Technische Universität München
TUM School of Computation, Information and Technology

Evaluation of Risks and Security Controls for Industrial Components within Connected Manufacturing

Alexander Giehl

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitz: Prof. Dr.-Ing. Georg Sigl

Prüfende der Dissertation:

1. Prof. Dr. Claudia Eckert
2. Prof. Dr.-Ing. Reiner Anderl

Die Dissertation wurde am 26.10.2023 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 26.04.2024 angenommen.

Acknowledgments

First of all, I would like to thank Prof. Dr. Claudia Eckert for giving me the opportunity to conduct research within the area of information security. Her guidance and encouragement during the entire process of writing this thesis has been of great help to me. I would also like to thank Prof. Dr-Ing. Reiner Anderl for his interest in my thesis and for agreeing to be my second examiner.

Furthermore, I would like to thank my colleagues at Fraunhofer AISEC for always making time for discussions, providing feedback, and their overall support throughout the years. I especially thank Dr. Konstantin Böttinger for his constructive feedback and continuous encouragement.

Last but certainly not least, I thank my parents for their love and support in my life.

Abstract

The digital transformation of the manufacturing domain is taking place in countries all over the world. New digital technologies allow for an increase in productivity and allow new business models to be realized. The trend to stronger interconnection often results in information security challenges. In order to address these challenges, the peculiarities of the manufacturing domain need to be taken into account.

We present a comprehensive approach for conducting information security evaluations within manufacturing. Information security evaluations support in establishing the principles of security-by-design into industrial production. Our work takes the domain-specific requirements into account and is focused on providing future-oriented methods and tools.

We begin by describing a modeling method that allows us to capture the wide variety of manufacturing environments in existence. From there, we develop a testbed that allows for the integration of digital twinning technology and the simulation of attack scenarios specific for manufacturing. We further show how data for those experiments can be acquired from connected factories in a privacy-preserving manner.

Our results show that it is possible to conduct information security evaluations in manufacturing by considering its domain-specific requirements. Specifically, we consider the emerging technology of digital twins in our research. Our testbed, thus, offers a contribution to improve upon the current state of information security in manufacturing by being future-proof at the same time.

Zusammenfassung

Die digitale Transformation im Bereich des produzierenden Gewerbes findet weltweit in verschiedenen Ländern statt. Neue digitale Technologien ermöglichen eine Steigerung der Produktivität und die Umsetzung neuer Geschäftsmodelle. Die zunehmende Vernetzung führt jedoch oft auch zu Herausforderungen im Bereich der Informationssicherheit. Um diesen Herausforderungen zu begegnen, müssen die Besonderheiten des Fertigungsbereichs berücksichtigt werden.

Wir präsentieren einen umfassenden Ansatz zur Durchführung von Informationssicherheitsbewertungen in der Produktion. Informationssicherheitsbewertungen tragen dazu bei, die Prinzipien von sicherem Design in der industriellen Produktion zu etablieren. Unsere Arbeit berücksichtigt die domänen-spezifischen Anforderungen und konzentriert sich darauf, zukunftsorientierte Methoden und Werkzeuge bereitzustellen.

Wir beginnen mit der Beschreibung einer Modellierungsmethode, mit deren Hilfe wir die Vielfalt der existierenden Fertigungsumgebungen erfassen können. Darauf aufbauend entwickeln wir eine Testumgebung, die die Integration von digitalen Zwillingen sowie die Simulation von Angriffsszenarien innerhalb der Fertigung ermöglicht. Weiterhin zeigen wir, wie Daten für Experimente auf eine datenschutzorientierte Weise aus vernetzten Fabriken ausgeleitet werden können.

Unsere Ergebnisse zeigen, dass es möglich ist, Informationssicherheitsbewertungen in der Fertigung unter Berücksichtigung ihrer domain-spezifischen Anforderungen durchzuführen. Insbesondere betrachten wir zukunftsfähige Technologien wie digitale Zwillinge in unserer Forschung. Unsere Testumgebung trägt somit dazu bei, den aktuellen Stand der Informationssicherheit in der Produktion zu verbessern, während sie gleichzeitig zukunftssicher ist.

Contents

1 Introduction

In this chapter, we provide an overview on the topics and concepts discussed within the remainder of this thesis. We provide a comprehensive summary of our research that was conducted over the course of several years.

We start by giving our rationale for conducting our research in Section 1.1. We outline our motivation behind undertaking this study and provide a brief background on the domain our research is taking place, that is, manufacturing and industrial automation. We emphasize the need to address limitations in regard to information security and sketch the methods we use to achieve this.

This is followed by a discussion on the challenges encountered by us during our work on this thesis in Section 1.2. These challenges encompass a range of different aspects that required addressing by us. Those include methodological, theoretical, and practical aspects for which we needed to derive, test, and implement different solutions for. By exploring these challenges and by reflecting on them, we gained a deeper understanding in our area of research.

Understanding the background of our research as well as identifying the challenges we faced initially, we are able to formulate research questions. Our research questions are summarized in Section 1.3). They are guiding us through our entire work and help us on focusing our efforts in methodology selection, data collection, and evaluation.

This allowed us to achieve a number of results that form our scientific contributions to the field of information security evaluations in manufacturing. They all are listed in Section 1.4), our contributions allowed us to examine our research subject more accurately and in more depth than before. We produce practical results that shows our research has tangible results that can be used for future applications also beyond academia. Nonetheless, this thesis contributed to academic studies by disseminating knowledge through publications and conference presentations.

Finally, we provide an outline for the structure of our thesis in Section 1.5. We present a clear roadmap of the subsequent chapters that facilitates navigation through our study. With this, our flow of ideas is structured and specific information can be located more easily.

1.1 Motivation

Over the past decades, the domain of manufacturing is experiencing a substantial, ongoing transformation. This transformation started in the 1960s with the introduction of digital and automation technologies in the domain of manufacturing. This followed up on the previous introduction of steam engines and mass assembly. This new transformation is commonly referred to as the Third Industrial Revolution [1, 2, 3] (see Section 2.1). The Third Industrial Revolution introduced a tremendous change in the way products are manufactured. Some of the technologies that aided within that transformation are, for example, computer-aided design (CAD), computer-aided manufacturing (CAM), programmable logic controllers (PLCs), or robotics. All of these and other technologies are still used today and allow for more efficient production of goods. Also, products can be manufactured with increased precision, which results in a generally improved product quality than previously possible. The Third Industrial Revolution and its associated technologies form the basis for the currently observed trend in manufacturing. For this trend the term *Fourth Industrial Revolution* is used [4]. This is also described by other terms, for instance *Industrie 4.0* in Germany or Industrial Internet-of-Things (IIoT) in the USA [5]. It facilitates the ongoing integration of digital technologies in manufacturing by emphasizing connectivity and data exchange. While the Third Industrial Revolution introduced digital technologies such as computers, the Fourth Industrial Revolution builds upon these technologies and expands upon them with new approaches such as Internet-of-Things (IoT), artificial intelligence (AI), cloud computing, or big data analytics. Those digital technologies can be used in modern manufacturing environments to connect machines and devices located on the shopfloor within the factory. The aim is to create a network of smart machines and devices that can communicate with each other and exchange data in real-time. Thus, creating a so-called smart factory. This increase in connectivity and data exchange among machines within a smart factory is generally received favorably by plant operators. For them, digitization of the manufacturing process goes along with optimization of production processes, improvement in product quality, and reduction of costs. For example, sensors can be used to monitor the performance of machines and predict when maintenance is required, which reduces unplanned downtime and, thus, increases the overall productivity of the plant [46, 47]. Another example is using data analytics to analyze production data and optimize manufacturing processes

to reduce waste and also to increase efficiency. In addition to this, new business models are enabled with Industrie 4.0 [1]. One example for such a business model is Product-as-a-Service (PaaS). In PaaS, companies sell products not as a one-time purchase but rather as an ongoing service [6, 7, 8]. This involves providing ongoing maintenance, upgrades, and support for the PaaS with the intention of maintaining a continuous revenue stream. It is enabled by increased connectivity of devices via the internet and by more software-centered products. This ongoing support can include software updates, performance optimization, or addition of new features. PaaS can, in theory, be applied to a range of manufactured products, such as industrial machinery (e.g., tooling machines), appliances, or vehicles. Another example for a business model enabled by technologies from Industrie 4.0 is mass customization or customer-individual production [9, 10, 11]. With it, manufacturers can offer customized products by producing smaller batch sizes and by adjusting production equipment more efficiently. This is enabled by data analytics, advanced robotics systems, and flexible manufacturing systems. Data analytics in turn is enabled, e.g., by the increased connectivity and the ubiquitous amount of sensors contained in modern devices. Manufacturers can change production based on collected customer data or customer orders and, thus, produce products that are made with individual customers or a group of customers in mind.

The technologies used in Industrie 4.0, like IoT and cloud computing, create a complex and interconnected digital ecosystem that poses significant challenges from an information security perspective [12, 13, 14]. An increase in connectivity also leads to an increased attack surface for manufacturing environments and equipment located within (see Section 2.1.3). This is due to the fact that, instead of novel technologies being introduced in the domain of manufacturing, the rate of change within the domain of manufacturing is still slower than in other domains where IT is being used [15, 16, 17, 18]. Also, manufacturing has properties that are specific to this domain. Together, the overall slow rate of change and the custom properties of manufacturing make information security challenging within this domain [19]. One reason why information security is challenging in manufacturing is the high requirement for availability, which means that manufacturing equipment must be operational and accessible at all times to ensure uninterrupted production [20] (see also Section 4.1.2). Any downtime or disruption of service in manufacturing systems can result in significant production losses, which themselves result in revenue losses. Thus, plant operators have low tolerance for service failures of manufacturing

equipment. In case of unexpected failures, this can even result in physical damage to equipment or personal. Downtime in manufacturing can be caused by a number of different reasons, for example, failure of equipment, power outages, or unexpected software behavior. All of these reasons and many others can be caused, for instance, by a cyber attack that is executed on the interconnected devices and their networks. It is, thus, reasonable to protect the availability of the manufacturing equipment from cyber attacks.

However, the protection of manufacturing equipment from cyber attacks is complicated by other factors such as the widespread use of legacy systems within modern production setups. Many manufacturing systems still rely on legacy equipment and software that was initially not designed considering information security. Manufacturing systems were, for the most part of their history, being considered as isolated systems with only limited or no access to other systems. When isolated systems without security features get connected to a larger (public) network, a pathway for outside attackers to those systems is created. Such systems and their software are typically not maintained on a regular basis or even at all. This results in their operating systems and software not being up to date and, therefore, vulnerable to cyber attacks. The widespread reliance on such legacy systems in manufacturing posses a significant information security challenge due to their outdated security features, their lack of ongoing support, and vulnerabilities being left unaddressed. Moreover, most of such legacy systems also lack basic security features, e.g., the use of encryption or authentication measures. In addition, legacy systems may not be able to receive critical security updates and patches, e.g., as updates are no longer produced for these systems or are difficult to bring to offline devices. This is further complicated by the high runtimes and slow exchange rates of manufacturing systems, components, and protocols. The total operation for manufacturing equipment with 20 or more years where a device or technology can be in service is comparably long. Therefore, production equipment is often designed to run for years up to decades without significant changes or updates. This can make it difficult to implement security updates and patches in a timely manner when new vulnerabilities are discovered. In many manufacturing environments, manufacturing systems are left vulnerable to exploits that maybe known for years and might never receive a security update. Furthermore, because manufacturing equipment is designed to be in operation for a prolonged time span, it often relies on outdated hardware and software components. These components may not even be able to support modern

security protocols, leaving systems vulnerable to attacks even if a security patch is available. This lack of resources is further aggravated by the high constraints in those resources. These constraints are raised by the manufacturing equipment and manifest in, e.g., limited memory, processing power, and storage capacity for manufacturing devices. This results in a limited ability to implement security updates as outlined above. Furthermore, it is in general challenging to implement strong or even sufficient encryption on manufacturing devices [21, 22, 17]. Without proper methods of encryption, many manufacturing devices are unable to protect sensitive data and to prevent unauthorized access. This is further complicated by the fact that there are real-time requirements present in a production environment. Real-time capability of manufacturing environment is essential in production because most production processes require precise control and coordination of multiple systems and components. Thus, there exist several requirements for real-time monitoring and control within manufacturing environments in order to ensure the production process is executed and behaves as expected. For example, in a manufacturing setup that uses a sequential assembly line, each step within the production process is timed and synchronized in order to ensure that the manufactured product is assembled in the specified way and meets its quality criteria. Security features integrated within the product, e.g., encryption protocols or authentication mechanisms, can introduce additional requirements that can result in an increased demand on computational resources and, thus, introduce a processing overhead. This overhead can potentially impact system performance and its responsiveness to real-time requirements. For the reasons stated so far in this section, manufacturing environments are heterogeneous in nature. Manufacturing systems evolve over time, with new components and technologies being added as needed to support changing business models and production requirements. Furthermore, custom-made control equipment is present in many of these systems. For example, a manufacturing system may use specialized hardware to enable high-speed data acquisition or custom software for real-time control. Especially in manufacturing execution systems (MESs), the middle layer of hard- and software in factory automation, custom solutions and software are present [9]. As a result, manufacturing systems have the tendency to become more complex and heterogeneous over time. Integrating security measures in such a heterogeneous environment is challenging, especially when also dealing with legacy systems not originally designed with support for security controls in mind.

To summarize, manufacturing systems are often considered less secure than other IT systems in more established IT domains such office networks or e-commerce systems. This is due to the reasons discussed above. These reasons include the strong requirement for availability for manufacturing equipment, the common use of legacy systems, the long operation time of manufacturing systems, their inherent resource constraints, and the heterogeneous manufacturing environment. Other reasons like supply-chain security and post-quantum security also add to the overall threat landscape within manufacturing systems but are not within the scope of this thesis.

Thus, it is important to consider information security when developing new business models, enabling technologies, and enhancing existing manufacturing environments. Attacks on unprotected devices and their possible effects can have a huge impact on other systems and connected processes. This is illustrated by a number of attacks on manufacturing systems in the recent past [16, 18] (see also Section 2.1.3). Arguably the most prominent example of a cyber attack on a manufacturing system is the Stuxnet malware that surfaced between 2009 and 2011 [23]. It is considered to be an event that accelerated the interest in manufacturing security and IIoT security [24, 13]. The reason for this is that the Stuxnet malware showed a high level of sophistication in its design and resulted ultimately in the destruction of physical manufacturing equipment. In order to achieve this, the Stuxnet malware used a multi-staged attack vector and exploited several zero-day vulnerabilities in the process. Those vulnerabilities were located in commercial operating systems as well as in industrial control systems (ICS) and allowed Stuxnet to infiltrate, move within, and manipulate the targeted systems. A total of four zero-day exploits was used by Stuxnet for this. It infiltrated presumably by breaching the air gap via infected USB devices, then propagated within the IT network until reaching its targets within the operational technology (OT) network (which was also isolated from the IT network). Stuxnet was able to manipulate the programmable logic controllers (PLCs) used in industrial control systems. The malware was programmed to target specific PLCs used in an Iranian nuclear facility and alter their programming in order to cause a malfunction within centrifuges presumably used for uranium enrichment. The malware specifically targeted those PLCs and, while doing so, avoided affecting other systems or causing any visible damage until the right conditions were met. Thus, Stuxnet avoided detection and behaved stealthy. Due to the stealthy nature of Stuxnet, long-term infiltration of the target facility was possible with the result of wide reaching physical destruction of OT equipment. Stuxnet highlights the

potential of cyber attacks to target manufacturing facilities with their OT devices and their impact on those systems as it caused physical damage within the facility. Stuxnet is a striking example of a cyber attack on the manufacturing domain. Most observed attacks, however, do not show the high level of sophistication exhibited by Stuxnet. For example, ransomware attacks have become a major threat to manufacturing companies in recent years [18, 25]. Ransomware attacks are a type of attack where attackers access computer networks and encrypt data found within the network. Then, the attackers demand a ransom payment from the affected company in exchange for the decryption key. Modern manufacturing companies are particularly vulnerable to ransomware attacks as their manufacturing processes are highly reliant on data and its availability. Making production data unavailable can cause significant disruption to the manufacturing process. The attacks are eased by some of the properties in manufacturing as discussed above, e.g., outdated software and lack of (strong) security countermeasures. However, attackers are more aware of manufacturing as a potential target and the potential impact of the executed attacks on manufacturing facilities can potentially be high. In addition to this, the total number of attacks on manufacturing is steadily increasing rapidly over the last years. These trends show that the domain of manufacturing is in need of information security.

This is supported by conducting information security evaluations. Information security evaluations describe methods and tools that allow to determine the capabilities of a system in regard to information security. These systems include existing and implemented technological solutions as well as upcoming technologies in manufacturing, e.g., digital twins [26, 1, 27, 28]. Information security evaluations hereby refers to the process of assessing the effectiveness of implemented or planed security measures. The purpose of information security evaluations is to identify potential vulnerabilities, weaknesses, and gaps in the existing security measures, as well as to provide recommendations for improving the overall security of a system or organization [29]. Information security evaluations can be used for every domain that is employing IT systems. This includes the domain of manufacturing as well as other related domains such as critical infrastructures or automotive [30, 31]. The usage of information security evaluations in manufacturing, thus, aids in determining potential threats as well as in evaluating countermeasures against those threats. A widespread method for conducting information security evaluations in manufacturing are information security testbeds [32, 24, 33]. A testbed is a scien-

tific environment or platform for conducting scientific rigorous experimentation for testing equipment and technologies [34, 27]. Such testbeds provide a controlled environment where security measures can be tested and evaluated without affecting the actual manufacturing operations. This way, the availability of the manufacturing environment is not affected making testbeds very relevant for information security evaluations in manufacturing. Thus, testbeds allow for a more thorough evaluation of security measures, including their effectiveness, scalability, and compatibility with existing systems as it would be possible within a real-world setup on a manufacturing shopfloor. This further applies to studying a system's behavior under attack conditions. Here, testbeds allow to study attacks and attacker behavior without the risk of harm to the actual manufacturing equipment. This way, testbeds can be used to test and evaluate new security controls and allow manufacturers to assess the potential benefits and risks of these security controls before they are deployed in a real-world setting. For the reasons stated above, using testbeds for information security in manufacturing offers some benefits. However, testbeds also have limitations to their usefulness in studying security and cyber attacks in manufacturing. Testbeds, as any simulation, suffer form an inherent lack of realism [35, 36]. Testbeds cannot accurately reflect the real-world manufacturing environment as they are limited by their simulation technique. This can limit the usefulness of the evaluation results as they may not be applicable to real-world scenarios. Another related aspect is limited variability [37, 38]. Testbeds are not able to capture the full range of variables and interactions that occur in a real-world manufacturing environment especially within a connected environment. These limits the scope of the evaluation and the accuracy of the results. Another aspect is that long-term testing and evaluation is typically not considered by research testbeds as they are constructed for specific use cases and scenarios [33]. If the testbed does not accompany the simulated systems over a prolonged time span, it limits the testbed's ability to identify and address issues that arise over time, e.g., from cyber attacks such as Stuxnet.

Testbeds share similarities with one enabling technology in Industrie 4.0: the digital twin. A digital twin is a virtual replica of a manufacturing system, either already existing or still being in development [39, 40]. Like testbeds, digital twins are used for the evaluation of components or systems [41, 33]. Both, testbeds and digital twins, rely on simulation to create a virtual model of a manufacturing system. They differ here in their simulation techniques, their approach to testbed construction, and in their security design objectives (see Section 3.1). Despite these different di-

mensions between testbed and digital twin, both provide a controlled environment for evaluating and optimizing manufacturing systems in regard to information security. They allow to test and evaluate new technologies, software, and protocols in a controlled environment as well as enable manufacturers to simulate and optimize system performance. The data basis differs as testbeds tend to consume historical data, i.e., pre-recorded sets of data that are played back into the testbed, whereas digital twins are also constructed to consume data from their physical counterpart in real-time. Nevertheless, this indicates another similarity between the two concepts, that is their data-driven approach. Both, testbeds and digital twins, rely on data to simulate and optimize manufacturing systems.

