer is considered a form of processing under the GDPR and the *RKI Server* acts as a data hub for the three university research groups, this means that it needs to prevent unauthorized distribution of data to any of those external processors. We assume that this requirement has not yet been addressed on a lower level of the architecture either and therefore is still completely missing from the design. The team now considers possible options to include *Tech.PR.4* into the architecture and on which level it should be done. At the same time, it is apparent that an entire group of standard components forming the controller data backup system are missing from the top-level architecture entirely. While the implementation of a simple backup database could be delegated to the lower-level architecture of the *RKI Database*-component, the DPRA also requires the backup to be situated within a separate network which is not possible under the current top-level design. Lastly, the *Ministry for Science and Health RLP* is categorized as a Private Data Holder. *Tech.DH.9* requires data holders to only provide private data to controllers or processors if it is required for one or more legitimate processing purposes. Since those purposes are determined by the controller and could change with time, all private data should be ingested by the controller first to avoid processors accidentally retaining or processing data that is not required anymore.

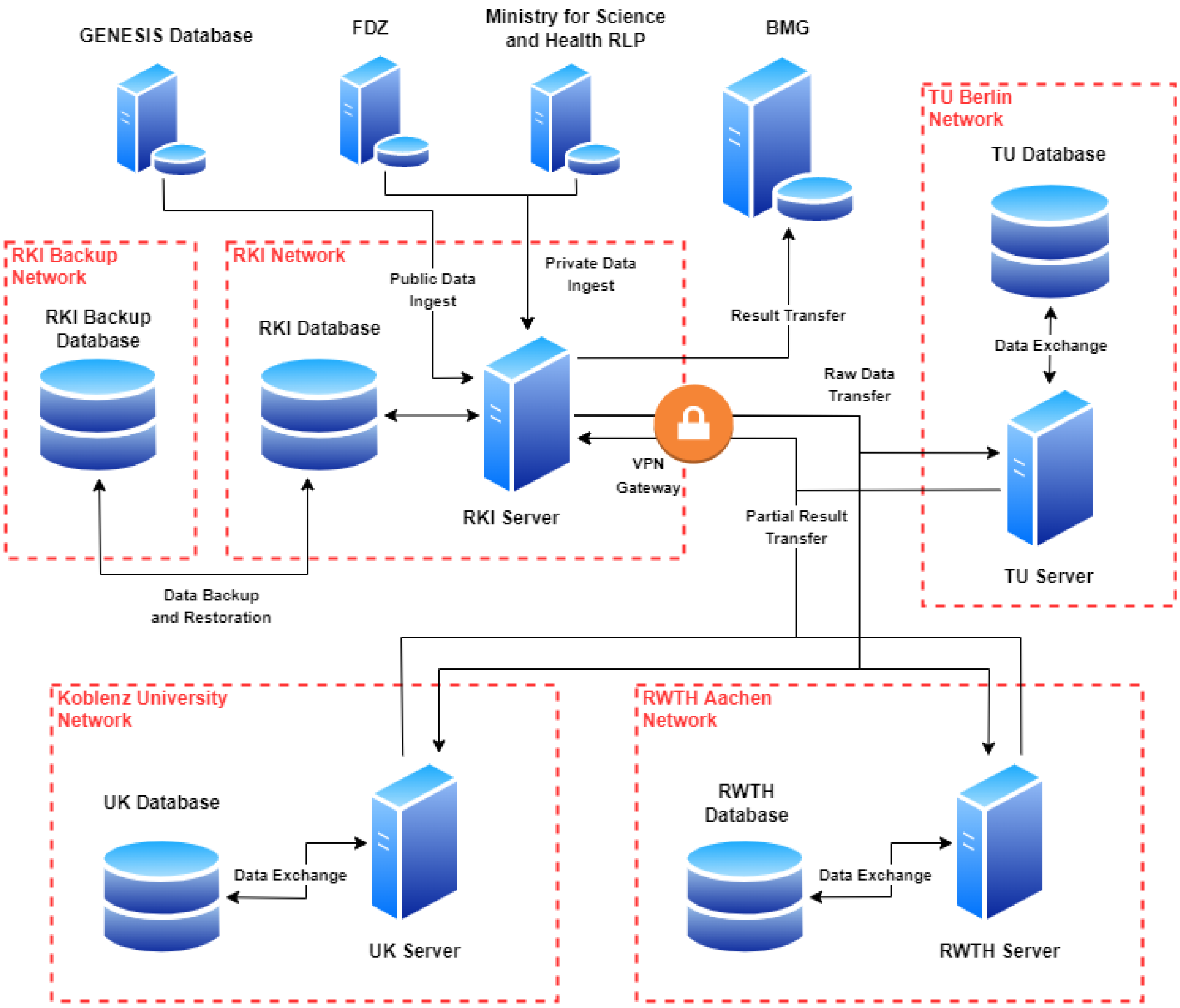


**Figure 7.6:** Top-level diagram of the planned HDIS after application of the DPRA.

To address the discrepancies between the DPRA and the initial architecture, as well implement missing technical requirements, the development team makes several changes to the initial system architecture which result in a new version shown in Figure 7.6. The missing backup database has been implemented verbatim to the DPRA-design with a separate network in order to protect the *RKI Backup Database* in the case of the *RKI Network* being compromised. To fulfill the *Tech.PR.4*, they channel the *Raw Data Transfer* and *Partial Result Transfer* data flows through a new *VPN Gateway*-component which prevents access to the *RKI Server* without previous authentication by the respective processor. Lastly, the ingest of private data from the *Ministry for Science and Health RLP* has been relocated to the *RKI Server* as well in order to facilitate control over the data and its distribution for processing.