The concept of a digital twin in manufacturing, however, is extended beyond the scope of a testbed. The digital twin in manufacturing accompanies its physical counterpart during its entire lifecycle and allows high accuracy testing of industrial equipment [28] (see Section 2.1.2 and Section 4.3 for details and definition). Testbeds are typically constructed for a specific purpose, e.g., training of personal or vulnerability assessment [33]. Digital twins, on the other hand, are a replica of a real system and can, therefore, in theory be used for the same applications as the real system. Furthermore, as the digital twin accompanies its physical counterpart during its entire lifecycle, i.e., from the engineering phase over its operation until end-of-life, a digital twin is also can support in detecting advanced persistent threats (APTs) like Stuxnet. Finally, the level of accuracy and realism in the simulation of the manufacturing system (i.e., the fidelity of the simulation) is considered to be the most distinguishing and relevant property for digital twins [28, 42]. In the context of digital twins, fidelity is a critical part in construction of the digital twins because its objective is to create a virtual replica of the physical system that is as accurate and realistic as possible. The higher the fidelity of the digital twin, the more accurate the predictions and optimization recommendations can be. This is, however, the key challenge in the realization and implementation of digital twins in manufacturing at the moment [18].

The construction of a digital twin with a high level of fidelity is currently not achievable [42]. For a true high fidelity digital twin, the acquisition of large amounts of data from sensors and other sources from a manufacturing environment in real-time to simulate the behavior and performance of the system is required. Given the current state of manufacturing with legacy systems present, strong constraints of resources, and heterogeneous technological landscapes, a solution appears to be out

of reach for now. Technical solutions like increased network speeds on the shopfloor level and higher computational processing power in IIoT equipment can mitigate this in the future if they become available and are deployed within manufacturing environments. Despite this, digital twins offer high potential for improving the state of information security within manufacturing (see above). As they are not practically achievable at the moment, they can be used for information security use cases in manufacturing in the future when properly prepared. This thesis explores how and to what extend digital twin technology can be used to improve upon the current state-of-the-art in information security in manufacturing. We offer a platform architecture that can be used in modern business contexts and allows to include information security evaluations within manufacturing now and for future applications.

1.2 Challenges

In this section, we discuss the challenges encountered by us during the course of our research. These challenges encompass methodological, technical, and interdisciplinary obstacles. They provide the basis to understanding the context and the results of our study. Our motivation for this thesis given in the previous section already hints at some of the challenges we faced during our research. This section provides a comprehensive discussion of the challenges relevant to our research. To summarize, the main challenges faced by us are:

1. **Insecure development lifecycle for information security**:
   A modern, connected factory is characterized by a large number of interconnected components and interdependent processes [1]. These processes and the underlying technologies have been established over a long period of time, especially in countries with a long tradition of industrialization such as Germany. Information security was not considered during the establishment of now existing infrastructure [15]. The reason for this includes a lack of awareness for the potential risks associated when using information security within manufacturing. Rather, the focus of plant operators was more on physical security measures related to safety in order to protect personal and equipment. Information security was not considered to be a major concern when digitized manufacturing systems were first developed and introduced broadly within the 1960s [2]. This was enforced by the cost and complexity of implementing information security in manufacturing devices as well as the lack of regulatory requirements for information security within manufacturing. Information security controls using cryptographic primitives, e.g., for encryption or authentication, introduce an overhead to communication and processing time [22]. This can be challenging when operating under hard real-time requirements as it is typically the case within manufacturing [21]. Additionally, most regulatory requirements addressed to manufacturing plants and industrial automation were concerned with safety initially. Information security for manufacturing only received attention comparably late to safety regulations [43]. This results in a lack of standardization for information security in manufacturing. This can make it challenging to develop effective information security solutions. This is especially true when considering relatively new technologies such as digital

twins and how they can be applied to increase the state fo information security within manufacturing [28].

2. **Domain-specific requirements for information security**:
   Manufacturing environments, as diverse as they may be, share a number of common properties that also reflect on the requirements for information security [16]. Information security in manufacturing is, thus, an interdisciplinary field of study. Manufacturing systems and equipment used for industrial automation is required and expected to be in operation over a prolonged time span. Typically, a time frame of at least 20 years can be assumed for operation [44, 45]. During this period of time, down times are needed to be kept to an absolute minimum [46, 47]. The protection goal of availability is, thus, the most important concern for plant operators and, thus, all other activities within a plant are second to sustaining availability [48, 20]. This reflects in the choices made by decision makers such as plant operators in manufacturing. Therefore, we are required to understand our target domain in order to provide reasonable security controls. An understanding of the manufacturing domain is necessary in order to identify the specific security risks and requirements associated with manufacturing. This can require specialized knowledge of manufacturing processes, equipment, and technologies. Furthermore, different manufacturing environments probably have different security requirements based on such factors as their installed equipment, established processes, and data involved. Researchers may need to address a diverse set of security requirements, which can be challenging and time-consuming. Such requirements need to be taken into account also when designing development testbeds and digital twins for information security within manufacturing [28, 33]. Satisfying domain-specific requirements is necessary for the effective integration of information security within manufacturing. Otherwise, security controls and evaluations are not able to function properly. Addressing the domain-specific requirements of manufacturing further increases the acceptance of information security controls and, in addition, makes those controls more effective in their operating environment.
3. **Data sovereignty by private sector**:
   Modern manufacturing systems are data-driven and produce a large amount of data by themselves [49]. The data produced by modern manufacturing systems can provide valuable insights into production processes, vulnerabili-

ties of the equipment, and supply chain operations. However, access to this data is regulated by private companies as the data produced is part of their business operations. As a consequence, data for research is difficult to acquire from the private sector. A variety of factors contributes to this including the protection needs for intellectual property (IP) by plant operators and IP owners as they operate within a competitive environment [46, 47]. Most manufacturing companies consider their data to be proprietary information that provides a competitive advantage to them. Consequently, if data is accessed by other entities such as competitors or public research can result in losing this advantage. As a result, data related to production is kept private and collaboration can be limited by hesitation to share data. Another factor complementing this is the proprietary nature of most equipment used in industrial automation [50, 51, 17]. This can make it difficult, also for plant operators, to access these systems and their data. This makes accessing data for research purposes difficult as the acquisition of meaningful data is often not possible directly from plants. However, data acquisition and analysis is particularly important for information security evaluations and data-driven use cases like digital twins [52, 28]. Without access to sufficient data, researchers face significant challenges in effectively testing the efficiency of their information security solutions in this area. Therefore, a method for acquiring data from manufacturing environments supports in providing substantiated evaluations as well as a basis for future research activities.

These are the challenges that required addressing by us in order to increase the state of information security in connected manufacturing. The impact of these challenges on the research process and the subsequent findings are the topics of the following chapters.

1.3 Research Questions

The challenges described in Section 1.2 give rise to the following research questions we aim to address within the context of this thesis:

- **Research Question 1:** How does a platform for digital twins have to be designed to enable information security evaluations for manufacturing environments?
- **Research Question 2:** What modeling technique is suitable for the study of attacks and countermeasures within a connected manufacturing system?
- **Research Question 3:** How can data be extracted from manufacturing environments for information security evaluations among competitors?

Research Questions 1 focuses on the design of a platform for digital twins that can facilitate information security evaluations in modern manufacturing environments. Digital twins are virtual representations of physical objects or systems that can be used for monitoring, analysis, and optimization of various processes [40] as well as for information security in manufacturing [28]. Our platform must take future technological developments into account in order to enable a digital twin to be used for information security evaluations [42]. In this context, digital twins can be used as part of a simulation of the production process for testing different attack scenarios and to draw conclusions from them. Information security evaluations are crucial in manufacturing environments to ensure that protection goals in manufacturing are not threatened or violated by attackers [48, 20]. This includes the availability of the production line as well as the confidentiality of sensitive data such as production plans or intellectual property such as product geometry [46, 47]. If designed accordingly, a platform for digital twins can enable such evaluations [33] and provide a basis for future manufacturing business models [1]. The platform would need to be designed with the ability to evaluate information security threats in manufacturing, damage to production equipment caused by malware or manipulation of production parameters [53]. Finally, the platform needs to be designed with scalability and flexibility in mind to accommodate the evolving needs of manufacturing environments especially in the context of connected manufacturing. This involves the integration of novel techniques for the integration of digital twins in production environments [54, 42].

Research Questions 2 focuses on identifying a modeling technique that can be used to study attacks and countermeasures within a connected manufacturing envi-

ronment [55]. Specifically, an environment where digital twins can or might be used in the future [28]. Connected manufacturing systems rely on networked devices and communication technologies to exchange large amounts of data, preferably in real-time. To study attacks and countermeasures within a connected manufacturing system, a modeling technique that can capture the system's complexity, dynamics, and interactions is needed. This is related to the type of manufacturing system being modeled, either discrete event manufacturing systems or continuous flow manufacturing systems. Each of these manufacturing comes with its own challenges [56, 57]. The behavior of the corresponding manufacturing systems need to be captured by the modeling technique. Furthermore, the behavior of attackers in a connected manufacturing system as well as the impact of cyber attacks on manufacturing environment need to be expressed by modeling of the system. Additionally, modeling of systems that rely strongly on software and algorithms for control of the managed process must be addressed by the modeling technique. The modeling technique serves as the basis for the simulation of the platform as outlined by Research Question 1. In combination, both can be used to simulate the effects of cyber attacks on the system's behavior and evaluate digital twins for information security in manufacturing.

Research Question 3 focuses on extracting data from manufacturing environments in a secure and private manner for the purpose of conducting information security evaluations among competitors. Extracting data from manufacturing environments can be a complex and sensitive process, as it involves accessing potentially sensitive information about the production process and the manufactured products [46, 47]. Additionally, companies may share concerns of sharing data for collaborative data analysis with their competitors. This data analysis are, however, a potential application for future business models [1, 8] and also might yield more immediate benefits, e.g., condition monitoring or predictive maintenance. Shared data analysis can also be useful for the purpose of information security evaluation as they allow to combine insights from different manufacturing setups [12, 16]. To address this research question, secure and private-preserving data extraction methods need to be used. This involves appropriate security controls as well as algorithms for keeping data private but useful for evaluations. With the right approach, companies can extract valuable insights from manufacturing environments while maintaining the confidentiality of sensitive information.

To summarize, current and future manufacturing environments often lack information security controls and processes for their integration into plant operations [12, 58]. Therefore, we need to understand the specifics of the manufacturing domain in order to understand how to do and how to integrate information security evaluations in manufacturing. We also need to understand and demonstrate the effects that the presence and absence of security controls can have in manufacturing environments. For this, we develop methods and a technical architecture to perform information security evaluations. Those are combined within a testbed that serves as the basis for our thesis. This enables us to show the feasibility of our approach. Also, future scenarios need to be considered by us. As outlined in the previous sections, a transformation is taking place within manufacturing. With this, new risks and challenges for the domain of manufacturing arise; however, it is also a chance for improving upon the current state of information security. Overall, our research questions highlight the complex and multifaceted nature of information security in manufacturing environments and the need for careful consideration of several factors when addressing these challenges.

1.4 Contributions

In this thesis, we aim at creating a better understanding on how to conduct meaningful information security evaluations in connected manufacturing with the help of digital twins. For this, we develop domain-specific methods that assist in conducting information security evaluations in manufacturing. We show how such evaluations can be integrated into existing processes and technologies. We further consider different use cases and evaluate them within a technical architecture developed by us. Also, we consider the topic of data acquisition, which further assists us in conducting meaningful analysis and can serve as a basis for ensuring the acquisition of meaningful data for future research.

To summarize, this thesis brings the following contributions to the field of information security in manufacturing:

- **Contribution 1: Integration of information security evaluations in manufacturing via modeling and simulation.**
  One contribution by us to the field of information security in manufacturing focuses on the integration of information security evaluations in the manufacturing domain through the use of modeling and simulation techniques. By this, we address the increasing need for enhanced integration of information security in manufacturing, as it is becoming more interconnected and reliant on digital technologies. By utilizing modeling and simulation via UML state machine charts, we propose a framework that enables the evaluation of information security in manufacturing [9]. We achieve this by initially developing a virtual environment that is extended by digital twinning technologies [55, 59]. Our testbed environment can be used to identify vulnerabilities and potential cyber-attacks, enabling the study of information security with digital twins. Our solution provides a comprehensive approach to information security evaluations that can be used in manufacturing. The use of modeling and simulation techniques allows for a more efficient and cost-effective evaluation of security measures, while also enabling future technologies by the integration of digital twins. We provide several improvements on modeling and simulation of information security within manufacturing. This includes a metric [60] and modeling approach [9, 55] specifically developed for manufacturing facilities.
- **Contribution 2: A future-oriented testbed for enabling digital twins in manufacturing.**

Another contribution by us is focused on the development of a future-oriented testbed for enabling digital twins in manufacturing. Digital twins are virtual representations of physical systems and have received increasing attention in the recent years due to their potential in manufacturing, also for use cases in information security [28]. We propose a testbed that enables information security evaluations within connected manufacturing by making use of digital twin technology [55]. Our testbed comprises various hardware and software components, including sensors, actuators, and simulation tools. By using this testbed, it is possible to create and test digital twins together with their physical counterpart. It enables the simulation of different attack scenarios for information security together with information security controls. Furthermore, our future-oriented design of the testbed ensures that the testbed remains relevant and useful as digital twin technology continues to evolve. We achieve this by relying on testbed guidelines and established processes in manufacturing. We provide a reference framework for the testbed and an implementation tested with several scenarios [60, 21, 55, 47]. The testbed is designed for integration of virtual components into a simulation. Furthermore, it allows for a context switch between real and virtual devices, which aids in verification of the simulation results.

- **Contribution 3: A privacy preserving data acquisition method for industrial environments.**
  Our next contribution is focused on the development of a privacy-preserving data acquisition method for industrial environments. Data acquisition is a critical process in the manufacturing industry, as it enables researchers to collect and analyze data from manufacturing systems for conducting research, e.g., on information security within manufacturing. However, there are concerns about the privacy of the data such as protection of intellectual property [46, 47]. We proposes a method that enables for the collection of large amounts of data from manufacturing environments while preserving the privacy of sensitive information. The method is based on a distributed architecture, where data is collected from multiple sources, anonymized with differential privacy, and transmitted to a cloud environment off-premise. The usage of differential privacy ensures that sensitive information remains private, while still allowing for analyzing data for certain use cases. In our case, these use cases are related to information security with other use cases such as predictive maintenance

or condition monitoring reaming possible as well. Our contribution provides a practical solution to the challenge of data acquisition from industrial environments. The proposed architecture of the method also allows for scalability and flexibility, enabling different companies to share data for analysis with each other. Overall, the proposed method has the potential to enhance data collection and analysis in industrial environments while addressing privacy concerns.

- **Contribution 4: Synthesis of anonymization with information security evaluation in connected manufacturing.**
  Our final contribution is focused on the synthesis of anonymization with information security evaluation and digital twins in connected manufacturing. Connected manufacturing refers to the use of interconnected digital technologies used in modern manufacturing environments [3]. We propose a unified framework that integrates anonymization techniques with information security evaluations using digital twin technology for manufacturing. By integrating these techniques, our framework enables us to collect and analyze data while preserving the privacy of sensitive information in order to conduct a meaningful evaluation of information security in the domain of manufacturing. The previously discussed contributions, Contribution 1 and Contribution 2, are concerned with the integration and evaluation of information security within the domain of manufacturing. We further show how to make use of the potential within connected manufacturing. For this, we provide a unified architecture for privacy preserving data acquisition from Contribution 3 [46, 47] with integrated testbed [55]. By using common interfaces and open-source technology, we provide the outline for a future-oriented, connected platform within connected manufacturing focused on information security.

The above contributions are verified through the feedback received from the scientific community on our publications as first author [9, 55, 60, 46, 21, 47]. The central idea of integrating information security evaluations in manufacturing is presented at the flagship automation conference of the IEEE Robotics and Automation Society: the 2020 IEEE 16th International Conference on Automation Science and Engineering (IEEE CASE 2020) [55]. The second major topic of this thesis, data extraction from connected manufacturing, is, furthermore, presented at IEEE CASE 2021 [47]. Furthermore, feedback was received from ACM conferences with a focus

on automation science [9, 60] and from IEEE conferences with a focus on information security [46, 21].

In addition, there are also our co-authored scientific publications [22, 49, 61, 62, 63] and our publications in trade journals in the areas of production, automation, and operational technology (OT) [64, 65, 66, 67, 68]. The foundation for the scientific contributions were made in the author's master's thesis where simulation of attacks in smart grid are discussed [30][1].

Furthermore, our contributions are verified by plant operators and decision makers in industry. In the context of several research projects we developed working solutions that take future business developments and modern technologies into account. Our efforts are aided by the following research projects:

- **IUNO**[2] (2015-2018) [3]
  IUNO is the German national reference project for security in Industrie 4.0. Together with 21 partners, four demonstration use cases are developed and implemented [10, 11].
  Notable deliverables with our contributions are [69, 10, 11, 70, 8, 71, 72, 73].
- **IUNO Insec**[3] (2018-2022)
  IUNO Insec is the follow-up project to IUNO and aims specifically at small- and medium-sized enterprises (SMEs). IUNO Insec is connected to four practical research projects with IUNO Insec being the main project.
- **A4O**[4] (2019-2020)
  Anonymization4Optimization aims at creating a secure and privacy-preserving architecture for cloud-based data analytics. A4O collaborates with several tooling machine manufacturers and operators.

Combining the feedback we received within these projects and from the discussions around our scientific publications, we believe that this thesis offers a sound and innovative but also usable contribution to the field of information security in manufacturing.

[1] Note that no work conducted in the author's master thesis has been reused or replicated in any way. Knowledge, experience and insight into modeling and simulation of OT gained from the thesis, however, has.

[2] Funding by Federal Ministry of Education and Research (BMBF), grant number 16KIS0324.
[3] Funding by BMBF, grant number 16KIS0933K.
[4] Funding by Arbeitsgemeinschaft industrieller Forschungsvereinigungen (AiF), grant number 20449 N.

1.5 Structure

We discuss the structure of our thesis in this section. The intention behind this is to provide guidance towards the reader and give hints on what topics might be of interest to a particular focused reviewer of our work. First, we briefly summarize the content of the individual chapters contained within this thesis. For a more detailed overview, we refer to the introductions of each chapter and to the summaries provided at the end of some of the chapters.

In Chapter 2 we give the background on the topics relevant to understanding our work in this thesis. Specifically, we examine the domain of manufacturing, information security within that domain, digital twins, modeling with the Unified Modeling Language (UML), and privacy. We discuss historical backgrounds and relevant individual works in these areas.

Recent work relevant to our thesis is given by Chapter 3. There we detail the published work that is relevant in categorizing our research and contributions within the broader research landscape. In particular we discuss paper and research related to information security with digital twins, modeling of digital twins, privacy-preservation in manufacturing with differential privacy, and privacy-preservation in manufacturing with federated learning.

In Chapter 4 we give the overall architecture for our work. That includes discussion on attacker models in manufacturing as well as relevant attack scenarios. These attack scenarios are introduced by us in a general manner and then detailed by providing descriptions of attack vectors. These attack scenarios are the basis for the evaluation of our work. We conclude Chapter 4 with the overall architecture for our testbed and data sharing applications.