While we limited our example to the top-level of the system architecture, in a real scenario this process would be much more comprehensive and would be propagated to lower levels of the architecture as well until all technical requirements and DPRA-components have been included.

# Chapter 8

# Conclusion

In this last chapter, we return to our five initial research questions presented in Section 1.3, and discuss how and to what degree the goals defined by them have been achieved. With this in mind we then consider aspects that couldn't be covered in this thesis due to restrictions in time and scope as well as further lines of research that could be worthwhile subjects to future works.

## 8.1 Sources for Data Protection Requirements

In Section 1.3.1, we set ourselves the goal of finding legal, industrial, governmental and scientific sources that could serve as a basis for the definition of base requirements for the inclusion of data protection in HDIS. We further defined the three quality metrics of legal relevance, ubiquity of scope and stability with regards to its content to evaluate them. Using these metrics, we conducted a structured search and assessment of potential sources which we described in Chapter 3. There, we discussed each group of sources and their respective advantages and disadvantages and evaluated some of the most prominent representatives of each group in more detail. During the process, we added usability as a fourth metric as we found multiple sources to be good with regards to the first three criteria but unusable for this thesis due to their scope and methodology being aimed at larger organizations.

In the end, we came to the conclusion that legal sources in general and specific national and international data protection law in particular provide the best balance with regards to our quality criteria. This led to the selection of the GDPR and BDSG as the primary sources for the definition of our legal data protection requirements which serve as a basis for all further research steps. Besides their self-evident legal relevance and ubiquity as the top-level data protection regulations for the European Union and Germany respectively, they also provide excellent stability. High level laws tend to remain effective for longer periods of time with any changes being made to the original release instead of releasing an entirely new document, meaning they can be easily traced. While legal texts can suffer from unclear wording in places, their structured format makes them usable for smaller groups of developers as well as larger teams. Considering industrial and governmental standards, we found all considered examples to perform well in terms of stability and ubiquity due to an extensive scope, ongoing support and comprehensive change management. On the other hand, they perform less well with regards to legal relevance since even though their application can be used as an argument towards compliance with legal regulations there is no explicit reference to them in said regulations meaning that they can only receive legal relevance via case law. Industrial standards further suffered from usability issues due to their extensive scope and the fact that their proposed methodologies are unfeasible for use by individuals or small groups. Finally, scientific sources appeared to be the least valuable category of sources for our purposes. Not only do they have no inherent legal relevance, but also suffer from generally short periods of actuality and a lack of traceability for future updates.

**Industrial Standards and Future Legislation.** Due to our project's small scale, the scope of potential sources that could be feasibly considered was rather limited. A larger team would extend the amount of relevant sources that could be analyzed and would allow for the consideration of larger scale industrial standards, like the ISO-27000 series and the Common Criteria[23, 21], which could greatly improve the quality of requirements. Additionally, there were sources that we consider to be highly important but weren't released at the time of this thesis. Chief among them is the Artificial Intelligence Act, which is currently being developed by the European Union and is likely to be highly important for HDIS once effective.

## 8.2 Extraction of Data Protection Requirements.

The second research question, stated in Chapter 1.3.2, was mostly concerned with identifying a suitable framework to formulate and organize our legal requirements. We addressed this question in Chapter 4 where we first introduced the general idea of requirements engineering as a discipline and then discussed the specific issue of ambiguity and how it can be addressed through the use of requirement templates. After laying the theoretical foundation, we then continued with introducing our methodology for the practical extraction of legal data protection requirements from the GDPR and BDSG, describing our process step by step. Along the way, we introduced a number of categories under which we organize the finished repository and introduced the MASTeR-template, which we used for formalizing the requirements themselves. Lastly, we presented our results and discussed our findings with regards to the distribution of requirements across categories. The most conspicuous observation was the severe imbalance between the number of requirements adhering to specific categories. Requirements to transparency, which are mostly concerned with the information that has to be disclosed to data holders and other third parties by the controllers and processors, made up nearly half of all entries while more technical aspects like availability where only represented by a handful of individual requirements.

**Review of Legal Requirements.** While the process of requirements extraction worked well from a practical perspective, the nature of the used sources proved challenging with regards to the quality of the produced requirements. Legal texts in general and high-level legislation in particular pose the challenge of containing a high proportion of vague statements and legal technical terms. Even though the extraction process was conducted with all due diligence, the very limited legal expertise of the author and the lack of an expert reviewer during the thesis period means that there is a significant residual risk of individual misinterpretations affecting the quality of the finished requirements repository. A future independent review of the legal requirements by a legal expert could solve this issue and have a positive cascading impact on all artifacts resulting from this work.