In Chapter 5 we present our modeling approach that is used within this thesis. Our approach allows us to express information in manufacturing while providing a sound basis for the implementation of our work. We provide a reference scenario that describes how the attack scenarios described by Chapter 4 are embedded within a larger context of a specific manufacturing architecture with specific processes.

In Chapter 6 we go into details of the realization of our testbed for conducting information security evaluations in manufacturing with digital twins. We describe how the testbed is designed, realized, implemented, and evaluated. Two of the three attack scenarios given by Chapter 4 are the main point of interest here for

the evaluation of our testbed. The model and the integration of our digital twin is discussed in this section as well.

The testbed is one part of our conceptual architecture, the other part being our privacy-preserving data sharing application and platform. It is the topic of Chapter 7 where the topic of data acquisition within competitive environments is addressed. We describe our collaborative approach that involved data collection from heterogeneous manufacturing environment for provided value to all participants in that scheme in this chapter.

Our work and its results are summarized within Chapter 8. Among the discussion of our contributions, we also discuss possible future work with an outlook on how to further expand upon our work.

In principle, the chapters described above can be read in the provided order. However, the interested readers might very well skip ahead to the chapters that appear most relevant to them.

2 Background

In this chapter we discuss the relevant background to this thesis. We cover the relevant topics that contribute in large parts to understanding our work in this thesis. The background provided here is of a general nature. While it may touch down on outstanding individual research, we refer to Chapter 3 for a detailed delimination of our work from that of other authors. The purpose of this chapter is to provide a historical background where necessary and to provide a larger frame of reference to those topics.

In Section 2.1, we introduce the domain where our research is applied to: manufacturing in general and connected manufacturing in specific. The historical development of manufacturing has been driven by innovation over a period of more than three centuries [2]. We introduce these historical developments and highlight the key technologies that drove the continued development of the domain of manufacturing. The recent developments within this domain receive heightened attention from us. That is, the developments described by us with the term connected manufacturing [5]. We also touch down on the topic of terminology and provide insights into that domain.

Next, we discuss the background of the formal modeling language UML in Section 2.2. UML is the abbreviation for Unified Modeling Language, a commonly used abstract modeling framework that is defined by an official documentation [74]. We discuss UML and the specific components that are part if. For the most part, this involves different modeling techniques and approaches. In particular, we discuss UML state machine diagrams that are used by us for modeling within this thesis. We give an overview over the basic concepts and elements of UML state machine charts, their specific application to our research interests in discussed separately by us in Chapter 5).

In the last section of this chapter, we elaborate on the broad field of privacy. We discuss the term of privacy and narrow it down to the aspects that are of interest to us within the context of our work. That means we discuss the evolution of

privacy models and give a brief overview about some of their recent entries. We provide descriptive examples with it in order to illustrate applications of this field of research to real-world problems. From there, we discuss a group of privacy-enhancing technologies (PETs) and show how they can be generally be used for enhancing upon privacy models. We note that the terminology on PETs is not fixed at the moment [75], however, we provide our own meaning which is adopted throughout this thesis.

2.1 Manufacturing

Manufacturing is a process that outputs products and goods by processing a variety of different inputs [1, 2, 3]. These inputs include raw resources, advanced materials, labor, machines, tools, biological processes, and chemical processes. Manufacturing involves transforming these inputs to a specific product or good [9]. This is typically conducted through a series steps and production stages. The term manufacturing can refer to a wide range of activities, e.g., from producing goods by handicraft up to automated production of high-tech equipment. Most commonly, the term manufacturing is applied to industrial design, in which the different inputs are used and transformed into finished goods and products at a large scale. This is also our understanding of the term manufacturing that we use within this thesis. The manufacturing industry encompasses a wide range of sectors, for example, food products, textiles, electronics, vehicles, or aerospace.

Manufacturing can be observed throughout human history [76]. Its origins lie within artisanry and handicraft with the Latin term *manu factura* literally meaning *'a working hand'*. From there, manufacturing has changed significantly, especially within the last 350 years. Its evolution has been characterized by significant changes during that time frame with each transition between manufacturing periods being marked by technological advancements and innovations. The development of manufacturing is typically divided into four main periods, especially in discussions about the future development of the manufacturing sector [5]. Here, the historically established term of the *Industrial Revolution* is taken up [2]. Each area is understood to experience an industrial revolution, thus, counting a total of four industrial revolutions. The first of these periods is referred to as the *(First) Industrial Revolution*. This time period, also known as the period of mechanical manufacturing, describes the mechanization of handicraft in the late 18th century. Before the First Industrial Revolution, manufacturing was primarily hand labor using simple tools. Production was further confined to small workshops or private homes. The mechanization of these traditional production methods involving almost exclusively manual labor was enabled through the use of water and steam power. Within these newly developed production technologies, steam engines represent the key technological innovation. With this, the transition into mechanical manufacturing began. The mechanization of previously manual manufacturing processes resulted in a tremendous increase in production output. This enabled for the first time in history the mass production

of goods. This meant that existing industries could be massively extended and that new industries could be founded, for example, textile manufacturing or iron and coal mining. In addition to this, the First Industrial Revolution also had a tremendous impact on the overall society with its economy and even culture. It significantly changed to the way people lived and worked. It also laid the foundation for many of the modern manufacturing processes that can be observed today. From there, the next period in the history of manufacturing, the so-called *Second Industrial Revolution*, was reached. The transition to this time period was marked by the widespread introduction of electrical power. This is why the period of the Second Industrial Revolution is also referred to as electrical manufacturing. This is due to the new forms of power, i.e., electricity produced by the natural resources gas and oil, which emerged from the late 19th century onward. Electricity, again, tremendously changed the way manufacturing was conducted. This manifested in the way how factories and plants operated, as electricity replaced steam power as the primary source of power. Electric motors allowed for more precise and efficient control of machinery, resulting in an increase in production capacity. Electricity further enabled the creation of new industrial sectors such as steel, chemicals, and vehicles. Especially within the automotive industry, new production techniques were introduced. The assembly line and division of labor allowed to further increase efficient production of goods at large scale. Mass production itself was enabled by advanced tools, i.e., machine tools or tooling machines (the two terms are used interchangeably). Mechanized tools were also used during the First Industrial Revolution, however, due to the availability of electrical power, these machine tools could also be electrified. Before the widespread introduction of electricity in manufacturing, machine tools were powered by belts connected to steam engines. These mechanical power sources naturally are prone to vibrating and rattling, which made the machine tools difficult to operate and limited their potential for mass production as they required frequent adjustments and maintenance. This is not the case anymore when tooling machines are powered by electric motors. Electrical motors improved the precision and control of tooling machines. They allowed for more precise and controllable operation of machine tools resulting in their ability to create parts with higher accuracy and consistency. Also, electric motors required fewer stops for maintenance and refueling than mechanical engines. The development of machine tools from mechanized to electrified manufacturing equipment highlights that also improvement of existing technologies aided

in the evolution of manufacturing. Disruptive technological innovations, however, remain the main driver within the evolution of manufacturing.

Building up on the technological innovations from the First and Second Industrial Revolution, the *Third Industrial Revolution* emerged. It was enabled by continuing technological development from the previous revolutions and also introducing new digital technologies. Therefore, the Third Industrial Revolution is also referred to as digital manufacturing. Digital signal processing began in 1947 with the invention of the first transistor [77, 78]. From there, computer technology started to spread into almost all aspects of society. It is generally considered, that the Third Industrial Revolution started in 1969 with the development of the Advanced Research Projects Agency Network (ARPANET), the Internet's predecessor [79]. From this point on, computer technology and electronics were also introduced slowly but steadily in manufacturing giving rise to industrial automation [2]. From this point in time onward, manufacturing systems and processes received another significant change resulting from advancements in technology. These advancements started to take hold in manufacturing from the 1970s onward. During this time, manufacturing changed significantly, with the slow but steady introduction of computer technology into the domain of manufacturing. These new computer technologies included the use of microprocessors and software, which allowed plant operators to further automate the process of manufacturing products and goods. This could be achieved, among other things, with the introduction and consequent improvement of robotics for the use in manufacturing. In addition to this, programmable logic controllers (PLCs) saw their deployment and consequent use in manufacturing from the 1980s onward. PLCs are specialized automation devices designed for the usage in the rough industrial environments, e.g., directly on the shopfloor, where the products are manufactured. A PLC is an embedded computing device specifically designed for the purpose of industrial automation. They are tasked with the control of industrial processes. For this, they can interface and communicate with a large variety of different sensors and actuators as well as with other industrial devices. The main advantage of PLCs is that they are digitally programmable devices. This allows them to be used in in a variety of different setups and industries as well as being reprogrammed and, thus, being repurposed and adjusted to changing manufacturing requirements. PLCs allowed for the automation of more complex tasks then before and so the use of robots in manufacturing became more widespread. This includes manufacturing sectors such as automotive assembly lines or semiconductor fabrication. In current

discussions on the Third Industrial revolution, devices such as industrial robots and PLCs are summarized under the term of cyber-physical systems (CPSs). A CPS can be defined as a technology that manages the digital interconnection of physical assets and computing devices in an embedded component [53]. These CPSs are characterized through increasing complexity; they also provide more computational power than devices used in manufacturing before. The properties of CPS allow for extended interconnection of manufacturing devices and equipment between each other as faster and widespread means of communication technology, e.g., Ethernet, are employed on the shop floor [1]. Furthermore, data acquisition and analysis is enabled by those technologies in a way not possible before; this is often summarized by the term *big data*.

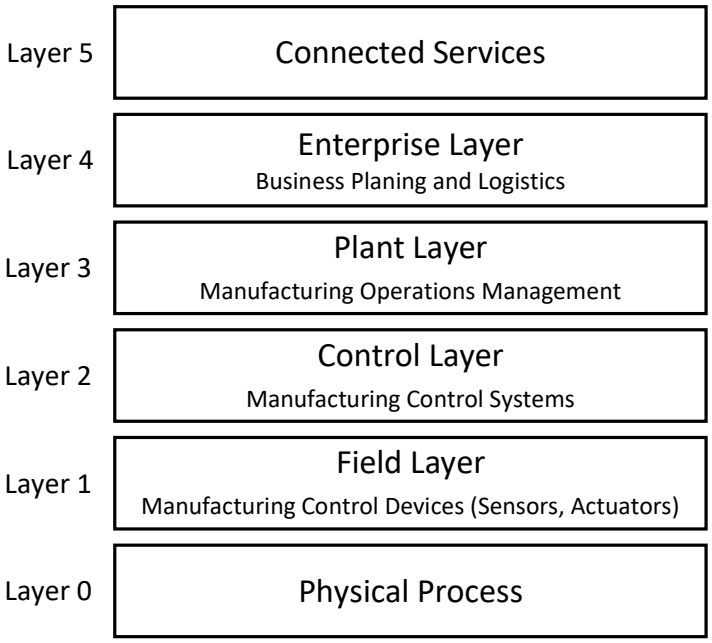


Figure 2.1: Layers of automation with extension towards Industrie 4.0.

The increased level of digital automation in the Third Industrial Revolution is subject to hierarchical structuring [80, 81]. Typically, the automation hierarchy is expressed as a number of layers expressing logical separation of automation equipment such as PLCs in manufacturing. The number of layers referenced to by literature, however, can vary depending on author and use case described. This is still

common to observe in literature even though standardization on industrial automation is available [82, 83] as the standards do not capture all aspects of industrial automation, especially the still changing and developing technologies in like Industrie 4.0 [21, 55]. This can be regarded as a new layer in itself as illustrated by Figure 2.1. Here, we adopt a view based on new developments in industrial automation that are related to Industrie 4.0 [84]. Industrie 4.0 can be seen on top of the figure named 'Connected Services'. This layer encompass a variety of business applications and enabling technologies. This is described in more detail later in this section. Further going from top to bottom of Figure 2.1, the layers of industrial automation are numbered according to established patterns [82, 80]. Layer 4 and Layer 3 are concerned with management applications that occur inside a plant. Layer 4 here is responsible for business planing and logistics and is also referred to as the enterprise layer. The enterprise layer consists of business-level applications such as enterprise resource planning (ERP) software, supply chain management systems, and other software tools for managing business operations. Together, these applications are able to provide a high-level view on the automated manufacturing system. This information is used for decision making about resource allocation in production and other business-related activities. Layer 3 is the the layer responsible for the management of manufacturing operations. It is also referred to as the plant layer. For the plant to effectively manage plant operations, a MES (Manufacturing Execution System) is implemented in the manufacturing environment. The MES collects and processes data from other industrial equipment, derives decisions based on the acquired data, and issues control signals back to the devices. PLCs, among other devices, are located on Layer 3 and mainly responsible for executing these operations. The devices located in the plant layer often communicate using industrial protocols such as Modbus, Profibus, or Ethernet/IP. The plant layer is the bridge between the enterprise systems and the actual manufacturing equipment. The manufacturing equipment is contained in Layer 2 and Layer 1. Both layers are related to manufacturing control with varying responsibilities. Layer 2, the control layer, is responsible for the collection of data from sensors located at the lower layer. Controllers receive that data from sensors and use it to send control signals to actuators to reach a defined outcome. For example, a controller connected to a temperature sensor might receive temperature readings from that sensor and adjust another connected heating or cooling device to maintain a set temperature range. Control systems at this level typically operate in real-time, responding to changes

in the process as they happen. These sensors and actuators are located at the field level, i.e., Layer 1. This layer is responsible for capturing raw data from physical processes and converting it into digital signals for processing by higher layers. Sensors measure and report changes in physical variables, e.g., temperature or position, whereas actuators control physical devices such as valves, motors, or pumps. Finally, the actual physical process of the plant is located at Layer 0. There are many possible physical processes that can be executed within a plant. They can be distinguished roughly in discrete and continuous manufacturing processes (see also the related discussion on CVDS and DEDS in Section 3.1.1). Continuous processes are such processes where raw materials are continuously received as input from the production line. This type of manufacturing processes are performed by industrial sectors such as chemical processing, oil refining, or power generation. Discrete manufacturing processes, on the other hand, involve the production of goods that are individual, distinct units. Examples for such products are vehicles, electronic devices, or furniture. Discrete manufacturing often involves complex quality control procedures to ensure that each unit meets its desired specifications.

The layers of automation as described are a fundamental concept in manufacturing. This model describes a structured approach to the organization of control and automation systems. However, with continuing evolution in the technological landscape, the automation model is also evolving as is modern manufacturing. This evolution emphasizes the increased integration of CPS, cloud computing, and other enabling technologies towards connected manufacturing.

2.1.1 Connected Manufacturing

The term connected manufacturing is used by us to denote the continuing transformation of manufacturing [1, 2, 3] (see previous section). Connected manufacturing emphasizes the increased strong interconnectivity of modern manufacturing systems. Interconnectivity in modern manufacturing refers to the integration of different machines, systems, and processes within a manufacturing facility, as well as the ability of these different elements to communicate with each other and share data in real-time. This integration is achieved through networked data exchange and analysis. Connecting different machines and devices allows to monitor and adjust manufacturing processes in real-time. This is also one of the findings from the German research project IUNO, which identified networked manufacturing as the key enabler of connected manufacturing [3]. With this, it is possible to em-

ploy a wide variety of different digital technologies [85]. Examples for this include machine-to-machine communication, cloud-based technologies, or machine learning. This increase in connectivity and data exchange among machines and factories is regarded by plant operators and managers with interest. The digitization of manufacturing processes allows for increased optimization of manufacturing processes, improvements in product quality, and cost reduction.

There are a variety of terms in literature referring to this process of transformation in manufacturing. Examples for such descriptive terms are *Industrie 4.0* in Germany or *Industrial Internet of Things* in the USA. Other countries also are starting to adopt or are already implementing similar manufacturing policies and strategies. Today, many countries are adopting similar initiatives that aim at coordinating the development of these new business models and of enabling more effective technologies within the domain of manufacturing [5]. In Table 2.1 on Page 44 we provide an overview compiled by us of selected initiatives. We selected initiatives from countries with a long standing tradition of industrial manufacturing dating back to the 19th century, e.g., the United Kingdom or France. Further, countries that experienced the main part of their industrialization in the 20th century, e.g., Indonesia, Brunei, or Malaysia, are included in Table 2.1. Also, initiatives that are currently in the planing phase, e.g., in Myanmar or Laos, or that are already implemented for several years, e.g., Industrie 4.0 in Germany or Society 5.0 in Japan, are contained in the table. This illustrates that governments worldwide are participating within connected manufacturing. It is, therefore, a broad international phenomena. This strong international focus of the selected nations highlights the importance of connected manufacturing in the modern world. For a comprehensive analysis of such government policies and initiatives see [5].

The government policies that often describe technological trends and provide guidelines for their implementation for national industrial manufacturing are subsidized with the term *Fourth Industrial Revolution* [90]. The Fourth Industrial Revolution is aptly named after the previous technological transitions in manufacturing [2] (see previous section) that are also labeled as revolutions. As with the previous technological transitions in manufacturing, the Fourth Industrial Revolution also builds upon the technologies of the previous time period, i.e., the Third Industrial Revolution. The widespread adoption of digital technologies and the development of the internet characterize the previous period. The Fourth Industrial Revolution now represents another significant increase within scale and speed of manufacturing.

The Fourth Industrial Revolution or connected manufacturing is characterized by the ongoing integration and combination of digital, physical, and potentially even biological systems. This is, for example, promoted by such technologies as machine learning, robotics, 3D printing, or biotechnology. Especially robotics and machine learning are of interest to us in this thesis. Thus, we discuss the integration of physical and digital worlds in the following. Especially the concept of digital twins has potential in connected manufacturing [91, 28, 40].

2.1.2 Digital Twins

In the context of engineering, the concept of *twins* or *twinning* refers to the creation of a replica of a physical object or system. This replica is designed to mimic the behavior of the physical object or system in or close to real-time. The concept of making use of twins in engineering has been explored by NASA within the Apollo space program [39]. Here, at least two identical space vehicles were constructed. One of the space vehicles was sent into space while the other vehicle remained grounded at NASA's premises. The idea behind this was to mirror the behavior of the space faring vehicle with the vehicle that remained on Earth. Within this context, the remaining vehicle was called the *twin*. The twin was then used for a variety of purposes, e.g., training of astronauts pre-flight or simulation of planned flight maneuvers and alternatives to those maneuvers. In order to make the simulated maneuvers as realistic as possible, data captured in-flight from the space faring vehicle was used if such data was available to the NASA engineers. With this, it was possible to make a prediction on the possible outcome of a planned flight maneuver before it was conducted by the vehicle in space. The predictions established with the use of twins are founded on a more realistic data basis than computer-based simulation models. This concept was later adopted by the aerospace industry with the Iron Bird. An Iron Bird was a ground-based aircraft that was also used for testing and training purposes like NASA's twin. However, due to the unavailability of some parts of the aircraft, some of the components were replaced by virtual components or computer systems. This technological progress starts the evolution of the twin concept that, therefore, originated within the aerospace industry. In the examples regarding NASA's Apollo space program and the Iron Birds, both entities involved are for the most part *physical* objects. Within this context, any kind of prototype being used to mirror the behavior of another physical object can be regarded as its

twin. In this work, however, we consider the case that the twin is a purely *virtual* replica of the physical object.

Such a virtual replica of a physical object is called a *digital twin* in current literature [40]. This concept of having a virtual replica of a real-world physical object has been explored in literature before under different terms. Examples of such terms include Computer Integrated Manufacturing (CIM) from the 1980s or Virtual Manufacturing Systems (VMS) from the 1990s [42] among others (see also Section 3.1). Both, CIM and VMS as well as other related concepts and technologies, make use of computer simulations to construct the twin. A digital twin and a simulation are both virtual representations of physical objects or systems, but there are differences between them. A simulation is a computer program that models the behavior of a physical system under specific conditions or scenarios [36]. Computer programs used for simulation employ a set of mathematical algorithms that describe the behavior of the physical system. The results generated by executing the computer program are the output of the simulation. This output can then be used to predict the behavior of the physical system or to test assumptions made about it. A specific simulation and the corresponding choice of algorithms typically depends on a specific use case or a set of use cases at hand. For example, these use cases can include designing a prototype, testing a system, or optimizing it. Opposed to simulations, a digital twin is understood to be a more accurate representation of the physical system. This improved accuracy (or *fidelity* [27], see Section 4.3) over simulations is achieved by the complexity of the models used to create the digital twin. Simulation models can range from simple to complex models depending on their intended use cases. Simulation models can also include partial models with different levels of fidelity within one simulation model. These partial models can range from high-fidelity models that simulate specific aspects of the system to low-fidelity models for simulation of larger parts of a system that are considered less relevant than others [54]. Digital twins require a high level of fidelity to accurately replicate the behavior of the physical system [28]. They typically require the integration of different models and data sources to provide a representation of the system that is as complete as possible [42]. Additionally, a digital twin is considered real-time virtual model of a physical object or system that replicates its behavior, performance, and characteristics. This can also be achieved if the digital twin consumes historical data rather than real-time data [28]. A digital twin can be created by combining real-time data from sensors on the shopfloor with other sources, models, and algorithms that simulate the physical

system's behavior [56, 57]. The high-fidelity nature of digital twin models can cover a variety of use cases that simulation-based models not suited for [28], e.g., monitoring and maintenance. Thus, the digital twin concept is an extension to traditional simulation testbeds [33].

The concept of digital twins was first introduced to academia in 2002 and 2003 by Michael Grieves during a scientific conference and a subsequent lecture on engineering at the University of Michigan [39, 42]. Grieves initially proposed the idea of digital twins as a way to improve the design and manufacturing of complex products, such as airplanes and automobiles. The original concept from Grieves involves a virtual model of a physical product. The virtual model would replicate the real world product in terms of physical characteristics and its behavior. This virtual model, referred to as a *digital twin*, could then be used to simulate the product's behavior within various use cases, identify potential issues, and optimize the design as well as the manufacturing processes required to manufacture the product. This concept introduced by Grieves is related to product lifecycle management (PLM). PLM is a process to manage the entire lifecycle of a product, from initial product conception and design to manufacturing, operation in the field, and disposal. A similar terminology more closely related ot manufacturing is project management lifecycle (PML) [92, 93], which is discussed in more detail in Chapter 4. Both, PLM and PML, require the fusion of data across different departments, e.g., from the engineering department, or the service department. This is, as the names imply, because PLM/PML are processes that accompany the product during its entire lifecycle. This was also the main point of focus for Grieves and his initial digital twin concept [39]. Within this concept, a digital twin is composed of three major components, i.e., the physical product, a virtual representation of that product (the actual digital twin), and the bi-directional data connection between both [42]. Here, the bi-directional data connection feeds data from the physical counterpart to the virtual representation in one direction. In the other direction, that is from the virtual representation to the physical counterpart, information and processes are transferred. A more thorough definition of the concept of digital twins was later provided in 2010 by John Vickers of NASA [91, 42]. Within a NASA roadmap document on modeling and simulation, a digital twin was defined as *"an integrated multi-physics, multi-scale, probabilistic simulation of a vehicle or system that uses the best available physical models, sensor updates, fleet history, etc., to mirror the life of its flying twin. It is ultra-realistic and may consider one or more important*

and interdependent vehicle systems" [94, 91]. This definition was focused more on the technical requirements of digital twins and achieved widespread appeal by the engineering community. We discuss current definitions of digital twins more thoroughly in Section 4.3. Thus, the work of Grieves and Vickers is considered to lay the groundwork for the concept of digital twins that started to gain traction from the year 2010 on [40]. The concept of digital twins is consequently being worked on since 2010 by academia as well as engineering [95, 40]. Initially, digital twins were only used by aerospace and later adopted by other domains. For manufacturing, earliest works regarding digital twins can be dated to 2013 [91, 96]. Since then, digital twins received increased attention from the manufacturing research community. For applications within the field of information security, specifically within the domain of manufacturing, see Section 3.1.

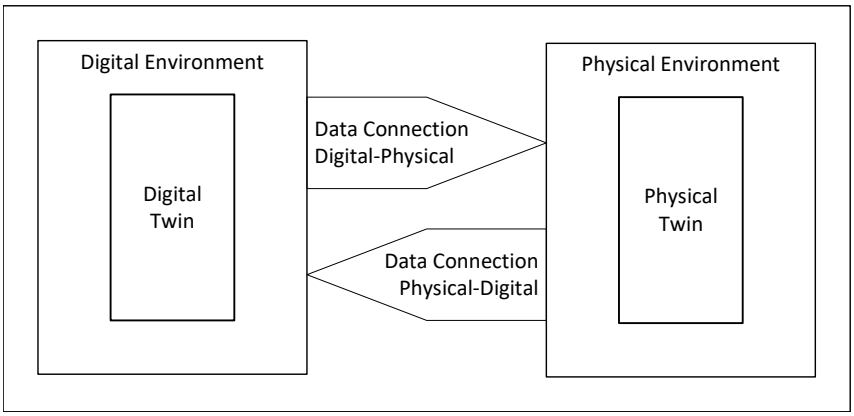


Figure 2.2: Abstract representation of the concept of a digital twin.

For the remainder of this section, we introduce a simple conceptual model of a digital twin. Here, we discuss its main components according to Grieves and add recent terminology to that model. The abstract model of a digital twin is seen in Figure 2.2. The three major components of a digital twin that have been identified by Grieves are seen in the figure [39, 42]. The physical product is located on the right of Figure 2.2, whereas the virtual representation is located on the left. They are named physical twin and digital twin respectively. Both of them are enclosed in their respective environments: a physical environment for the physical twin and a digital environment for the digital twin. The environment represent the world each entity is located in. For the physical twin, this is the real world with its invariant laws of physics and its business and organization processes. For the digital twin,

the digital twin represents the computer system(s) where the twin is executed and stored on. Both twins and their environments are connected via a bi-directional data connection as indicated by the arrows in the middle of Figure 2.2. Here, the direction for the flow of data is shown separately; a connection from the digital to the physical environment (labeled as 'Data Connection Digital-Physical') and a data connection from the physical environment (labeled as 'Data Connection Physical-Digital'). The Data Connection Digital-Physical is sometimes referred to as *digital trigger* whereas the Data Connection Physical-Digital is sometimes referred to as *digital shadow* [40]. The discussed abstract concept of a digital twin is further refined in Chapter 4.

2.1.3 Threats in Modern Manufacturing

Digital technologies implemented in modern manufacturing environments allow for increased productivity by introducing automation (see previous sections) but at the same time introduce various new attack vectors [97]. In this section, we illustrate the threat landscape modern manufacturing environments experience. We illustrate general trends and discuss key attacks in detail. These examples provide an insight into the common attack patterns (see Section 4.2) and how these exploit the major challenges for information security in manufacturing [15] (see Section 1.1 and Section 1.2) We start our discussion with some of the first cyber attacks that occurred in manufacturing and continue on to current trends. For a more detailed discussion of attackers, their capabilities, and motivations see Section 4.1.

Attacks on modern industrial equipment are often presumed to have started in 1982 with an supposed cyber attack on a trans-Siberian pipeline in the former Soviet Union (nowadays Russia) [12, 98]. A "logic bomb"[1] is claimed to have caused an explosion of the pipeline resulting in tremendous damage. Allegedly, state actors are considered to be behind this act of sabotage. These story, however, seems implausible to us and might not have taken place. Literature sources describing this attack with a sufficient level of detail are not available. To the best of our knowledge, the original claim of this supposed attack was from the autobiography of former Secretary of the Air Force Thomas Reed [99]. Autobiographies in general may be influenced by the author's opinions. Furthermore, [99] appears to be the single source for this claim. Historically speaking, the alleged act of sabotage took place in the backdrop of the 1981-82 NATO Siberian gas pipeline dispute [100, 101]. Sources

[1] Logic bombs are computer programs that start malicious activities after a certain logical condition is met.

examining this conflict from a legal [100] and political [101] perspective make no mentions of explosions or any (attempted) acts of sabotage for that matter. Despite these lack of sources, the alleged attack on a pipeline is often cited as the first cyber attack on modern industrial equipment, e.g., in [12, 98] and by other authors.

Another often cited incident that is documented rather extensively by law enforcement is the Maroochy Shire incident that occurred in Australia in the year 2000 [102, 13, 18]. Here, a former contractor of a local water treatment company accessed 142 water pumps [103] remotely with a laptop and a specialized industrial radio transmitter. Over a three month period, the attacker drove around the Maroochy Shire area with his car and used the stolen OT hardware and software and insider knowledge to activate water pumps and to release untreated sewage water. The motive was cited at the court trials with revenge, as the attacker failed to secure employment with the contracting company. Therefore, the attack is often cited as a prime example for a "disgruntled employee" [104, 105, 19] (see also Section 4.1.1). The attack resulted in heavy environmental damage as 800 kiloliters [16] of raw sewage water were released into waterways and local parks. The attacker used the login credentials known by him to access the industrial systems at the site. Passwords were not changed on a regular basis at the company. Furthermore, the radio communication was not secured, and no access log were recorded by the OT systems. Further examples for such locally isolated attacks are reported in literature [13, 98].

In recent years, a trend towards the establishment of threat campaigns can be recognized [13, 12, 106, 68]. The starting point of this trend can be set in 2003 with the *Slammer* worm. That computer worm was capable of affecting nuclear power stations. While such attacks are considered to have no specific target, targeted attacks continued in the recent years, e.g., the prominent *Stuxent* attack on nuclear facilities located in Iran (discovered in 2009) [23, 107, 108] (see Section 1.1). The Stuxnet malware uses a multi-staged attack vector with a broad attack surface. It infiltrates probably by breaching the air gap via infected USB devices, then propagates within the IT network until reaching its targets within the (also separated) OT network. Stuxnet showed high complexity and sophistication in the attack by exploiting four zero day vulnerabilities, using presumably stolen vendor certificates for device driver manipulation, and specific domain knowledge about its target. Furthermore, the malware evolved considerably as patched vulnerabilities and revoked certificates were replaced by new exploits and certificates. It used, among others, malware code specifically targeting the exact types of PLCs used in the plant. It

used them for further propagation within the network and for infection of OT communication modules. Due to the stealthy nature of Stuxnet, long term infiltration with wide reaching physical destruction of OT equipment occurred. Given the sophistication of the attack, a general defense against such attacks maybe hard to achieve. One possible countermeasure can be intrusion detection systems (IDSs) but there proper realization comes with its own challenges [109].

More recent examples include the series attacks on power systems in Ukraine between the years 2014 and 2016 [13, 103, 68]. Several major cyber attacks on critical energy infrastructure in Ukraine have been reported. The energy sector shares similarities with the manufacturing domain, down to the industrial equipment used within the networks. At the end of the year 2015, a large-scale power outage occurred in the city of Kiev that affected around 225.000 people and lasted for several hours. The cause was a piece of malware later named *BlackEnergy*. The attackers used BlackEnergy to initially compromise the energy company's IT network and then manually moved from there into the control network responsible for the power supply. This ultimately enabled them to disrupt substations, disconnect them from the power grid, and trigger widespread power outages. One year later, in 2016, another large-scale power outage occurred. This time, an improved version of the previously used malware was deployed called *Industroyer*. This malware is considered to be the second malware after Stuxnet specifically developed for causing disruption of a physical industrial process [103]. It features a modular design with high potential for reusability. The malware was able to access at least four different OT protocols and displayed different attack capabilities. Apparently, it was specifically developed for attacks on the energy sector designed to interrupt the energy supply. Known vulnerabilities in the outdated Windows XP SP3 operating system used by the affected energy provider served as entry point.

Manufacturing systems are also victim to the most recent trends in cyber attacks. This trend is summarized by the term ransomware [110, 25]. It describes a type of malware that infiltrates a target system and denies the user access to the data stored on the device. This is achieved by encrypting data with a public key, where the corresponding private key is stored on an external server under the control of the attacker. For decryption of the data, a ransom payment is demanded. A public incident where a manufacturing system was the victim of ransomware is documented by the Norsk Hydro incident in 2019 [18, 111]. The Norwegian aluminum manufacturing company Norsk Hydro was affected by the ransomware *LockerGoga*. The initial

compromisation of the company's network probably occured via its IT systems and propagated from there to the OT network. Most likely, the IT and OT systems used the same infected Windows Active Directory server. This increased the attack surface of the malware towards the OT network. The subsequent encryption of OT systems lead to a significant impact on Norsk Hydro's manufacturing operations, as production facilities were disrupted since the devices could not be operated anymore. As reaction to the attack, all plants were disconnected from the network and all IT systems were shutdown. Manual operation of the manufacturing was started to continue production at least partially. Plants not designed for manual operation, as a consequence, were not able to continue operation. The financial damage for the company is estimated with 41-46 million Euro due to the loss of availability in manufacturing systems. Norsk Hydro quickly responded to the attack by alerting its shareholders and implementing its incident response plan. They worked with external information security experts and law enforcement agencies while maintaining a constant public information policy. The company decided against paying the ransom, but instead restored its systems from backups, which was a time-consuming operation.

The transparency demonstrated by Norsk Hydro is rare when examining the total number of attacks on manufacturing companies. The number of unreported attacks is likely high as many companies may fear a loss in reputation, which could translate to a loss in revenue due to a reduced number of orders. This can make it difficult to accurately assess the true extent of the current threat of cyber attacks to manufacturing. However, the number of cyber attacks appears to constantly increase as manufacturing is becoming a more attractive target for attackers.

Table 2.1: International connected manufacturing initiatives (sorted alphabetically by country name) [2, 86, 87, 88, 89].

Country	**Name of government initiatives**
Australia	Australia's Tech Future
Brunei	Brunei Vision 2035
Bulgaria	Kontseptsia Industria 4.0
Cambodia	Cambodian ICT Masterplan 2020
China	Made in China 2025
Czech Republic	Průmysl 4.0
France	Industri du Futur
Germany	Industrie 4.0
Hungary	Irinyi Plan
India	Digital India
Indonesia	Making Indonesia 4.0
Ireland	Ireland's Industry 4.0 Strategy 2020-2025
Italy	Fabrica Intelligente
Japan	Society 5.0
Laos	*Currently in planing phase*
Lithuania	Pramonė 4.0
Malaysia	National Policy on Industry 4.0
Mexico	Crafting the Future
Myanmar	*Currently in planing phase*
Netherlands	Smart Industry
New Zealand	Building a digital nation
Philippines	Inclusive Innovation Industrial Strategy
Poland	Future Industry Platform
Portugal	Indùstria 4.0
Romania	National Strategy for Romanian Digital Agenda 2020
Singapore	Singapore's Industry 4.0
Slovakia	Smart Industry Platform
Slovenia	Slovenian Industrial Policy 2013
South Africa	*Currently in planing phase*
South Korea	Manufacturing Innovation 3.0
Spain	Industria Conectada 4.0
Sweden	Produktion 2030
Thailand	Thailand 4.0
UK	High Value Manufacturing Catapult
USA	Industrial Internet of Things, Advanced Manufacturing
Vietnam	Strengthening Vietnam's capacity to leverage the 4th Industrial Revolution

2.2 UML

The Unified Modeling Language (UML) is a standard notation used for modeling object-oriented software systems [74, 112]. The UML standard contains a set of diagrams that can be used to represent different aspects of a system. These aspects include its structure, behavior, and its interactions. In this section, we provide a brief overview on UML and its role in computer security. Special focus is given to UML state machine diagrams that are discussed separately in Section 2.2.1. For our discussion, we focus on the most recent, formally adopted version of the UML specification, i.e., Version 2.5.1 as of December 2017 [74].

UML was first introduced in the year 1996 and is based on previous work in the field of object-oriented programming [113]. The initial goal of UML is to provide a standardized notation for modeling object-oriented software systems. UML evolved to become a more widely used standard for software modeling across different industries. This development resulted in the publication of an approved international standard by the International Organization for Standardization (ISO) in 2005 [114]. UML provides a set of diagrams that can be used to represent different aspects of a software system. These diagrams can be broadly categorized into two types: structural diagrams and behavior diagrams. Structural diagrams represent the static aspects of a software system such as objects, classes, components, and relationships between them. Behavior diagrams, on the other hand, represent the dynamic aspects of a software system with the interactions between objects and the flow of control within the system. Each of these two types of diagram types further contains a number of sub-types. Examples of structural diagram sub-types include class diagrams or component diagrams. Class diagrams are used to represent the classes, interfaces, and their relationships within a software system. Component diagrams also include interfaces but are more focused on representing the physical components of a system. This does also, again, include the relationships between the physical components. Examples for behavior diagram sub-types are use case diagrams, sequence diagrams, or state machine diagrams. Note that there are more diagrams specified in UML than the examples we provided above. The examples given above focus on some of the in our opinion most commonly used UML structural and behavior diagrams. Use case diagrams are used to represent the different use cases or scenarios that can be realized with a system. State machine diagrams are used to represent the behavior of a system (see Section 2.2.1).

UML is intended as a flexible notation and is, thus, not limited to its usage in software engineering. It can be used to model different types of systems, including software systems, hardware systems, business systems, or technical systems. In the field of computer security, UML is used for modeling different aspects of security systems [115, 116, 117, 118] (see also Section 3.1.1). These include modeling security policies, modeling security mechanisms, and modeling security requirements. One application of UML in computer security is modeling security systems. UML can also used to model security policies, which are the rules that govern access to a system. Security policies can be modeled, e.g., by using use case diagrams. In this context, a use case diagram can be used to show the different scenarios that can occur when people or other systems interact with the security control. From there, security requirements can be be derived from the use case diagram and used within the design of the security control. UML can also used to model security mechanisms, which are the tools and techniques used to enforce security policies. Security mechanisms can be modeled using activity diagrams, which represent the workflow of the security system. State machine diagrams can also be used in this context for the representation of different states that a security system can be in. From a high-level perspective, two of such states can be, for example, states labeled '*secure*' or '*compromised*' (see Section 5.1 for a more detailed discussion and illustrative examples). In the following, we focus on state machine diagrams.

2.2.1 State Machine Diagrams

As discussed in the previous section, UML state machine diagrams are a sub-type of UML behavior diagrams. The state machine diagrams main purpose is to represent the behavior of *objects* in a system over time. Objects are not explicitly represented visually within an UML state machine chart. The behavior of an object here is represented as a set of states and the transitions between them. The transitions between the states are triggered by events .The system reacts to the event when it takes place. The reaction of the system depends on the system's current internal state and the event's type. A reaction to an event by the system can include an update to its internal state as well as a transition into another state. The pattern of states and triggered state transitions as described above can be formally represented by the notion of a Finite State Machine (FSM). A FSM also makes the handling of an event explicitly dependent on both the type of event and the current system state is introduced when handling the event. Thus, one of the defining characteristics of

UML state machines is their event-driven nature. This event-driven approach is also similar to that of a FSM. Here, the system is represented as a set of states and transitions that occur between those states. Each of the transitions is triggered either by an event or by another condition. A FSM is typically used to model systems with a fixed set of states and transitions. UML state machines, however, can model systems with dynamic behavior that are subject to change. This is because UML state machines can have more complex transitions, such as those triggered by a combination of events or conditions. Furthermore, they can also include nested states (see below) in which one state contains another state machine.