## 8.3 Potential Risks and Threats to HDIS.

Next, we considered potential threats to HDIS in Chapter 5 in order to answer the third research question (see Section 1.3.4). We began with an introduction of important technical terms necessary to discuss this aspect of system security and a description of our methodology for conducting a threat analysis for HDIS. We decided to focus on the two aspects of threats to artificial intelligence and threats to general system security and defined types of targeted assets on whose security we would focus our research. Starting with threats to artificial intelligence, we first discussed various sources for gathering information about this category and introduced important concepts from the corresponding field of research. We then continued by analyzing various types of attacks on and inherent issues of AI, first in general and then with regards to their individual relevance for Health Data Intelligence Systems. We also considered potential mitigation strategies that could be used to address these issues during development and operation.

For general system security, our process of finding up-to-date sources for threat intelligence, identifying prominent threats, assessing their relevance for HDIS and lastly discussing effective mitigation strategies remained the same. However, the availability of better organized and curated sources like the CWE and ATT&CK allowed for a more structured approach which also took professional assessments of the risk and severity of each threat into account. It also facilitated the research of suitable mitigation strategies which we compiled, together with their respective threats, into the Threat Repository.

**Intelligence on AI-related Threats.** While researching general system threats was comparatively straightforward, our analysis of threats related to artificial intelligence was complicated by a lack of structured sources and statistical data on specific threats. The majority of sources consisted of scientific papers which introduced new attacks on AI-enabled systems themselves before proposing potential solutions and mitigations for them. While non-research-related sources like the Artificial Intelligence Incident Database[1] exist, they only provide surface-level information about individual incidents. This lack of information coincides with a lack of statistical data which would allow to assess the actual significance of various threat categories. A comprehensive, statistical analysis of the occurrence of different threat types in a real world-context would be very valuable, not only to improve the results of this thesis, but also to allow researchers to focus future projects on the most important threats.

**Future Sources.** As for the legal requirements, there are also sources relevant to our threat analysis that have not been released yet. The most important is D3FEND[2] which is a knowledge base containing a growing compilation of mitigation strategies, aimed at addressing the attack strategies contained in its sister project ATT&CK. D3FEND is currently in a beta stage and receiving frequent updates by the contributor community. Right now, the majority of mitigation strategies in our threat repository were selected based on short references within the threat sources themselves, which are not optimal as a source. In the future, the Threat Repository could be updated to include additional mitigation strategies from D3FEND and replace the current references with more comprehensive ones.

[1] https://incidentdatabase.ai/

[2] https://d3fend.mitre.org/

## 8.4 Designing a Reference Architecture

After laying the groundwork with the compilation of the Legal Requirements and Threat Repositories, we continued with answering the fourth research question which was concerned with finding a feasible way of enabling developers to implement data protection into their system architecture by design. To achieve this, our goal was to design a reference architecture focused on data protection that could be applied by any future development team planning to create a Health Data Intelligence System following the general specifications established in Chapter 2.

The concept of a reference architecture isn't clearly defined and therefore doesn't entail a standard set of documents or artifacts that need to be included. We therefore decided to have our Data Protection Reference Architecture consist of two major artifacts. The first is the Technical Requirements Repository, where we consider each type of high-level component we assume to be present in every HDIS and what technical requirements they would have to fulfill to comply with the legal requirements compiled in the Legal Requirements Repository. We also included requirements for the implementation of mitigation strategies from the Threat Repository to address the threats identified in Chapter 5. The second artifact is a set of data-flow diagrams visualizing the concrete data stores, data flows and processes described in Section 6.4, which we consider essential for achieving basic compliance with legal data protection requirements. These components are deliberately abstract as their purpose is to act as a framework by which developers can structure their concrete design and implementation, using technologies and specific components they consider to be the most suitable to their respective system.

**Concrete Security Technologies and Products.** While we originally had the intention of including recommendations for specific technologies into the DPRA, we eventually decided against it during the design process. On one hand, choosing and recommending concrete solutions and security products would have introduced another dynamic factor into our artifacts that would have to be monitored constantly to prevent the promotion of potentially obsolete technologies. On the other hand, it would bear the risk of steering developers into specific lines of thinking when researching and selecting solutions for the implementation of their HDIS. In our attempt at finding a good trade-off between reducing the workload for future development teams and keeping the reference architecture flexible, we therefore relegated any decision regarding the selection of concrete technologies and products to the developers using the DPRA and focused instead on the requirements that those technologies will have to fulfill.

**Organizational Data Protection Measures.** Data Protection cannot be achieved solely through technical measures but has to be considered from an organizational and operational perspective as well. As we pointed out in Section 4.3, around half of the legal requirements don't directly impact HDIS itself, but are rather concerned with information that has to be provided to either the general public or individual data subjects and supervisory authorities on demand. The exact information depends on a multitude of individual factors like the precise sets of data used by the HDIS, where that data is sourced from, by what methods it is processed and on what legal basis it will be disclosed to which third parties. Since all these aspects can be highly dynamic during the operation lifecycle of a system, this mandatory information has to be reviewed and updated frequently. Other important aspects are easy access to and a user-friendly presentation of the information. All of these factors strongly influence public trust in an organization and its systems which – as mentioned in Section 5.3.5 – is a vital factor for any project working with private data, even more so if they are reliant on public funding. A future project could analyze the operational perspective of HDIS and analyze the potential for the creation of a second framework to facilitate the identification and provision of all information necessary to ensure compliance with transparency-requirements. It might also be possible to devise standardized procedures and organizational structures that could be adapted by future HDIS development projects.

## 8.5 Applying the Reference Architecture

With our last research question, which is covered by Chapter 7, we wanted to find and present a suitable methodology for applying the DPRA and its supporting artifacts and consider their respective usability. For this, we considered a classic development process based on the waterfall model and inserted three additional steps to include them into the process. They include the determination of legal requirements using the Legal Requirement Repository, the determination of internal and external threats with the help of the Threat Analysis Repository, and finally the enhancement of the system architecture to provide basic data protection using the Technical Requirements Repository and the DPRA. We then provided an example for how the inclusion of these additional steps could play out in a real world context. To that end, we defined the premise for a theoretical HDIS project and described how each step could be integrated into its development process.

**Assessment of Usability.** While our example presented in Section 7.5 might provide an initial idea of how our process could be applied in an actual project, it still remains a theoretical scenario and thus can't provide any empiric insight into the usability of the DPRA and other artifacts in a real world context. For this, we'd require our architecture to first be applied in such a project and collect comprehensive feedback from the development team as to how they used the artifacts, how easy it was to incorporate them into their usual processes and of what value it was to the process in terms of additional security and saved development time.

**Coordinated Change Management.** Even though we tried to take longevity of our results into account by focusing on ubiquitous, high level solutions, the field of system security is highly dynamic and there is a considerable risk of individual entries of our repositories becoming outdated or fully obsolete with time. To at least partially counteract this, we included the verification of the contents of our repositories before use as an important part of our application process described in Chapter 7. This verification however, only immediately updates the local copy used by the respective development team and doesn't affect the original artifacts on the project repository. It would therefore be valuable to devise and implement a coordinated process that would allow for the suggestion and discussion of updates to the original repository by interested users. This would not only allow future projects to benefit off of the insights of past ones, but also improve the quality of the repository through the inclusion of a wider community of users who could introduce their individual expertise.

# Acronyms

**ATT&CK** Adversarial Tactics, Techniques and Common Knowledge – Knowledge base operated by the MITRE Corporation which collects and categorizes cyberattack and intrusion tactics [46].

**BDSG** Bundesdatenschutzgesetz – Federal data protection act of Germany which is supplemented by further industry-specific regulations and individual data protection acts released by each German Federal State [1].

**BSI** Bundesamt für Sicherheit in der Informationstechnik – Top-level German federal agency which defines information and cybersecurity standards for the German federal government and provides recommendations for non-governmental organizations.

**CAPEC** Common Attack Pattern Enumeration – Knowledge base released by the US Department of Homeland Security and operated by the MITRE Corporation. It collects and categorizes known cyberattack patterns.

**CWE** Common Weakness Enumeration – Knowledge base operated by the MITRE Corporation which collects and categorizes internal weaknesses of information systems.

**DPRA** Data Protection Reference Architecture – One of the major research artifacts generated in the wake of this project. It is described in detail in Chapter 6.

**GDPR** General Data Protection Regulation – Regulation enacted by the European Union which focuses on data protection and privacy within its borders [2].

**GSK** IT-Grundschutz-Kompendium 2022 – Compendium of requirements for released by the BSI to facilitate the implementation of information system security for Germany-based organizations [28].

**HDIS** Health Data Intelligence System – Decision support system whose development is the subject of the *KI und Covid*-project [11].

**RKI** Robert Koch-Institut – Top-level German federal agency which is responsible for disease control and prevention..

**TLS** ENISA Threat Landscape 2021 – Annual report released by the European Union Agency for Cybersecurity. It describes the current status and new developments in cybersecurity, focusing on threats, threat actors and mitigation strategies [58].

# Bibliography

[1] Bundesministerium des Inneren, für Bau und Heimat. *Bundesdatenschutzgesetz: BDSG*. June 23, 2021. URL: http://www.gesetze-im-internet.de/bdsg_2018/ (visited on 09/27/2021).

[2] European Parliament, Council of the European Union. *General Data Protection Regulation*. Apr. 27, 2016. URL: https://gdpr.eu/tag/gdpr/ (visited on 09/27/2021).

[3] Majed Alshammari and Andrew Simpson. „Towards a Principled Approach for Engineering Privacy by Design". In: *Privacy Technologies and Policy*. Ed. by Erich Schweighofer et al. Vol. 10518. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pp. 161–177. ISBN: 978-3-319-67279-3. DOI: 10.1007/978-3-319-67280-9_9.

[4] Gerardo Schneider. „Is Privacy by Construction Possible?" In: *Leveraging Applications of Formal Methods, Verification and Validation. Modeling*. Ed. by Tiziana Margaria and Bernhard Steffen. Vol. 11244. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018, pp. 471–485. ISBN: 978-3-030-03417-7. DOI: 10.1007/978-3-030-03418-4_28.

[5] Sebastian Bretthauer. „Herausforderungen der Digitalisierung im Gesundheitswesen: Corona-Warn-App, Forschungsdaten und Künstliche Intelligenz". In: *Die Verwaltung* 54 (2021), pp. 411–441.

[6] Bundesministerium für Gesundheit, ed. *E-Health. Digitalisierung im Gesundheitswesen*. URL: https://www.bundesgesundheitsministerium.de/e-health-initiative.html (visited on 02/08/2022).