UML state machine diagrams are a type of behavior diagrams that model the behavior of a system as a set of states and transitions between the states. Each state represents a condition or situation in which the object can exist. The transitions represent the possible situations in which the modeled object can move between states. The movement is triggered by some event or condition. State machines are commonly used to model complex systems with multiple states and transitions. They can be used to model a wide range of systems, including hardware systems, control systems, communication systems, and software systems. Note that these systems given are not necessarily limited to a software system and can also include models complex real-word systems such as manufacturing sites or industrial plants.

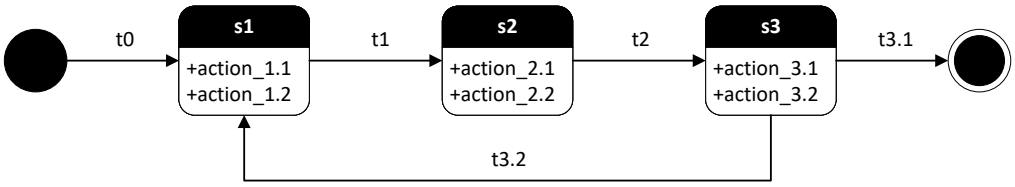


Figure 2.3: Basic notation for UML state machines with states, events, transitions, and actions.

The basic UML state machine diagram notation consists of a set of states, transitions, events, and actions. This is illustrated by the state machine depicted in Figure 2.3. Rectangles with rounded corners represent *states*. A state is a condition or situation in which the object can exist. States are represented as rectangles with rounded corners and are labeled with a name that describes the condition or situation. This is seen in in Figure 2.3 by the states named *s1*, *s2*, and *s3*. States represented by rectangles make up the bulk of states in a state machine diagram;

there can be as many rectangular states in a diagram as required. Apart from the rectangular states, two special states are part of every UML state machine diagram. These special states are called the *entry state* and *exit state*. An entry state is a state that is entered when an object first enters the state machine. This state is represented as a solid circle with an incoming transition and is sometimes labeled with the keyword '*entry*' (see left hand side of Figure 2.3; the label of the entry state is omitted for simplicity). The entry state can be used to define the initial state of the object and all the actions that need to be performed when the object enters the state machine. For example, in a state machine that models a vending machine, the entry state could be used to define that the machine enters into the '*idle*' state when it is first turned on and that it should display a message containing user instructions on the machine's screen. An exit state, on the other hand, is a state that represents the end of the object's behavior in the state machine. This state is represented as a solid circle with no outgoing transitions and is sometimes labeled with the keyword '*final*' (see right hand side of Figure 2.3, label also omitted). The final state is typically used to indicate that the object has completed its task or has reached the end of its lifecycle. For example, in a state machine that models a vending machine, the final state could be used to indicate that the product has been dispensed successfully and that the machine is now idle again. Each UML state machine diagram consists of at least one rectangular state, one entry state, and one final state.

All states are connected by *transitions*. A transition is a change of state in the modeled object. Transitions are represented as arrows between states and are labeled with the event that triggers the transition. This is seen in in Figure 2.3 by the arrows labeled *t0*, *t1*, *t2*, *t3.1*, and *t3.2*. As seen by state *s1* and its ingoing arrows with the event labels *t0* and *t3.2*, a state can have multiple ingoing transitions. Also, as seen by state *s3* and its outgoing transitions with the event labels *t3.1* and *t3.2*, a state can have have more than one outgoing transition as well. A transition between two states is, as already indicated, triggered by an *event*. An event is a trigger that causes the object to move from one state to another. The labels on the transitions represent these events and can be any type of signal or condition that causes the transition to occur. The final basic notation introduced by Figure 2.3 is that of an *action*. An action is a behavior or effect that occurs when a transition is executed. Actions can be associated with states or transitions and are represented as behavior that occurs when the transition is executed. Figure 2.3 so far shows only actions that are associated with states. They are depicted by a plus sign ('+') as prefix and

a textual descriptor. For example, for state *s1*, there are two associated actions, i.e., *action_1.1* and *action_1.2*. The same is true for the other two states, *s2* and *s3*, which also have two associated actions each. This concludes the basic notations of UML state machine diagrams, i.e., states, transitions, events, and actions associated to states. In the following, more notations specifically for actions as well advanced concepts of UML state machine diagrams are discussed.

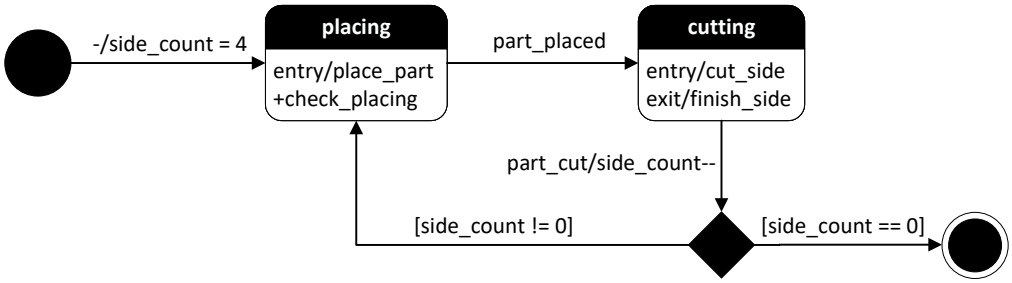


Figure 2.4: Extended notation of UML state machine with states, events, transitions, and actions.

The basic notation for UML state machine charts as discussed above can be extended upon. A selection of these extensions that are used in our work is shown in Figure 2.4. It shows an application of modeling complex systems with UML state machine charts. The system shown is a part of a manufacturing process where a piece of wood is processed by two machines. The piece of wood is the object transitioning through the model. Contained within the model is a placing machine (modeled by the state *placing*), which is responsible for turning the rectangular piece of wood in 90 degree angles. From there, it is transported to the *cutting* machine or state. Here, one side of the object is cut off in order to achieve the desired product geometry. Each side of the object needs to be processed, which means that a total of four placing and cutting actions need to be performed. Here, extended UML notations can be used to model this industrial process. Actions that are associated with states can be further detailed by defining *entry actions* and *exit actions*. An entry action is an action that is executed when an object enters a state. It is specified using the keyword '*entry*' followed by the respective actions. In Figure 2.4, this is shown by the *entry/place_part* action of the *placing* state. When the object enters this state, the piece of wood is turned. Similarly, it is cut when entering the state *cutting* as indicated by the *entry/cutting action*. An exit action, on the other hand, is an ac-

tion that is executed when an object exits a state. It is specified using the keyword '*exit*' followed the actions to be executed on the object's exit. In Figure 2.4, this is shown by the *exit/finish_side* action of the *cutting* state. By that action, some finalizing procedures are performed on the object, e.g., cleaning. Actions can also be associated to transitions or events. In the example given by Figure 2.4, this notation is used to control the amount of time the piece of wood is cut within the *cutting* state. As a rectangular piece of wood has a total of four sides that require cutting in our example, a controlling variable entitled *side_count* is introduced to the model. It is manipulated within the model at two specific occurrences: once when the object enters the model and once for every time a cutting operation is performed, i.e., when it exits the *cutting* state. In UML state machine notation, this is shown as an extension to an event similar to the entry action or exit action within a state. The action to be performed on transition is added to the event as seen by the associated action *side_count--* to the event *part_cut*.

Continuing with our example, we illustrate a way to model the flow of control in UML state machine diagrams. The variable *side_count* is manipulated at two instances within the industrial process modeled (see Figure 2.4). A choice is made in order to decide whether all sides are cut. This is achieved with a *choice* pseudostate that allows us to specify a decision point in the state machine. A choice is denoted as a black diamond shape (see middle in the bottom half of Figure 2.4). A choice is used to represent a branching point in the state machine, where the behavior of the object may depend on some condition or event. A choice is similar to an action in that it specifies a behavior that is executed by the object in response to some event. However, while an action typically represents a single step in the object's behavior, a choice represents a decision point where the object's behavior can branch in different directions depending on some condition. In our example, the variable *side_count* is checked at the choice. If all sides of the object are processed that state machine terminates by entering its exit state; otherwise, execution is continued by entering the *placing* state again. This check is performed by *guard* conditions. A guard condition is a condition that must be satisfied so a transition can occur between the states where the guard is located. Guards are represented as a Boolean expression. If that expression is true, the transition occurs, otherwise, no transition occurs. They are enclosed in square brackets for better visual distinguishing them from actions. In Figure 2.4, the conditions *side_count != 0* and *side_count==0* are evaluated to control the flow of the object through the state machine diagram.

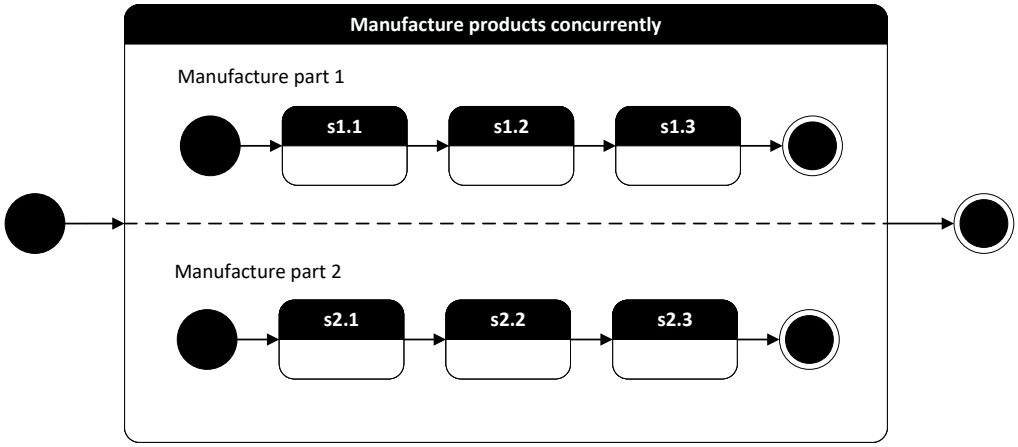


Figure 2.5: UML state machine diagram of concurrent behavior with orthogonal regions within an abstract manufacturing system.

Modeling control flow is important when using UML state machines diagrams for the modeling of complex systems. One property for such systems is the presence of concurrent behavior meaning the parallel execution of tasks. This implies that more than one object is processed by the state machine diagram. For UML state machine diagrams, concurrent behavior is represented by *orthogonal regions*. An orthogonal region is a set of states and transitions that can execute concurrently with other orthogonal regions in the same state machine. Each orthogonal region has its own set of states and transitions and can have its own initial state. An abstract UML state machine diagram modeling concurrent behavior with orthogonal regions is shown by Figure 2.5. Here, a simple setup of two concurrent production lines is shown. Orthogonal regions are represented in UML state machines as rectangular boxes stacked vertically or horizontally, separated by thin dashed lines each. Each box contains its own set of states and transitions, and is labeled with a name that describes the concurrent behavior it represents (*Manufacture part 1* and *Manufacture part 2* in Figure 2.5). Concurrent execution in orthogonal regions is coordinated by events and signals. Events can be directed to a specific region or broadcast to all regions, while signals are broadcast to all regions. Transitions in each orthogonal region can be triggered by events or signals received by that region, and can also send events or signals to other regions or to the outside model. Orthogonal regions allow state machines to model complex systems with multiple concurrent behaviors, such as manufacturing plants that produce multiple parts in parallel. By dividing

the behavior of the system into separate orthogonal regions, the model can be easier to understand and modify. Also, the model can better capture the real-world behavior of the system being modeled if it contains concurrent behavior.

One of the challenges in using state machine diagrams is to ensure that the diagram accurately models the behavior of the system. Depending on the complexity of the system, this can require a deep understanding of the system so all possible states and transitions can be included within the model. For complex systems, a UML state machine chart can result in a large model that may be difficult to adjust and maintain. For this, UML state machine diagrams introduce the semantic of *hierarchically nested states*. Hierarchically nested states are used in UML state machine diagrams to model complex behaviors that can be broken down into smaller, more manageable parts. With this approach, a state can include several other states, so-called sub-states. The containing state is referred to as super-state and together with its sub-states a hierarchical structure is formed. These sub-states can also contain their own sub-states, which then results in a nested structure. Hierarchically nested states are represented in UML state machine diagrams as states within other states. The sub-states are connected to the containing state with an arrow to the containing state. A sub-state can have its own entry and exit actions, and can also have transitions that lead to other sub-states or to the containing state. For example, the State *s1.1* in Figure 2.5 can be such a sub-state where a more detailed sequence of events is executed once the state machine enters this state. To continue with this example, the manufacturing procedure described by Figure 2.4 can be executed when the object enters State *s1.1*. Once the sequence of events of Figure 2.4 is finished, the object returns from the sub-state and continues traversal of the state machine shown in Figure 2.5 to enter State *s1.2*, where another sub-state may be entered.

This concludes our discussion of the semantics UML state machine charts. Note that additional notations are specified by UML [74]. These additional notations are, however, not further discussed as they are not included in our modeling approach (see Chapter 5).

2.3 Privacy

In this section we give a succinct background on privacy, particularly the field of data privacy. Privacy in general is a term that is, similar to digital twins (see Section 2.1.2), often used in literature but a generally accepted definition appears to be difficult to reach [119]. In general, privacy can be described as the state of being free from unwanted or unauthorized intrusion, observation, or surveillance. Our focus in this thesis is on the privacy of data and the means by which these privacy can be established and protected. We discuss the historical development of these privacy-preserving concepts and technologies and conclude with a discussion on the most recent of these technologies.

The perspective of privacy in discussions can vary tremendously, which is why privacy is often considered a term difficult to define. This is due to the multidimensional and context-dependent nature of privacy. Several aspects and factors can contribute to the complexity of defining privacy such as cultural differences or the context of the discussion. Privacy can encompass the overall concept of individuals to control their personal information. It can, furthermore, encompass various aspects. For example, the authors of [120] list several classifications for private information. According to [120], privacy includes data and image privacy, which involves protecting personal data from unauthorized access and allowing individuals to control its use. The privacy of the person involves keeping their physical properties and characteristics private, such as genetic codes. Privacy of behavior, decision, and action pertains to the right to keep actions and sensitive information, like sexual preferences and political activities, private. Privacy of location and space involves the freedom to move around in public spaces without being tracked or identified. Privacy of communication ensures that individuals can avoid interception of their communications. Privacy of association allows individuals to associate freely without being monitored. Lastly, privacy of thoughts and feelings entails the right not to have one's thoughts and emotions intruded upon. As can be seen from these aspects and categories of privacy, the general term of privacy encompasses all aspects of an individual's private life.

A more narrow and concise definition for certain aspects of privacy is data privacy. Data privacy, on the other hand, specifically focuses on the protection of personal information, which is referred to as personally identifiable information (PII) in literature [121]. Data privacy is discussed by academia within the context of data

collection, storage, processing, and sharing. Data privacy, thus, is concerned with the protection of sensitive data and information from unauthorized access and disclosure. This can include information security controls such as encryption and access control [122]. However, there is also ample research available within the field of protecting privacy that is stored in external databases where third parties or even the general public has access to [121, 123]. This field of research is privacy-preserving data publishing (PPDP). Within PPDP, a wide variety of privacy-preserving data publishing techniques are discussed. We introduce the basic techniques used within PPDP and give illustrative examples for the major techniques involved within it.

One of the first privacy-enhancing techniques developed is pseudonymization of data records is. Pseudonymization is a simple and straightforward approach that involves replacing PII in a dataset with pseudonyms. It aims to prevent the direct identification of individuals in data sets by altering explicitly identifying attributes, thus, generating pseudonyms. These pseudonyms can be altered values of the original data, completely random generated data, or encrypted versions of the original data. It is important to note that pseudonymization by itself is an insufficient approach for protecting privacy. Simply replacing identifying attributes with different values alone does not provide adequate protection of privacy. The resulting pseudomyzed data set might still be unique and an adversary can still identify individual records. This is illustrated by a number of recorded data breaches such as the Gravatar incident or the New York City Cab Driver incident to name only a few [124]. Similar to pseudonymization is another approach towards PPDP called de-identification [121, 123]. Here, PII are deleted from the the data record instead of being replaced with another value. This approach to PPDP is also meant to prevent the identification of individuals in (published) datasets, as explicitly identifying attributes are removed completely from the final record stored within the database. While this prevents some of the attacks possible on pseudonymized data, other attacks are still possible and achievable with relative ease for an adversary. An adversary can link data in the de-identified dataset with other datasets or with background knowledge gathered from other sources. This way, it is still possible to re-identify an individuals personal records from a de-identified dataset, even when such information like name, address, or phone number are not present within the data set. Other information in the data set like ZIP code, date of birth, place of birth, gender, etc. can be sufficient for re-identification. This has been prominently illustrated with the case of the Governor of Massachusetts in 2001 [125]. Here, in-

formation from a publicly accessible medical data set with de-identified records was matched with that of a purchasable voter list in order to match records from the governor of Massachusetts at the time and, thus, disclose their medical history. Against the background of successful attacks on privacy like this and of legislative initiatives on medical patients' privacy, researchers have started to develop improved methods for ensuring data privacy. De-identification may still be used in privacy-preserving applications but it is usually only considered as an initial step before more efficient privacy-preserving techniques are applied.

2.3.1 Privacy Models

These privacy-preserving techniques that are nowadays tasked with performing most privacy-preserving tasks in PPDP and other areas are referred to as *anonymization*. For us, anonymization describes a range different approaches and techniques. They are applied to a data set in order to introduce a privacy guarantee to the data set. This privacy guarantee refers to the assurance of the implemented anonymization technique to protect the privacy of data sets to a certain degree. This degree of privacy protection offered by the anonymization approaches is usually mathematically defined and can be assessed [121, 126]. One goal of applying anonymization is to preserve interesting information in the anonymized data sets for subsequent data analysis. All attributes of a dataset, in principle, can contain valuable information. What attributes of a data set are considered valuable depends strongly on the use cases or research interests at hand [46, 47]. In any case, those attributes are required to remain useful for the data analyst after the anonymization process. That is, the data set needs to provide a certain degree of utility. Anonymization is applied to a variety of contexts that include data sharing, or research such as with medical studies. Therefore, different anonymization techniques have been developed in order to address a variety of use cases. Some of these use cases make it necessary to address complex requirements for sufficient privacy guarantees to be met.

Therefore, literature tends to refer to more sophisticated anonymization techniques as privacy models. A privacy model refers to a conceptual framework or a set of rules that define the requirements and mechanisms for ensuring privacy and for delivering a certain privacy guarantee. We provide an overview about some of the most common privacy models. For a comprehensive overview, we refer the interested reader to surveys published about this subject, e.g., those of [121, 123]. The first privacy model we discuss is the privacy model k-anonymity [127]. k-anonymity is

named after its primary variable, k, used within its mathematical model for defining the concept of this privacy model. As k-anonymity can be considered the first of the modern privacy models, this sparked a general trend in PPDP research, where privacy models are often named after the variables' names used in definition of the respective privacy model. k-anonymity now aims to protect individuals' identities in a data set by ensuring that each individual's information is indistinguishable from at least $k - 1$ other individuals. In other words, the data set to be published is anonymized in such a manner that the minimum amount of occurrences of identical identifiers is at least k. A data set that achieves this definition is said to be k-anonymous. Let us consider an example for this. Assume we have a data set that contains medical data records. Each attribute within each record is divided in information by which an individual is identifiable and such attributes that are of research interest to medical professionals. The identifiable attributes in our example are the individuals' age, gender, and ZIP code. If there is only one individual in a certain ZIP code area of a certain age and gender, than that person can be identified by someone who might have knowledge of that person. Then, that attacker knows the medical history of that person which is contained within the interesting attributes of the data set. The set of information that makes an individual identifiable as described is also referred to as a quasi-identifier (QID) in contrast to a direct identifier (an ID). k-anonymity now aims to alter at least k other data records in such a manner that at least k other data records share the same QID. This can be achieved by a variety of algorithms [121]. A common method for achieving k-anonymity is generalization of the attributes within the QID. For example, the specific age contained in our data set can be replaced by an age span, e.g., replacing all ages between 31 and 39 by that time span, i.e., 31 - 39. This way, the sensitive information belonging to a QID is protected by a statistical measure. Say that $k = 5$ in our example, then an attacker would need to guess to which person the sensitive information belongs. The odds of success for an attacker are, in the case of $k = 5$, 20% for guessing the correct individual. A higher k offers a better privacy, however, may reduce the utility of the data as attributes that are generalized in a too abstract manner may not be usable anymore for research purposes. Therefore, privacy needs to balanced with the utility required, which is in general a common theme in the field of privacy models and PPDP [121, 47]. k-anonymity prevents the re-identification of specific individuals within a dataset but is not effective against other attacks on privacy. Also, as k-anonymity is developed with medical records and data sets in

mind, it is not applicable to many other types of data, e.g., time series data. For that reason, a number of different privacy models have been develop that offer some protection against certain attacks and are applicable to certain types of data sets. They are named in accordance to the naming convention inadvertently introduced by [127] with their privacy model k-anonymity. Examples for such privacy models are k^m-anonymity, l-diversity, t-closeness, or (h, k, p)-coherence [121].

2.3.1.1 Differential Privacy

We now discuss another privacy model in more depth that offers several benefits over the privacy models mentioned above. That privacy model is referred to as ϵ-differential privacy or, in more recent publications, just as differential privacy (DP) [128]. differential privacy is a privacy model that is built upon previous research in the field of information recovery [129]. More specifically, differential privacy aims at formalizing the concept behind the Fundamental Law of Information Recovery [130]. The Fundamental Law of Information Recovery is a concept that examines the trade-off between privacy and utility in data analysis as mentioned above. Roughly, the concept states that with enough queries into a data set or database, at least some private information from the database can be retrieved. This suggests that as more information is released or shared, there is an increased risk of privacy breaches or re-identification. Differential privacy directly addresses this issue by introducing controlled noise into the data. By doing so, a differentially private data set limits the amount of information that can be extracted from it. Thereby, it reduces the overall privacy risk while some level of utility is maintained. In 2006, Cynthia Dwork et al. published their research on the amount of noise that is required to be added to a data set in order to achieve reasonable privacy guarantees [126]. Also, they proposed a generalized mechanism and formalized the concept of differential privacy. The privacy model of differential privacy is targeted on privacy preservation when performing data analysis or releasing aggregate statistics [131]. Let us consider another example to highlight the idea of differential privacy. Assume a dataset containing information about individuals' ages. Without a privacy model, releasing the average age of the individuals in the dataset does provide information about each individual's age. When applying differential privacy, noise is added intentionally to the dataset and, consequently, also within the average computed from the perturbed data. Suppose the real average age of the of the individuals in the data set is 38 years. After differential privacy is applied, the query response might be slightly

perturbed, which could result in an average age of 39.5 years. This randomization already can make it challenging to determine the real ages of individuals within the dataset.

So far, we discussed differential privacy in the context of the Fundamental Law of Information Recovery. Differential privacy can, however, also be regarded to implement the principle of plausible deniability [126, 132, 130]. Plausible deniability refers to the ability of an individual to deny knowledge or involvement in a particular action or event. Typically, this implies that the individual's knowledge or involvement cannot be easily proven. The term of plausible deniability is often used in a political context. In the context of PPDP, plausible deniability refers to the ability of an individual to deny having a data record of itself included in a given data set. Also, the data set owner, e.g., a medical or another research institution, can claim that individual data points or sensitive information cannot be accurately attributed to specific individuals. This protects the privacy of an individual by reducing the risk of re-identification. This illustrated by the formal definition of differential privacy that is discussed in depth in Section 7.3.1. The possible benefits of having plausible deniability in a statistical data set can be illustrated by the following coin tossing example. Similar methods like in this example are used within the social sciences when questioning individuals about socially unaccepted and potentially illegal activities such as sexual or criminal behavior. In order to protect their participants, noise is introduced during collection of the data set. This is achieved by letting the the participant flip a coin. Depending on the outcome of the coin toss, which is either heads or tails, the participant either answers the questions it is being asked truthfully (in case the coin toss returns heads) or tosses another coin (in case of tails). If the second coin toss returns heads, the participant answers the question truthfully. However, if the second coin toss returns tails, then the participant is expected to give an incorrect response to the question. With this simple method, a degree of noise is introduced into the data set. For the participant, it offers the benefit of plausible deniability as the coin tosses are not recorded or observed and the participant can simply state that both its coin tosses turned out to be tails (or any other combination of coin toss results). This way, the participant is protected from legal prosecution or social stigmata. The researchers, on the other hand, can still make use of the data as the distribution of the noise is known and can be compensated by statistical methods. By adjusting the amount of noise added, e.g., by introducing additional coin tosses, the trade-off between privacy and utility can be

controlled. More noise, i.e., more coin tosses, lead to stronger privacy but potentially less accurate and useful data, while less noise improves utility but weakens privacy guarantees. Note, that the above discussion on plausible deniability can also apply to other privacy models, however, the means and the type of noise or randomization introduced by the respective privacy then varies, as may any privacy guarantees given by the privacy model.

To summarize, the core idea behind differential privacy is to introduce controlled randomness, i.e., the noise, into the data set in order to prevent unauthorized extraction of sensitive information from the data. By adding noise, differential privacy makes it difficult for an attacker to distinguish the contribution of a specific individual within the dataset.

2.3.2 Privacy Enhancing Technologies

In this section we give an overview about privacy enhancing technologies (PETs), which are also referred to as privacy-preserving technologies in literature [75]. For the discussion within the context of this thesis, we exclusively use the term PET when discussing the following type of technologies. PETs refer to a collection of IT measures that enforce or enhance the privacy of sensitive data [133]. Sensitive data processed by systems located in the target domain of a PET is often discussed within the context of personal data related to individuals. For our context, also data collected from tooling machines within manufacturing is considered [134, 135] (see Section 3.2.1). Privacy protection of sensitive data by PETs is typically achieved by minimizing the collection or by minimizing the fidelity of the collected sensitive information [136, 121]. This is related to our discussion on privacy models in Section 2.3.1. In fact, privacy models and PETs are closely interrelated within the design of privacy controls [135, 75, 137]. As mentioned above, a privacy model is a theoretical framework that follows some sort of mathematically defined formalism [126, 121]. This theoretical framework provided by privacy models can serve as design guideline for PETs, as the privacy model outlines the main objectives and requirements that a PET needs to meet to ensure that privacy is guaranteed. PETs implement the principles of the privacy model while considering current technologies and usability of the application. A privacy model, furthermore, can be used to evaluate the effectiveness of PETs. By comparing the performance of a PET against the theoretical boundaries imposed by the definition of the privacy model, an assessment on the performance of a PET can be conducted. A PET consists, therefore, of a col-

lection of tools, applications, or systems that provide and enable privacy-preserving measures on the data set within a given architecture such as a manufacturing environment. They enforce or enhance the privacy of the target domain in regards to their transactions and the overall digital communication that takes place. The understanding of what constitutes a PET can vary hugely depending on the research area and goals of the research conducted [135]. In order to provide an overview on the development of PETs, we give a brief historical overview on the development of this area of research before we conclude with the most recent PETs.

PETs have been developed over several decades in the past and are usually evolving alongside the internet and the broad application of digital technology in every day life [135]. This trend is usually followed by increasing concerns over privacy and data protection, which favored the development of PETs [136]. Early PETs are, thus, closely related to the initial foundations of the development of the internet and modern communication technologies in the 1980s. The origins of PETs can be traced back to the year 1981, where according to literature the first application of a PET was proposed [135]. The authors of [138] apply in their work public key infrastructures and digital pseudonyms are used as PETs in order to propose an email system that provides privacy guarantees for anonymized email communication. From there, as further cryptographic technologies emerged, PETs were often considered in context with these cryptographic primitives and were aiming to secure data and communication with them. Examples for this includes public key infrastructure (PKI), which is based on the public-private key cryptographic systems. Also, the period of the 1980s saw more legislation coming into effect that governed individuals' privacy and also influenced PETs such as the OECD's Guidelines on the Protection of Privacy and Transborder Flows of Personal Data in 1980 [139].

With the beginning introduction of the internet and its rise to popularity in the 1990s, more PETs were being developed. This includes, for example, such technologies as Secure Sockets Layer (SSL) and its successor, Transport Layer Security (TLS) [140]. The term of privacy-enhancing technologies was also coined within this time period. It was first introduced in 1995 within a report compiled by the Information and Privacy Commissioner of Ontario and the Dutch Data Protection Authority [139, 136, 135]. Furthermore, in 1996 the Health Insurance Portability and Accountability Act (HIPAA) became legislation within the USA. HIPAA's aim is protecting the privacy of patient data, specifically those stored in electronic health records [121]. The HIPAA also sparked a tremendous increase within the research of

privacy models. The first of the models developed during this time was k-anonymity (see Section 2.3.1 for more details on the historical development of privacy models).

The area from the year 2000 onward saw a continuation of the trends outlined by the 1980s and 1990s. From the perspective of communication technology, privacy issues became more of a publicly discussed concern. This was due to the social media platforms and big data applications that saw new potential for data extraction and generation of value associated to that data [141]. Also, further legislation aimed at protecting individuals privacy were introduced by governmental bodies worldwide. For example, in 2009, the Health Information Technology for Economic and Clinical Health Act (HITECH) was passed into legislation within the USA. HITECH can be regarded as an improvement upon the HIPAA and introduces also new regulations for privacy protection in the medical and healthcare domains. Another example with the intend of broader application is the General Data Protection Regulation (GDPR) first published by the EU in 2016 [142]. With the ongoing application of machine learning techniques to data sets, a new paradigm of PETs emerged [143]. Differential privacy, for example, has become a key technique in preserving privacy in machine learning datasets [144, 145, 146].

We already discussed differential privacy as a privacy model in Section 2.3.1.1. Differential privacy is, first of all, a definition rather than an algorithm [130]. We adopt a view on PETs within that work that sees PETs as a means to implement a privacy model. That is, a privacy model for us constitutes to theoretical definitions and mathematical formalism rather than specification on the the concrete algorithms and technologies used to implement it [126, 130]. The realization of a privacy model for us is conducted within the context of a PET, which includes the required algorithms, tools, and technologies necessary to provide the privacy guarantees the privacy model is capable of. In Section 2.3.1.1 we discussed differential privacy in the context of privacy models, which is the view supported by most authors [126, 130, 121, 147]. Therefore, we consider other technologies as a PET rather then differential privacy. These technologies include approaches such as homomorphic encryption, secure multi-party computation, or federated learning [143]. In the following section, we discuss federated learning and provide a background on this PET.

2.3.2.1 Federated Learning

Federated Learning is a machine learning approach that is also considered to be a PET [148, 149, 150]. Federated learning is a decentralized approach to machine learning and allows a model to be trained by using the data stored on multiple decentralized devices. These decentralized client devices each are in possession of their local data sample that is required to train the algorithm on a central server. This way, the algorithm can receive data from the client devices without the need for exchanging the data itself. That is, the data remains on the client devices, which execute a partial training process on a model provided by the central server. Then, the trained model or the parameters of the model are sent back to the central sever. By this approach, the data can remain on the client devices and the central server can use the partial models to derive an aggregated model from it. The central server can thus arrive at a fully trained model without the need to possess the actual training data. Federated learning is useful in scenarios where privacy is critical or where data cannot be centrally stored due to regulatory constraints or technical limitations [151, 152].

Federated learning is a comparable recent concept when seen in the context of PETs [143] (see previous section). It was first published by researchers at Google in 2016 and can be regarded as a response to growing privacy concerns of end users for data privacy [148, 141, 150]. Federated learning was initially conceived for the application on mobile end devices, i.e., smartphones. The originally proposed architecture allowed for data acquisition from a large number of client devices without the need to transfer raw data from those devices back to a central server. Initial use cases for federated learning employed algorithms tailored to improve mobile services such predictive text for keyboard inputs. From there, more research is conducted on additional use cases but also on how to improve the efficiency of federated learning and how to secure federated learning from malicious participants [153]. Though federated learning was conceptualized for mobile end devices, it can be applied to all setups where distributed data collection occurs such as in industrial settings [151, 152, 149].

We conclude this section by providing a basic overview of the architecture and algorithms of federated learning. This architecture is generally applicable to several different use cases and can be extended upon as required. This basic architecture with its components is given by Figure 2.6. On the bottom of Figure 2.6, the client devices are depicted. These client devices are in possession of the data and correspond to the data subjects in Figure 2.7 discussed in more detail in the following

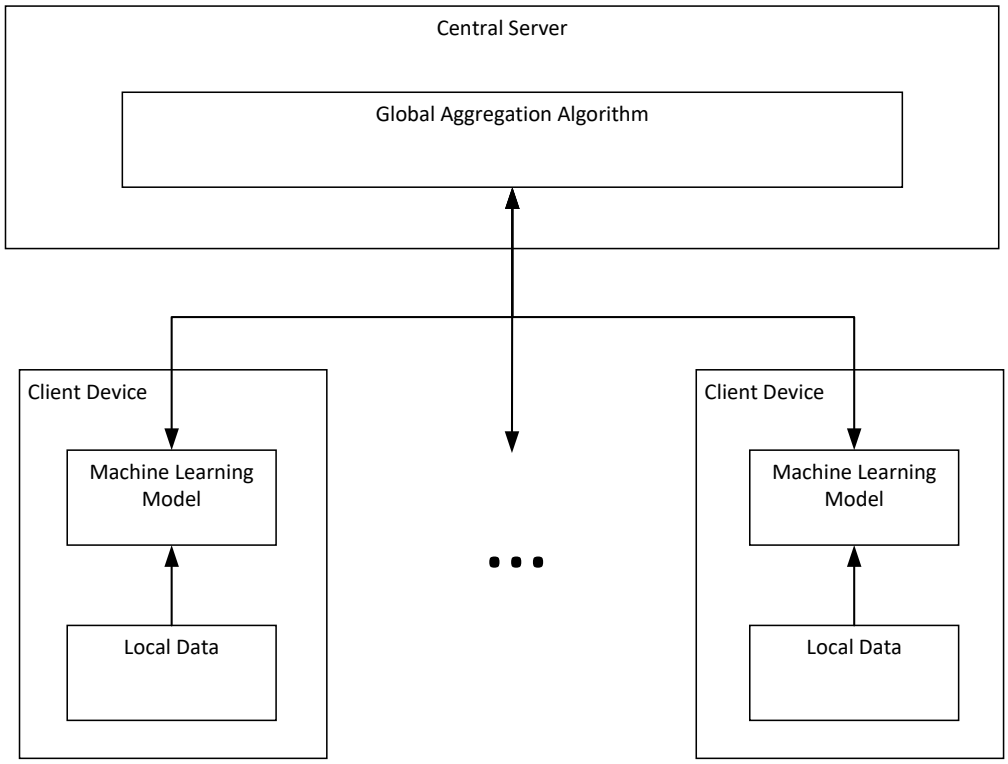


Figure 2.6: Conceptional system architecture for federated learning applications including central server, client devices, machine learning model, global aggregation algorithm, and local data sets.

section [130]. The client device can represent a variety of different devices such as mobile IoT devices or stationary tooling machines of a production facility. Each of the client devices holds a part of the data required to train the machine learning model. This model is distributed by the central server to each of the client devices. The client devices then update the machine learning model via a local training process. That requires the client device to also possess enough computational power and storage capability in order to perform the local training process. Modern CPS used in manufacturing as introduced by Section 1.1 (cf. also Section 2.1.1 and Section 4.2.1 for more information on CPS and their capabilities) are considered to be able to perform such computations within tolerable boundaries [152, 150]. After training of the model is completed, the updated model and not the data used for the training process is sent back to the central server. The central server then aggregates the updates it receives via a global aggregation algorithm [148]. So, the

central server creates an improved global model. This global model is then employed for the use case the architecture was realized for or can also be distributed back to the client devices for further local training and improvement of the global model. This process continues iteratively, the global model can continuously be improved upon as new training data is available for the client devices or as new client devices are introduced to the architecture.

2.3.3 Privacy Pipeline

In this section we discuss the general approach to applying privacy-preserving methods and algorithms, that is, a privacy model in combination with PETs, to a data set. This is a general approach and is extended upon in most real-life applications, e.g., within the domain of healthcare in order to address domain-specific requirements [154]. Doing so is usually a collaborative effort among various stakeholders within an organization. Depending on the use cases for the privacy-preserving data set, this task can become increasingly complex as the application of privacy-preserving methods in the form of privacy models such as Differential Privacy to data sets is a process that involves multiple steps and participation.

This process is shown by Figure 2.7, where an abstract view on the participating entities and the different steps involved in generating a privacy-preserving data set are depicted. The terminology in Figure 2.7 is adopted in parts from [130], whereas the flow of events is adopted from best practices of applying anonymization to medical data [155, 154, 121]. From top to bottom, we start with the data subjects located on the very top of Figure 2.7. The data subjects are the individuals from whom the individual data records that constitute the original data set is collected from. They are, thus, the owners of the data that is being collected and processed for later analyses. Data subjects, as the data owners, have certain rights to their data depending on the legislative context they are subject to. Data subjects can include a wide variety of potential data sources like customers, patients, website users, tooling machines, or any other entity whose produced data is of interest to an organization. It is important for organizations to handle the personal data of data subjects in a compliant manner for upholding the data subjects' privacy. The collected and accumulated data from the data subjects is the raw data set, which contains sensitive information about the data subjects.