[7] Die Bundesregierung, ed. *Die Corona-Warn-App*. URL: https://www.bundesregierung.de/breg-de/themen/corona-warn-app (visited on 02/08/2022).

[8] Robert Koch Institut, ed. *COVID-Zertifikate der EU digital nachweisen*. URL: https://www.digitaler-impfnachweis-app.de/ (visited on 02/08/2022).

[9] Robert Koch Institut, ed. *Robert Koch-Institut: Covid-19 Dashboard*. 2022. URL: https://experience.arcgis.com/experience/478220a4c454480e823b17327b2bf1d4 (visited on 05/12/2022).

[10] *Bundestag stimmt für Infektionsschutzgesetz*. 2021. URL: https://www.tagesschau.de/inland/infektionsschutzgesetz-bundestag-corona-101.html (visited on 03/01/2022).

[11] Maria A. Wimmer et al. „KI und Covid: Erklärbarkeit und Entscheidungsunterstützung durch KI in Pandemie-Situationen". Koblenz, Feb. 17, 2021.

[12] M. Mahmudul Hasan. „Regulatory Requirements Compliance in Requirements Engineering". In: *International Journal of Systems and Service-Oriented Engineering (IJSSOE)* 6.4 (2016), pp. 22–35. ISSN: 1947-3052. DOI: 10.4018/IJSSOE.2016100102.

[13] Paul N. Otto and A. Antón. „Addressing Legal Requirements in Requirements Engineering“. In: *15th IEEE International Requirements Engineering Conference* (2007). DOI: 10.1109/RE.2007.65.

[14] Izar Tarandach and Matthew J. Coles. *Threat Modeling*. First edition. Sebastopol: O'Reilly, 2020. ISBN: 9781492056553. URL: https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=6395807.

[15] Christina Tikkinen-Piri, Anna Rohunen, and Jouni Markkula. „EU General Data Protection Regulation: Changes and implications for personal data collecting companies“. In: *Computer Law & Security Review* 34.1 (2018), pp. 134–153. ISSN: 02673649. DOI: 10.1016/J.CLSR.2017.05.015.

[16] Landtag Rheinland-Pfalz. *Landesdatenschutzgesetz: LDSG*. May 8, 2018. URL: https://landesrecht.rlp.de/bsrp/document/jlr-DSGRP2018rahmen (visited on 04/05/2022).

[17] European Commission, ed. *Your gateway to the European Union*. URL: https://european-union.europa.eu/index_en (visited on 04/04/2022).

[18] European Commission, ed. *Proposal for a Regulation of the European Parliament and of the Council - Laying Down Harmonised Rules on Artificial Intelligence (Artificial Intelligence Act) and Amending Certain Union Legislative Acts*. Apr. 4, 2022. URL: https://artificialintelligenceact.eu/the-act/.

[19] Bundesministerium für Gesundheit. *Gesetz zum Schutz elektronischer Patientendaten in der Telematikinfrastruktur (Patientendaten-Schutz-Gesetz): PDSG*. Oct. 14, 2020.

[20] *Sozialgesetzbuch Fünftes Buch - Gesetliche Krankenversicherung: SGB V*. Dec. 20, 1988.

[21] Bundesamt für Sicherheit in der Informationstechnik et al., ed. *Common Criteria for Information Technology Security Evaluation: Part 1: Introduction and general model*. 2017. URL: https://www.commoncriteriaportal.org/cc/ (visited on 02/22/2022).

[22] International Organization for Standardization, ed. *ISO/IEC 15408-1:2009(en) Information technology — Security techniques — Evaluation criteria for IT security — Part 1: Introduction and general model*. 2009. URL: https://www.iso.org/obp/ui/#iso:std:iso-iec:15408:-1:ed-3:v2:en (visited on 04/05/2022).

[23] International Organization for Standardization. *ISO/IEC 27001:2019(E): Information technology - Security techniques - Information security management systems - Requirements*. URL: https://www.iso.org/isoiec-27001-information-security.html (visited on 03/08/2022).

[24] Bundesamt für Sicherheit in der Informationstechnik, ed. *BSI-Standard 200-1: Management-Systeme für Informationssicherheit (ISMS)*. 2017. URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Grundschutz/BSI_Standards/standard_200_1.html (visited on 01/22/2022).

[25] Bundesamt für Sicherheit in der Informationstechnik, ed. *BSI-Standard 200-2: IT-Grundschutz-Methodik*. 2017. URL: https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/IT-Grundschutz/BSI-Standards/BSI-Standard-200-2-IT-Grundschutz-Methodik/bsi-standard-200-2-it-grundschutz-methodik_node.html (visited on 02/12/2022).

[26] Bundesamt für Sicherheit in der Informationstechnik, ed. *Kriterien für die Standortwahl von Rechenzentren: Standort-Kriterien RZ: Version 2.0*. 2019. URL: https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Informationen-und-Empfehlungen/Empfehlungen-nach-Angriffszielen/Hochverfuegbarkeit/Standort-Kriterien_HV-RZ/Standort-Kriterien_HV-RZ_node.html (visited on 02/12/2022).

[27] Bundesamt für Sicherheit in der Informationstechnik, ed. *AI Cloud Service Compliance Criteria Catalogue (AIC4)*. 2021. URL: https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Informationen-und-Empfehlungen/Kuenstliche-Intelligenz/AIC4/aic4_node.html (visited on 11/08/2021).