The raw data set is then processed by the data curator. The data curator is responsible for managing and handling the raw data set. The data curator ensures

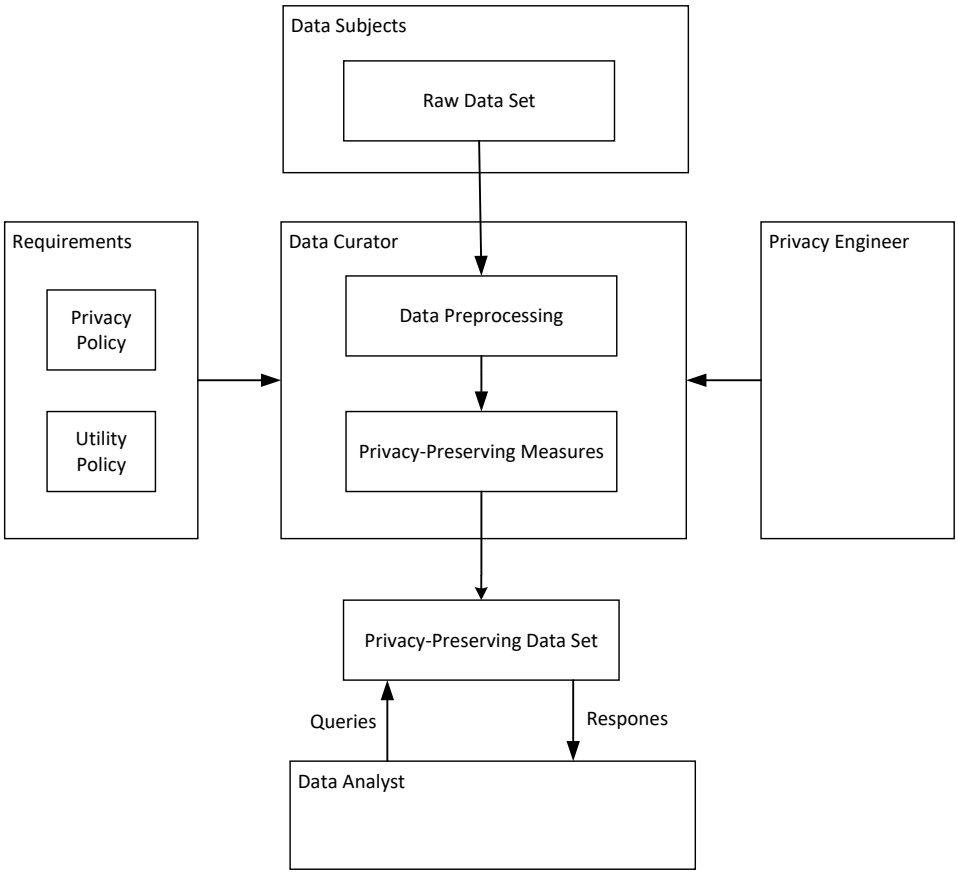


Figure 2.7: Pipeline for privacy-preserving data publishing.

that the data is processed accordingly by privacy-preserving measures. This is a two step process that begins with data preprocessing as most raw data sets require some preliminary actions for proper processing by privacy-preserving algorithms [46]. These preliminary actions can include simple formatting of the raw data set such as swapping of columns, changing of data formats, or pivoting. Additionally, first privacy-preserving measures can be applied during data preprocessing such as de-identification, that is, the deletion of directly identifying information as discussed above. After preprocessing of the data is finished, the actual privacy-preserving measures are applied to the preprocessed data set. That is, the specified privacy protection mechanisms, e.g., Differential Privacy, are executed on the preprocessed data set before any form of data analysis takes place. The selection of suitable privacy-preserving measures, i.e., the privacy model, is performed by including a

variety of stakeholders. These stakeholders, such as legislative bodies or the data analysts (see below), have a set of different requirements as seen in Figure 2.7 on the left-hand side. These requirements typically tend to be in tension with each other as they reflect the PPDP dilemma of providing data with sufficient utility while ensuring privacy guarantees are being met [126, 47]. Aptly named, these requirements are summarized by the privacy policy and the utility policy by Figure 2.7. These policies are known to the privacy engineer (right-hand side of Figure 2.7), who is an expert specializing in the design and implementation of privacy-preserving systems and mechanisms. This can include a wide range of algorithms and technologies such k-anonymity, differential privacy, and others [121]. Privacy engineers work closely with data curators in order to develop privacy-preserving solutions that are suitable for the given use cases and fulfill all relevant requirements. They further ensure the proper implementation and application of the privacy-preserving measures so they function as expected. Later, during operation of the privacy pipeline, the data curator can decide on the privacy parameters, such as the value for k or the amount of noise to be added to the preprocessed data set.

The successful conclusion of these steps results in the privacy-preserving data set that can be published or access to it can be granted to a selected audience of data analysts. The data analyst is the entity that performs the analysis on the anonymized data and can be an individual or a group of individuals but also an automated computer system. They perform data analysis tasks that are summarized by queries in our pipeline for privacy-preserving data publishing. Typically, data analysts use their queries to utilize statistical methods and algorithms to derive insights or conduct computations on the privacy-preserving data set. As the data analysts require a certain amount of utility, they are stakeholders that influence the requirements for data curation. They are the consumers of the privacy-preserving data sets, for example, medical professionals that conducted research on certain diseases and data sets provided by other medical institutions. Data analysts can include any number of potential users who want to utilize the aggregated information without directly accessing the sensitive data in the original raw data set. Therefore, data analysts are sometimes also referred to as data users.

3 Related Work

In this chapter, we discuss the work related to this thesis. In contrast to the discussion of Chapter 2, where we give a broad background on the topics touched by this thesis, our discussion in this chapter is directed towards individual work. We do, however, also include some discussion of a broader nature within this chapter where it appears necessary. Especially in the field of information security testbeds and privacy-preserving technologies, the sheer number of publications available within the research community makes it necessary to narrow down our focus. This is achieved by discussion of broader topics and trends that are sorted and assessed by us prior to the discussion of relevant, individual works.

We start with our discussion of related work in Section 3.1 with the topic of information security in manufacturing with digital twins. The general state of information security in manufacturing is the topic of Section 2.1.3, why we go into the specifics of information security with digital twins at this point. Digital twins, in theory, can be applied to broad range of areas with the domain of (connected) manufacturing constituting to one of the frequently discussed areas [42, 18]. Also, we consider the topic of modeling of digital twins separately and in detail. Providing a satisfying model for the digital twin is highly relevant as is also seen in Section 6.3.2.

The next part of our discussion on related work is on privacy-preservation in manufacturing given by Section 3.2. As can be seen from our discussion on the privacy in the domain of manufacturing in Section 2.3, this is a potentially vast field of research that provides an exhaustive body of literature. For this, we touch down on the on the individual topics of privacy-preservation in manufacturing with differential privacy and with federated learning. There exists some work on the application of both paradigms in manufacturing [144, 150] with some individual research papers relevant to our research. These works are discussed and examined by us in detail in order to provide proper context for our thesis.

These discussions of individual works in Section 3.1 and Section 3.2 provide the most relevant work and how these papers relate to our research. The summary of

our discussion is given by Section 3.3. Here, we discuss the novelty of our research in the context of the body of literature examined by us. This reflects on the state of the art as of compilation of this thesis in its current form in 2023.

3.1 Information Security in Manufacturing with Digital Twins

Digital twins are embodied within their respective context. This context is comprised of their target domain, e.g., aerospace or manufacturing, and the processes and data that is used for their realization [42]. An abstract summary of that context can be seen within Figure 2.2. Here, the digital twin is embodied within a virtual environment and connected to the physical environment where also the physical counterpart of the digital twin is located. The data transferred between the two environments is one of the key characteristics distinguishing a digital twin from a simulation [28] (see also Section 2.1.2). The quality of the data transferred is characterized by its accuracy. For digital twin and testbed development in general, the term *fidelity* is used to describe the resolution of the data fed into the digital twin [27, 33] (see also Section 4.3 for an in-depth exploration of fidelity and related concepts). The transmission of high-resolution data, i.e., large amounts of data, in real-time from the target domain, e.g., a manufacturing environment, to the digital twin is challenging in regards to the necessary technology [28, 42, 18]. This is due to the required network speed and processing power that is currently not available within the domain of manufacturing (see also Chapter 1). However, digital twins in manufacturing (and other domains as well) are a topic of study that received interest from academia and corporations in the recent years with many publications claiming to have constructed an implementation of a digital twin [59, 156, 33]. Thus, we include the discussion of information security testbeds developed for an industrial context in our discussion of related work. From there, we continue with a discussion of such publications that are more closely related to the concept of digital twins as previously discussed in Section 2.1.2.

A testbed in academia is in general a controlled experimental setup used to test and validate new theories, hypotheses, or scientific models [34, 27, 33]. Testbeds are often designed to mimic real-world conditions, allowing researchers to observe and measure the behavior of a system under various conditions. Testbeds in science can range from physical setups, such as laboratory experiments or field studies, to virtual simulations or computational models. They are used to examine complex systems or phenomena, e.g., weather patterns, ecosystem dynamics, or the behavior of subatomic particles. When related to engineering and technology, a testbed can be understood as a physical or virtual setup for conducting experiments, sim-

ulations, and tests to observe the behavior of a system or product under various conditions. With this procedure, a testbed can help with identifying potential problems or issues with a system and with improving the system's performance. From the definitions given within this paragraph, it can be seen that testbeds share some similar properties with digital twins [28, 59, 42]. These similar properties include their construction with virtual components and the intend to mimic some sort of real-world system. Thus, research related to digital twins is currently concerned with developing testbeds for executing the digital twin within the testbed's context. Therefore, we discuss the state-of-the-art on testbed research related to information security within the domain of manufacturing as a whole before we examine those publications and projects that offer the most advanced work on digital twins up to date.

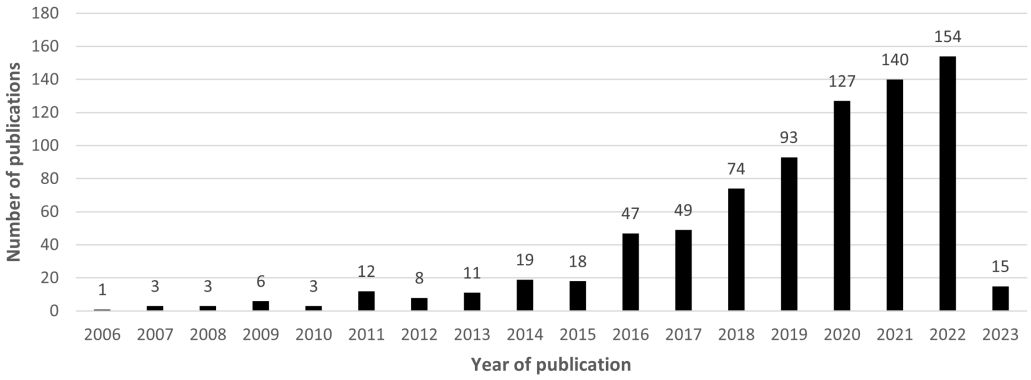


Figure 3.1: Number of publications for information security testbeds and digital twins in manufacturing with record date 16th March 2023.

A quantitative overview of the publications within the area of information security testbeds for manufacturing is provided by Figure 3.1. The figure shows the total number of published research papers grouped by the year of their publication. The data for this analysis was gathered by us from the database Scopus[1]. We searched the abstracts, titles, and keywords of all publications listed within Scopus for the search parameters 'security' and either 'testbed' or 'digital twin'. In addition to these two search parameters, we added a third parameter expressing our target domain of connected manufacturing. For the identification of this domain, at least one of the following terms must be included within the publications abstract, title, or keywords: 'ics' (for industrial control system), 'ot' (operational technology),

[1] `https://www.scopus.com/`, last accessed on March 16th, 2023.

cps (cyber-physical system), 'cyber-physical', 'scada' (supervisory control and data acquisition) [2]. From the results of our query, we can observe a trend of an increasing number of publications over the years. The first relevant publication was in 2006 [157]. From this year onward, a slow but steady increase in the number of publications by year can be noted until the number of publications suddenly spikes from the year 2015 (with 18 publications in total) to the year 2016 (with 47 total publications). The reason for this can be partially explained with the event of Stuxnet some years prior to this [23, 13, 24] (see also Section 2.1.3). Another explanation is the continuing trend in government-aided programs support the development of connected manufacturing worldwide [5] (see also Section 2.1). The first publications related to digital twins is found in the year 2017 [158, 159]. This shows a gap of around four years from the initial discussions of applying the concept of digital twins in manufacturing in 2013 [91] (see also Section 2.1.2) to the first experimental implementations of the digital twin concept in manufacturing in 2017. The number of publications experienced an accelerated growth from 2016 onward until its peak in 2022 with 154 publications. The trend described by us is also observed by other authors as well [32, 13, 24, 156, 33]. It is worth noting that the number of publications only including the search parameter 'testbed' (without 'digital twin') seems to start to decrease in 2022 (from 108 total publications to 96 total publications). It is, however, from the perspective of the year 2023, when this thesis is written, unclear if this hints at a general trend in decreasing number of publications or if this is an outlying data point. However, as the number of publications is steadily increasing when including the search parameter 'digital twin', it appears that testbeds tend to be overtaken by implementations of digital twins in manufacturing. This corresponds with the outlooks provided by other authors on this subject [28, 42, 40].

Our analysis on the research landscape of information security testbeds and digital twins for manufacturing yielded a total of 783 potentially relevant publications. Other, previously conducted research examined a subset of these publications. In 2015, [32] examined 30 publications whereas in 2021 [33] and [156] examined 57 and 61 testbeds respectively. Note that some of the publications discussed by [32] were also included in the discussions of [33, 156]. Surveys on the domain of information security for digital twins in manufacturing solely do to the best of your knowledge not exist currently. The most recent and comprehensive discussion of digital twins

[2] The full search parameter is: TITLE-ABS-KEY (security AND (testbed OR (digital AND twin)) AND (ics OR ot OR cps OR cyber-physical OR scada))

within the area of information security in manufacturing is given by [28]. Here, the authors focus on the discussion of potential information security use cases for digital twins in manufacturing and give a discussion of work related to digital twins for each use case. To expand upon the existing corpus of state-of-the-art research, we continue with our discussion of information security testbeds in manufacturing. In order to reduce the large amount of possibly relevant publications, i.e., 783 potentially relevant publications, we categorize those publications by different dimensions. These dimensions are the simulation technique, the use cases of the testbed, and the properties of testbed in regards to scientific experimentation [33, 34, 27]. Simulation technique refers to how the testbeds are constructed and what approach to simulation they implement. Testbeds can be constructed with real hardware and software that is executed on that hardware. That means that the testbeds are a prototype or a physical replica of another real-word component, e.g., with NASA's Apollo program or the Iron Birds [39] (see also Section 2.1.2). Software for the purpose of simulation is only used very sparsely if it is used at all. Testbeds using this simulation technique are referred to as physical testbeds. Another type of testbed is the hybrid or semi-physical testbed. They include at least one physical component. The remaining components are software components used for simulation of the environment for the physical components. Note that hybrid or semi-physical testbeds can, in principle, not function properly without the physical components. Thus, these testbeds are also referred to as hardware-in the-loop testbeds. Different simulation techniques are summarized by the term of hybrid or semi-physical testbeds, including emulation or virtualization. Finally, virtual testbeds are testbeds that make only us of software-based simulation techniques. Digital twins for us are located within the hybrid or semi-physical testbeds. The categorization of the publications among the described dimensions allows us to identify those that are relevant to the study of digital twins as they share properties of a digital twin [28] (see also Section 4.3). The relevant publications are now discussed in the remainder of this section.

In [54], the authors propose a methodology for deriving the construction of a testbed that can include a model for a digital twin. The authors are interested in constructing a cost-effective testbed. Within the context of their work, this means that different components of the testbed's library (see Section 6.1.5.2 for a discussion of what components the library for an industrial testbed can contain) are realized using different technologies. The authors provide an evaluation on the financial cost associated with each of these technologies in relation to the assumed fidelity they

could provide. Following up on these considerations, the authors present a method in order to derive a design specification for the construction of a testbed within some budget. Their approach finds that the digital twin should be implemented using technologies that can provide a higher fidelity, e.g., virtualization. Other components may be implemented with lesser expensive techniques such as network simulators. For evaluation purposes, the authors implement a small-scale testbed that does not receive real-life data as input. The work of [54] highlights the importance of considering design goals for the construction of testbeds that use digital twins. However, their work lacks in providing clear outlines on the design of digital twins and their data connection with the physical counterpart. An examination of relevant attack scenarios and attacker models are not included within their work and, thus, are not evaluated by their implementation. For this thesis, we adopt more recent and detailed design approaches for our work. Also, we consider attacker models and attack scenarios in detail in order to derive a meaningful approach to conducting information security evaluations within manufacturing. The design and the construction of digital twins including the actual level of fidelity they provide is discussed by us in more detail.

In [28], the authors provide a sound analysis on the state of digital twins in manufacturing for information security evaluations. Their work was released in 2019 and includes discussions on the concept of digital twins and their possible use cases for information security evaluations in manufacturing. They provide a working definition for digital twins that is applied for the description of their use cases. The use cases touch several aspects relevant for information security in manufacturing. The use cases are concerned with the product lifecycle of CPS and other industrial equipment. This lifecycle includes the development phase of the product, their operation in the field, and de-commissioning at the end of their product life. The theoretical discussions in their work provide valuable insights in the possible applications of digital twins in manufacturing. However, a practical evaluation or an in-depth discussion to the realization of digital twins for the described use cases is not included in their work. In our thesis, we consider the practical implications of realizing a digital twin by providing real-life test cases that may be related to some of the use cases spelled out by [28]. Also, the authors discuss digital twins not used within the domain of manufacturing but rather are conceived for usage in other domains such as automotive. Since 2019, some relevant work specific to digital twins in manufac-

turing surfaced within the research community and taken into regard by us for this thesis.

In [160], the authors describe the concept for a digital twin that is operated within a connected network within manufacturing. Their work contains a detailed discussion of the possible threats that can occur via the usage of digital twins in connected manufacturing (see also Section 4.2.2.4 for a discussion by us on that topic) and details possible countermeasures for the described threats. The authors discuss a state synchronization protocol for preventing attacks against the data connection between the digital twin and its physical counterpart. Their evaluation considers this protocol in regards of the frequency required for updating the digital twin in the light of their employment within a manufacturing context. The architecture derived for their evaluation as well as the steps taken for their conception are detailed within their work. The implementation of their digital twin is realized by a technological infrastructure that employs virtual machines for that purpose. However, their attacker model is based on the Dolev-Yao attacker [161], which can be considered insufficient for the challenges encountered within connected manufacturing (see Section 4.1.1 for a discussion on that topic). We use a detailed attacker model suitable for digital twins in manufacturing that is considered in the context of a generic attack vector model (see Section 4.2.1.1). Also, the main goal of their evaluation is their developed protocol for state synchronization of the twins rather than the digital twin itself. In our work, we consider the implications an attack can have on the manufacturing environment where the physical is operated in. Consequently, a discussion on the fidelity of digital twins is not included within their work. We discuss the fidelity of digital twins in Section 6.1.2.1 with great care.

In [162], the authors propose an open source approach to the design of a digital twinning platform in manufacturing. They provide a thorough construction of their architecture that considers many aspects relevant to digital twins and their employment within manufacturing. The core of their demonstrator, at least to our understanding of their work, is the microservice architecture they propose. In general, microservices appear not to be not well-suited for realization of digital twins. For the most part, this is related to performance and scalability issues. Managing the microservice architecture introduces additional overhead that may results in the system to pronounce a tangible delay in its operations. This way, real-time operation of the proposed digital twinning platform needs to be evaluated. Though real-time capabilities are considered by the authors within the design of their architecture, an

evaluation on this specific topic is not included in the published work. Our digital twinning platform considers and evaluates real-time considerations (see Section 6.1 and Section 7.1). Moreover, the authors do not consider information security in manufacturing as a possible use case for their digital twinning architecture. In fact, the authors point out that their approach based on micro-services may introduce new security vulnerabilities into manufacturing environments. It remains unclear how a proper protection of the system the platform is intended to be used on can be achieved (some hints on this are provided by in Section 4.2.2.4). In contrast to this, improving upon the current state of information security within the domain of connected manufacturing is our main motivation for conducting our research on digital twins. Fidelity of the digital twin that is envisioned to be used within their platform is considered by the authors. Although data acquisition is discussed and considered by their platform, it is currently unclear if a medium- or high-fidelity digital twin (see Section 6.1.2.1 where we provide a discussion on these) can be achieved within their architecture. The fidelity our digital twin can provide is discussed by us and considered in the design and implementation of our testbed.

In [163], the authors consider the integration of several digital twins to a network of connected digital twins within smart manufacturing. These twins are supposed to be connected via a variety of wired and wireless industrial communication protocols. The overall digital twin network is also connected to a private, on-premise cloud which can be regarded as reminiscent to a edge computing framework (see Chapter 7 on details how edge computing is regarded by us in this thesis). Furthermore, the whole digital twin network is connected to a cloud infrastructure reachable over the internet. The main focus of their from our point of view appears to be the design goal of interoperability for their proposed network of digital twins [33]. Interoperability is also discussed by us when laying out our design goals for the digital twinning testbed in Section 6.1.5.2. In fact, interoperability is one of several design goals that need to be considered in the construction of digital twins and testbeds related to that technology. These are not considered in the same manner as interoperability by the authors of [163]. The applicability of digital twins to promoting use cases from information security is recognized by the authors as they suggest that digital twins can provide security-by-design. This is, however, only one of the possible use cases that can be considered for increasing security within manufacturing with digital twins [28]. In our evaluation, we consider different angles on the broad topic of information security within manufacturing. Moreover, an evaluation or a reference

implementation of a network of digital twins is not included within the publication, whereas we provide a fully setup and working testbed implementation.