[28] *IT-Grundschutz-Kompendium*. Köln: Reguvis, 2022. ISBN: 978-3-8462-0906-6. URL: https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/IT-Grundschutz/IT-Grundschutz-Kompendium/it-grundschutz-kompendium_node.html (visited on 05/01/2022).

[29] Sandra Domenique Ringmann, Hanno Langweg, and Marcel Waldvogel. „Requirements for Legally Compliant Software Based on the GDPR“. In: *On the Move to Meaningful Internet Systems. OTM 2018 Conferences*. Ed. by Hervé Panetto et al. Vol. 11230. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018, pp. 258–276. ISBN: 978-3-030-02670-7.

[30] Klaus Pohl. *Requirements engineering: Fundamentals, principles, and techniques*. Berlin and Heidelberg: Springer, 2010. ISBN: 978-3-6421-2577-5.

[31] Martin Hollmann. *Legal Requirements_v1.0*. 2022. URL: https://github.com/DeadPenguin/Engineering-Data-Protection-Aware-Health-Data-Intelligence-Systems (visited on 08/14/2022).

[32] Elizabeth Hull, Ken Jackson, and Jeremy Dick. *Requirements Engineering*. London: Springer London, 2011. ISBN: 978-1-84996-404-3. DOI: 10.1007/978-1-84996-405-0.

[33] Ian Sommerville. *Software engineering*. 9th ed., [international ed.] Boston: Pearson, 2011. ISBN: 978-0-1370-3515-1.

[34] Karl Wiegers and Joy Beatty. *Software requirements*. 3. ed. [fully updated and expanded]. Best practices. Redmond, Wash.: Microsoft Press, 2013. ISBN: 978-0-7356-7966-5.

[35] Alistair Mavin et al. „Easy Approach to Requirements Syntax (EARS)“. In: *2009 17th IEEE International Requirements Engineering Conference*. IEEE, 2009, pp. 317–322. ISBN: 978-0-7695-3761-0. DOI: 10.1109/RE.2009.9.

[36] Axel Hoffmann and et. al. *Towards the Use of Software Requirement Patterns for Legal Requirements*. 2012. DOI: 10.2139/ssrn.2484455.

[37] Konstantinos Mokos and Panagiotis Katsaros. „A survey on the formalisation of system requirements and their validation“. In: *Array* 7 (2020), p. 100030. ISSN: 25900056. DOI: 10.1016/j.array.2020.100030.

[38] Muneera Bano. „Addressing the challenges of requirements ambiguity: A review of empirical literature“. In: *2015 IEEE Fifth International Workshop on Empirical Requirements Engineering (EmpiRE)*. IEEE, 2015, pp. 21–24. ISBN: 978-1-5090-0116-3. DOI: 10.1109/EmpiRE.2015.7431303.

[39] D. Majumdar et al. „Automated Requirements Modelling With Adv-Ears“. In: *International Journal of Information Technology Convergence and Services* 1.4 (2011), pp. 57–67. ISSN: 22311939. DOI: 10.5121/ijitcs.2011.1406.

[40] SOPHIST GmbH, ed. *MASTER: Schablonen für alle Fälle*. Nürnberg, 2019. URL: https://www.sophist.de/fileadmin/user_upload/Bilder_zu_Seiten/Publikationen/Wissen_for_free/MASTeR_Broschuere_5-Auflage_Komplett_Lesezeichen_Update_web.pdf (visited on 05/03/2022).

[41] AK Technik der Konferenz der unabhängigen Datenschutzaufsichtsbehörden des Bundes und, ed. *Das Standard-Datenschutzmodell: Eine Methode zur Datenschutzberatung und -prüfung auf der Basis einheitlicher Gewährleistungsziele: v.2.0b*. 2020. URL: https://www.datenschutzzentrum.de/sdm/ (visited on 01/25/2022).

[42] Donald Firesmith. „Specifying Reusable Security Requirements". In: *The Journal of Object Technology* 3.1 (2004), p. 61. DOI: 10.5381/jot.2004.3.1.c6.

[43] James Bayne. *An Overview of Threat and Risk Assessment*. Ed. by SANS Institute. 2021. URL: https://www.sans.org/white-papers/76/ (visited on 05/18/2022).

[44] Communications Security Establishment Canada, ed. *Harmonized Threat and Risk Assessment (TRA) Methodology*. 2007. URL: https://cyber.gc.ca/en/guidance/harmonized-tra-methodology-tra-1 (visited on 03/04/2022).

[45] National Institute of Standards and Technology, ed. *Guide for Conducting Risk Assessment*. 2012. URL: https://csrc.nist.gov/publications/detail/sp/800-30/rev-1/final (visited on 03/04/2022).

[46] MITRE, ed. *MITRE - ATT&CK*. 2021. URL: https://attack.mitre.org/ (visited on 03/01/2022).

[47] The MITRE Corporation, ed. *CAPEC - Common Attack Pattern Enumeration and Classification: A Community Resource for Identifying and Understanding Attacks*. Feb. 25, 2022. URL: https://capec.mitre.org/index.html (visited on 02/28/2022).

[48] The MITRE Corporation, ed. *CVE - Common Vulnerability Enumeration*. 2022. URL: https://cve.mitre.org/ (visited on 02/16/2022).

[49] The MITRE Corporation, ed. *CWE - Common Weakness Enumeration*. 2022. URL: https://cwe.mitre.org/ (visited on 02/16/2022).

[50] International Organization for Standardization. *ISO/IEC 27005:2018(E): Information technology — Security techniques — Information security risk management*. URL: https://www.iso.org/standard/75281.html (visited on 03/08/2022).

[51] Greenbone Networks GmbH, ed. *OpenVAS – Open Vulnerability Assessment Scanner*. URL: https://www.openvas.org/index.html (visited on 03/08/2022).

[52] Tenable, Inc., ed. *Nessus*. URL: https://www.tenable.com/products/nessus (visited on 03/08/2022).

[53] Dario Amodei et al. *Concrete Problems in AI Safety*. June 21, 2016. URL: https://arxiv.org/abs/1606.06565 (visited on 06/25/2022).

[54] Doowon Jeong. „Artificial Intelligence Security Threat, Crime, and Forensics: Taxonomy and Open Issues". In: *IEEE Access* 8 (2020), pp. 184560–184574. DOI: 10.1109/ACCESS.2020.3029280.

[55] Qiang Liu et al. „A Survey on Security Threats and Defensive Techniques of Machine Learning: A Data Driven View". In: *IEEE Access* 6 (2018), pp. 12103–12117. DOI: 10.1109/ACCESS.2018.2805680.

[56] Felipe Giuste et al. *Explainable Artificial Intelligence Methods in Combating Pandemics: A Systematic Review*. Dec. 23, 2012. URL: https://arxiv.org/abs/2112.12705 (visited on 06/15/2022).

[57] David Solans et al. *Human Response to an AI-Based Decision Support System: A User Study on the Effects of Accuracy and Bias*. Mar. 24, 2022. URL: http://arxiv.org/pdf/2203.15514v1.

[58] European Union Agency For Network And Information Security, ed. *ENISA Threat Landscape 2021*. 2021. URL: https://www.enisa.europa.eu/publications/enisa-threat-landscape-2021 (visited on 03/07/2022).

[59] Battista Biggio and Fabio Roli. „Wild Patterns: Ten Years After the Rise of Adversarial Machine Learning". In: *Pattern Recognition* 84.3 (2018), pp. 317–331. ISSN: 00313203. DOI: 10.1016/j.patcog.2018.07.023.

[60] Adversa AI, ed. *The Road to Secure and Trusted AI: The Decade of AI Security Challenges*. 2020. URL: https://adversa.ai/report-secure-and-trusted-ai/ (visited on 04/23/2022).

[61] Bundesamt für Sicherheit in der Informationstechnik, ed. *Secure, robust and transparent application of AI: Problems, measures and need for action*. 2021. URL: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/KI/Secure_robust_and_transparent_application_of_AI.pdf (visited on 04/22/2022).

[62] Bundesamt für Sicherheit in der Informationstechnik, ed. *Towards Auditable AI Systems*. 2021. URL: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/KI/Towards_Auditable_AI_Systems.pdf (visited on 04/22/2022).

[63] European Commission, ed. *Ethics Guidelines for Trustworthy AI*. 2019. URL: https://op.europa.eu/en/publication-detail/-/publication/d3988569-0434-11ea-8c1f-01aa75ed71a1 (visited on 04/23/2022).

[64] International Organization for Standardization, ed. *Standards by ISO/IEC JTC 1/SC 42 - Artificial Intelligence*. 2022. URL: https://www.iso.org/committee/6794475/x/catalogue/ (visited on 04/24/2022).

[65] Nicolas Papernot et al. „SoK: Security and Privacy in Machine Learning". In: *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2018, pp. 399–414. ISBN: 978-1-5386-4228-3. DOI: 10.1109/EuroSP.2018.00035.

[66] Mengchen Zhao et al. „Efficient Label Contamination Attacks Against Black-Box Learning Models". In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*. Ed. by Fahiem Bacchus and Carles Sierra. California: International Joint Conferences on Artificial Intelligence Organization, 2017, pp. 3945–3951. ISBN: 978-0-9992-4110-3. DOI: 10.24963/ijcai.2017/551.

[67] Ville Vakkuri, Kai-Kristian Kemell, and Pekka Abrahamsson. „Implementing Ethics in AI: Initial Results of an Industrial Multiple Case Study". In: *CoRR* abs/1906.12307 (2019). DOI: 10.48550/arXiv.1906.12307.

[68] Marco Barreno et al. „Can machine learning be secure?" In: *Proceedings of the 2006 ACM Symposium on Information, computer and communications security - ASIACCS '06*. Ed. by Ferng-Ching Lin. New York, New York, USA: ACM Press, 2006, p. 16. ISBN: 1595932720. DOI: 10.1145/1128817.1128824.

[69] Sean McGregor. *Preventing Repeated Real World AI Failures by Cataloging Incidents: The AI Incident Database*. Nov. 17, 2020. URL: https://arxiv.org/abs/2011.08512 (visited on 02/13/2022).

[70] Ian J. Goodfellow et al. *Generative Adversarial Networks*. June 10, 2014. URL: https://arxiv.org/abs/1406.2661 (visited on 06/27/2022).

[71] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. „Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures“. In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. Ed. by Indrajit Ray, Ninghui Li, and Christopher Kruegel. New York, NY, USA: ACM, 2015, pp. 1322–1333. ISBN: 9781450338325. DOI: 10.1145/2810103.2813677.

[72] Florian Tramèr et al. *Stealing Machine Learning Models via Prediction APIs*. Sept. 9, 2016. URL: https://arxiv.org/abs/1609.02943 (visited on 04/03/2022).

[73] Matthew Jagielski et al. *Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning*. 2018. DOI: 10.48550/arXiv.1804.00308.

[74] Seira Hidano et al. „Model Inversion Attacks for Prediction Systems: Without Knowledge of Non-Sensitive Attributes“. In: *2017 15th Annual Conference on Privacy, Security and Trust (PST)*. IEEE, 2017, pp. 115–11509. ISBN: 978-1-5386-2487-6. DOI: 10.1109/PST.2017.00023.

[75] Christian Szegedy et al. *Intriguing properties of neural networks*. Dec. 21, 2013. URL: https://arxiv.org/abs/1312.6199 (visited on 05/06/2022).

[76] Nicola Davis. „From oximeter to AI, where bias in medical devices may lurk“. In: *The Guardian* (Nov. 21, 2021). URL: https://www.theguardian.com/society/2021/nov/21/from-oximeters-to-ai-where-bias-in-medical-devices-may-lurk (visited on 04/20/2022).

[77] Stefano Sarao Mannelli et al. *Inducing bias is simpler than you think*. May 31, 2022. URL: https://arxiv.org/abs/2205.15935 (visited on 04/14/2022).

[78] Swapnil Kangralkar. *Types of Biases in Data: Biases in data that we should all be aware of to build a reliable and fair machine learning model*. 2021. URL: https://towardsdatascience.com/types-of-biases-in-data-cafc4f2634fb (visited on 06/21/2022).

[79] Ninareh Mehrabi et al. *A Survey on Bias and Fairness in Machine Learning*. 2019. DOI: 10.48550/arXiv.1908.09635.

[80] Lindsay Weinberg. „Rethinking Fairness: An Interdisciplinary Survey of Critiques of Hegemonic ML Fairness Approaches“. In: *Journal of Artificial Intelligence Research* 74 (2022), pp. 75–109. DOI: 10.48550/arXiv.2205.04460.

[81] Martin Hollmann. *Threat Repository_v1.0*. 2022. URL: https://github.com/DeadPenguin/Engineering-Data-Protection-Aware-Health-Data-Intelligence-Systems (visited on 08/14/2022).

[82] Robert Koch Institut, ed. *COVID-19: Fallzahlen in Deutschland und weltweit*. 2022. URL: https://www.rki.de/DE/Content/InfAZ/N/Neuartiges_Coronavirus/Fallzahlen.html (visited on 06/21/2022).

[83] Felix Biessmann et al. *Automated Data Validation in Machine Learning Systems*. Ed. by Institute of Electrical and Electronics Engineers. 2021. URL: https://www.semanticscholar.org/paper/Automated-Data-Validation-in-Machine-Learning-Biessmann-Golebiowski/dba4edca4a55bde937077804b2f03cdd8b4cd419 (visited on 04/18/2022).

[84] Ling Huang et al. „Adversarial machine learning“. In: *Proceedings of the 4th ACM workshop on Security and artificial intelligence - AISec '11*. Ed. by Yan Chen et al. New York, New York, USA: ACM Press, 2011, p. 43. ISBN: 9781450310031. DOI: 10.1145/2046684.2046692.

[85] Ian Goodfellow, Patrick McDaniel, and Nicolas Papernot. „Making machine learning robust against adversarial inputs“. In: *Communications of the ACM* 61.7 (2018), pp. 56–66. ISSN: 0001-0782. DOI: 10.1145/3134599.

[86] Yongfeng Zhang and Xu Chen. „Explainable Recommendation: A Survey and New Perspectives“. In: *Foundations and Trends® in Information Retrieval* 14.1 (2020), pp. 1–101. ISSN: 1554-0669. DOI: 10.1561/1500000066.

[87] Yingqiang Ge et al. „Explainable Fairness in Recommendation“. In: *SIGIR ′22: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2022). DOI: 10.1145/3477495.3531973.

[88] Wojciech Samek et al. „Explaining Deep Neural Networks and Beyond: A Review of Methods and Applications“. In: *Proceedings of the IEEE* 109.3 (2021), pp. 247–278. ISSN: 0018-9219. DOI: 10.1109/JPROC.2021.3060483.

[89] „On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation“. In: *PloS one* 10.7 (2015), e0130140. DOI: 10.1371/journal.pone.0130140.

[90] MISP Project, ed. *MISP - Threat Sharing*. URL: https://www.misp-project.org/ (visited on 03/08/2022).

[91] Luatix, ed. *Open cyber threat intelligence platform*. URL: https://www.opencti.io/en/ (visited on 03/11/2022).

[92] Martin Hollmann. *Technical Requirements_v1.0*. 2022. URL: https://github.com/DeadPenguin/Engineering-Data-Protection-Aware-Health-Data-Intelligence-Systems (visited on 08/14/2022).