3.1.1 Modeling of Digital Twins in Manufacturing

In this section we discuss the current state of the art regarding modeling of digital twins for manufacturing. We begin by categorizing the different manufacturing systems. From there, we explore the usage of UML diagrams. Specifically, our area of interest is on UML state machine diagrams and their usage in manufacturing. After discussing modeling of digital twins for the domain of manufacturing, we continue with a discussion on how information security is a topic within this field of research.

Manufacturing systems are divided on a high level in two major categories: discrete and continuous manufacturing systems [164, 55]. Distinguishing characteristic between these two systems is their flow of material. Continuous manufacturing systems are involved with the production of goods that are continuously flowing. For the most port, this flow includes the mixing of different liquids, gases, or powders in chemical processes [165]. Examples for such chemical processes using continuous flow of materials are petroleum refining, polymer production, food processing, or water treatment. Products resulting from such continuous manufacturing processes are considered irreversible as they cannot be separated again in their original materials (or can only be separated again with a large amount of effort). On the other hand, discrete manufacturing is concerned with the production of separate, distinct goods. Examples for such goods include electronic, automotive, or furniture manufacturing. Goods resulting from such discrete manufacturing processes can, in contrast to continuous manufactured goods, be disassembled again into their raw components. It is worth noting that a combination of continuous and discrete manufacturing processes are encountered in some complex manufacturing process. Examples for complex manufacturing processes are semi conductor, pharmaceutical, or renewable energy manufacturing. For the work presented in this thesis, our focus is solely on discrete assembly systems.

The type of manufacturing system studied affects how the system is modeled [56, 57]. Modeling for manufacturing is studied in the context of control engineering [166]. Here, two major modeling categories are present mapping to the two types of manufacturing systems discussed above: discrete-event dynamic system (DEDS) for discrete manufacturing systems and continuous-variable dynamic sys-

tems (CVDS) for continuous manufacturing systems. As our focus is on discrete manufacturing systems, our focus is on DEDS. DEDS are characterized by discrete-state, event-driven modeling environments. Within such DEDS modeling environments, the state transition depends entirely on the occurrence of asynchronous discrete events over time. The characteristics of DEDS are part of several modeling techniques. One of these modeling techniques for DEDS can be found in the field of automata theory. It is concerned with the study of abstract machines and uses several formal notations for modeling. One of these formal notations are finite state machines (FSM). A FSM describes the properties of DEDS and is a suitable modeling technique. As described above in Section 3.1, digital twins are a virtual component meaning they are expressed as a software program. Also, this software program acts in the interconnected environment of a modern, digital factory, i.e., represented by connected manufacturing (see Section 2.1). Thus, the representation of a digital twin within a modeling technique as a connected software component is reasonable goal for our modeling approach.

A modeling approach that enables the representation of a digital twin as a software object is the modeling language encompassed within *state machine diagrams* of the UML. UML state machine diagrams or state machines for short are a representation of a technical process or device, mostly related to computer science but also are used in other domains. In the domain of manufacturing and control engineering, state machines are used for modeling the logic of devices such as programmable logical controllers (PLCs). The programming of these PLC devices with UML state machines is discussed in [167]. Here, the authors use UML state machines for graphical programming of PLC. These type of devices are used for defining control flow within a manufacturing system. This is further discussed for UML state machines in the context of mechatronic systems in [168] and for control automation in [169]. Other, more formal modeling approaches using FSM and UML state machines are found in [170]. Here, the authors emphasize the usage of knowledge-based systems rather than modeling discrete events for manufacturing. The reviewed literature on the use of UML state machines in manufacturing is limited to modeling singular devices and are extended upon to encompass larger parts of a discrete manufacturing system. Also, these manufacturing systems discussed so far do not make use of the concept of digital twins.

For digital twins, UML is used for modeling of the twin in different ways. Here, the distinction can be made between the two types of UML diagrams, structural and

behavioral diagrams, as discussed in Section 2.2. For structural diagrams, class diagrams are often used. Class diagrams are static structure diagrams. They represent a software system by its classes, their attributes, methods, and the relationships among those objects. This is used in the literature reviewed by us for modeling the digital twin as a software component. In [171], the commercial LEGO® MINDSTORMS® platform is used to build a physical and digital pair of twins. The context is limited to this platform and does not entail manufacturing systems. For manufacturing systems, the work of [172] shows how UML class diagrams are used to construct digital twins. The behavior specification of the constructed digital twins is limited to the capabilities of the static UML class diagrams, dynamic UML modeling techniques are not considered. Dynamic UML modeling techniques are used in [173]. Here, the authors employ activity and object diagrams. These UML diagrams are limited as they do not allow for modeling behavior over time or concurrency as well as reducing complexity via nested states (i.e., hierarchical modeling). These are important properties for modeling digital twins in the context of connected manufacturing as discussed in Section 5. Another important property is the ability for UML state machine diagrams is their ability to act as an event-driven modeling technique. Event-driven modeling is, furthermore, not part of UML sequence diagrams as they are used in the work of [174]. Their work is mostly focused on building and using repositories for digital twins rather modeling executed twins within a connected manufacturing environment.

UML in general and UML state machines in particular are used for modeling information security. Security applications are considered within UMLsec and its extensions [115, 116]. Here, UML state machines, among other diagram types, are enhanced to support the specification and modeling of security requirements in software systems. UMLsec provides a set of modeling constructs and notations for capturing security concerns and defining security policies, such as access control and confidentiality, at different levels of abstraction. However, UMLsec has some limitations that may hinder its use in manufacturing. One of the main limitations of UMLsec is that it is primarily designed for the specification of security requirements in software systems, and may not be suitable for modeling security concerns that occur or interact with the physical world. In manufacturing, security risks may involve physical access to machines or equipment, and UMLsec may not provide the necessary modeling constructs to capture such risks. In particular, the effects a cyber attack on the physical world can have is not captured by UMLsec.

Another limitation of UMLsec is that it may not provide a complete solution for modeling security concerns in complex manufacturing systems that involve multiple interconnected components and subsystems. UMLsec is typically focused on the modeling of individual software components, and may not provide the necessary mechanisms for modeling security concerns that span multiple components or subsystems as it is the case in connected manufacturing. The same limitations as for UMLsec apply also to SecureUML [117], another security-specific extension to the UML standard [118]. UML state machine diagrams are used for security by other authors. They employ the properties of UML state machine diagrams for security verification of software [175, 9]. This verification of software in regards to security is further extended on by [118]. All of the discussed literature have in common, that they are intended for security verification of software in general. From this general point of view, the specific needs of other domains are not taken into account. In particular, the manufacturing domain is not considered by the authors. Also, the UML state diagrams of the reviewed literature do not encompass security from the view of interconnected systems and the specific physical impact security incidents can have. This, however, needs to be considered when studying security in the domain of manufacturing. Recent surveys focusing on modeling of digital twins in manufacturing also do not consider security as a potential use case [57, 56]. On the contrary, it is mentioned in [56] that in order to improve the accuracy of behavioral modeling for digital twins, that anomalous data is often filtered. This filtering of anomalous data may, however, of key importance to study security incidents as attacks can be classified as anomalous behavior of the system [109, 176].

To summarize, UML state machine diagrams are well suited for modeling digital twins in discrete manufacturing systems. As discrete manufacturing systems are represent by the modeling category of DEDS, UML state machine diagrams represent their characteristics as they provide a way to model FSM. Furthermore, modeling digital twins via UML in manufacturing for the study of security use cases is not currently well understood.

3.2 Privacy-Preservation in Manufacturing

In this section we discuss related work on privacy-preservation in manufacturing. We start by discussing privacy-preservation in manufacturing via the privacy model of differential privacy in Section 3.2.1. Our approach is to consider relevant surveys if such recent surveys on the topic are available. We require also a brief discussion of terminology in order to categorize the available research and provide a streamlined discussion within the terminology used by us in this thesis. From there, we explore the application of federated learning in Section 3.2.2. We use the same approach as with our discussion of differential privacy in Section 3.2.1. In addition to this, we delve in the related work for privacy-preservation in manufacturing with federated learning for digital twins in Section 3.2.2.1.

3.2.1 Privacy-Preservation in Manufacturing with Differential Privacy

In this section, we discuss the current state of the art on the application of differential privacy to the domain of (connected) manufacturing. As indicated in Section 2.3.1.1, differential privacy has received tremendous attention by researchers, who apply the definition of differential privacy to a variety of different domains and use cases [130, 128]. For the formal definition see Section 7.3.1 or [126], for gaining an intuition on what is implied by the concept of differential privacy see Section 2.3.1.1 or [132, 131]. As the definition of differential privacy is general, differentially private algorithms can be applied to a variety of different scenarios and data types. The area of privacy-preserving data publishing in general provides an exhaustive body of literature, be it on the development of new privacy models [121, 123], their application to various domains [177, 144, 145, 146], their performance analysis [130, 153], or possible future applications [149]. For that, we narrow down the body of literature to be reviewed by us within this section. For this, we discuss only those publication from now on that are related to differential privacy and leave those with other privacy models such as k-anonymity out of scope [121, 134].

For keeping our discussion on privacy-preservation within manufacturing focused, we particularly examine those publications that are related to manufacturing rather than research in mathematical and algorithmic theories on privacy-preservation. In order to get a better grasp on the term of manufacturing, we discuss first how this topic is discussed by other academics and what terminology they use within their discussions. Within the three recent surveys that explore the application of differential

privacy to, in a broad sense, embedded devices[144, 145, 146], different terminology and structuring for the manufacturing domain are used. In [144], the authors discuss differential privacy for cyber-physical systems (CPS, see also Section 1.1 or Section 4.2.1) and name manufacturing as an use case for CPS. Specifically, they employ the term of Industrial Internet-of-Things (IIoT, see also Section 2.1.1 for a discussion of this term in a broader context) when referring to applications within connected manufacturing. In contrast to this, the authors of [145] explore applications of different privacy for IoT devices and regard the domain of IIoT as a special case of IoT that relates to the domain of manufacturing. Taking a different approach, the authors of the survey published as [146] in particular discuss the application of differential privacy for IIoT devices. However, they distinguish further by their area of applications, which are given in [146] as the following: industrial logistics systems, smart grid, industrial bio-engineering, industrial unmanned aerial vehicles (UAV), intelligent manufacturing, blockchain in industrial informatics, and industrial social networks. The other surveys also categorize their work according to different areas of application. So does [145] list smart grids, intelligent transport systems, healthcare, and the aforementioned IIoT as their areas of application for differential privacy within IoT. The authors of [144] also name similar areas of application: energy systems, transportation systems, healthcare and medical systems, and IIoT. For the two surveys of [144, 145], we regard after careful consideration of their the area of application IIoT to be corresponding to our understanding of connected manufacturing as outlined in Section 2.1.1. The same can be said for the area of application named as intelligent manufacturing by [146]. To summarize, literature on applying differential privacy to some sort of embedded devices mainly focuses on the domains of energy/smart grid, transportation systems, healthcare, and connected manufacturing (referred to as by other terms, that is IIoT and intelligent manufacturing). This is true for [144, 145], whereas [146] adds additional domains like UAV or social networks that are also out of scope for us.

Having now gained a better understanding of what is meant when the domain manufacturing is discussed within the context of privacy and anonymization, we now discuss the data sets encountered within the domain of manufacturing. Focusing on the data the privacy-preserving techniques are applied early on is an important step within developing sufficient privacy controls [121, 178, 179]. In literature, several data categories are discussed in the context of privacy-preservation, such as relational and transactional data in databases [180], sequential data like DNA, tra-

jectory data describing movements, data in the form of descriptive text for medical records, images, or data organized as time series. For the domain of (connected) manufacturing, especially trajectory and time series data are of relevance as they are the most common data categories encountered within that domain [181, 182, 144]. Industrial trajectory data is concerned with movable entities that are present on the shopfloor. Existing research applies its privacy models to trajectory data collected from human individuals, that is, employees within a plant, and is, thus, concerned with the privacy of individuals [182]. At this point, it is worth distinguishing industrial data further in regards to what exactly constitutes the data subject, i.e., the producer of the data [130] (see also Section 2.3.3). Data in industrial setting can be produced by human being such as operators or personal like it is the case with trajectory data typically encountered within manufacturing. Our focus in this research is on the data produced by tooling machines and the added value that can be extracted from them by different types of security and optimization methods [46, 47]. Therefore, we define trajectory data as out of scope within the context of this thesis. Data in the form of time series, however, is the primary data type produced by tooling machines and, therefore, the dominant data type occurring in manufacturing [46, 183, 47] (see also Section 7.2). Time series data in manufacturing refers to a type of data that describes measurements that occur and are recorded at regular intervals over a specific period of time [182]. Such data collected in a chronological order, with each data point associated with a specific timestamp indicating when the measurement was taken. Representing time series data as a graph (cf., for instance, Figure 7.2a) typically shows the recorded measurement along the y-axis and corresponding time where the measurement was taken along the x-axis. The frequency of data collection, i.e., the measurement intervals, can vary depending on the requirements of the manufacturing process and the available technical infrastructure. For most tooling machines and applications nowadays, however, that frequency is located somewhere in the range of milliseconds. Also, it is worth noting that there is also data in modern OT networks that is reminiscent to those encountered in IT networks like Internet Protocol addresses. While there exists some research on the application of anonymization towards those data within an industrial setting [134], this is also out of scope for our research interests.

Having discussed the wider view on the field of privacy-preservation with differential privacy in manufacturing, we now discuss individual work that is relevant to this thesis. To recapture, we discuss such articles that discuss differential privacy

within the domain of connected manufacturing, is concerned with the use of data produced by machines or devices rather than by human individuals, and is related to time series data.

In [184], the authors propose an approach to entropy minimization for differential privacy within manufacturing. More specifically, the authors discuss distributed controls systems within an industrial setting. Thus, their research focus is on the performance of their proposed approach, the application is limited to a set of shopfloor devices in a wireless sensor network (WSN). As WSN can be a valuable source for time-series data in manufacturing, it is limited in their scope as WSN only make out a portion of the devices used in (connected) manufacturing. The authors achieve mathematical proofs of lower bound properties for differential privacy in this setting. The distribution of the data sets outside the hypothesized manufacturing environment is not discussed. Though they do consider a central server in addition to a peer-to-peer architecture, the evaluation does not capture the architecture in an extended way.

In [185], the authors propose a edge-based computing framework that employs differential privacy. The data processed by the framework is as in the work of [184] procured from a WSN within a an industrial setting. The work is concerned with the acquisition of the raw data, their processing, and storage of the data in the proposed edge-computing framework. For this, the authors of [185] discuss a storage architecture that is composed of three layers in which the data is divided upon. Also, encryption schemes are discussed by the authors. The work is limited to processing raw data within an isolated setting of a manufacturing environment. Collaborative data sharing mechanisms are not within the focus of the authors. Cloud applications are considered but specifically in terms of bandwidth and transmission optimization. An application of their to an industrial use case is not discussed.

In [186], the authors propose a privacy-preserving framework for application within a smart manufacturing context. That corresponds, at least to our understanding of the work discussed by [186], to connected manufacturing (see Section 2.1.1). The authors apply differential privacy to data from a tooling machine that is controlled by Computerized Numerical Control (CNC), a procedure for the control of industrial devices such as tooling machines. The authors use data procured from a real-world industrial setting that implements a turning process for their analysis. As the data is collected from a real-world manufacturing environment outside a controlled laboratory setting, methods for automated data collection and distribution are not

discussed. That includes using edge devices or cloud infrastructures. Their use case is related on the optimization of machines as they study the optimization of power consumption of the machines used in the real-world setup. However, collaborative measures are not considered by the authors as is general information security.

3.2.2 Privacy-Preservation in Manufacturing with Federated Learning

In this section we discuss related work in regards to the application of PETs in manufacturing, specifically to federated learning. As discussed in Section 2.3.2.1, federated learning is a comparatively new field of research that has experienced tremendous interest by researchers [149, 150, 143, 147]. Originating from its original use cases for privacy-preserving machine learning used on smartphones, federated learning is applied to a variety of different new use cases and domains including the domain of manufacturing [148, 153, 151, 152]. For narrowing down the large body of research, we are interested in such work that is related to the work proposed by us in this thesis. That is, we examine that work more closely that is related to connected manufacturing. Taking our discussion on the terminology for connected manufacturing from Section 3.2.1 into account, we identified two very recent survey papers that concern themselves with IIoT and the application of federated learning [153, 150]. While the focus of [150] is solely on surveying such research papers that are concerned with IIoT, the survey by [153] also discusses other area of applications. From there, we identified the related work relevant for this thesis.

The authors of [187] propose an approach to the privacy-preserving application of machine learning in IoT systems. They use two PETs within their proposed scheme, that is, federated learning and blockchain technology. Federated learning appears suitable for connected manufacturing as it allows to work on heterogeneous data sets and can be executed on CPS devices as they are found in modern manufacturing environments [148, 149, 150, 153] (cf. also Section 2.3.2.1). Using blockchain technology in conjunction with federated learning can provide some benefits to industrial applications but also offers certain drawbacks [152]. For one, this is due to the strain blockchain imposes on the resources on devices located on the shopfloor. The process of verifying transactions and adding them to the blockchain is computationally intensive, which potentially slows down the overall learning process. This can be a significant problem in an IIoT environment with its imposed real-time constraints [21, 22, 17] (see also Section 7.1.1 for a continuation of this discussion). This is further complicated by the communication overhead of the blockchain architecture

as the constant communication between nodes for model updates and blockchain transactions can introduce significant network overhead. This network overhead may be manageable in such networks with faster network technologies. However, as laid out in Section 1.1, many legacy systems and communication protocols may still be in use and plant operators generally do not tolerate any disruptions in the availability of the manufacturing process [20]. Also, the application of blockchain technologies in manufacturing can lead to scalability issues that are also related to the real-time constraints in that domain. The capacity of a blockchain for handling transactions is significantly lower than that of a centralized systems or server. As the size and complexity of the IIoT network increases over time, this limitation can lead to bottlenecks and a negative impact on overall system performance and availability.

In [152], the authors discuss their proposed communication-efficient federated learning framework for application within IIoT. Their proposed architecture features a cloud infrastructure with several connected edge devices. Their work was published simultaneously to our previous work with [47]. That framework is designed with being resilient in regards to two specific types of threats: gradient leakage attacks and label-flipping attacks. Gradient leakage attacks assume an untrusted cloud provider that can access sensitive information whereas label-flipping attacks consider the possibility of a malicious node that tries to manipulate the learning process. The potential leakage of IP due to a malicious competitor as discussed by us in Section 4.2.2.3 is not considered within the threat model of [152]. Furthermore, their threat model and architecture do not incorporate security controls for securing the communication to the cloud and for providing control to the data subjects over their data. In addition to this, the capabilities and the realization of the cloud computing device within their scheme is not discussed in great detail. Moreover, their system currently lacks in being applied to use cases from an industrial context. Their evaluation is concerned primarily with the system's defensive capabilities for the attack vectors outlined above. More practical, industry-based use cases are not discussed and their evaluation lacks to demonstrate its effectiveness within real-world applications.

3.2.2.1 Privacy-Preservation in Manufacturing with Federated Learning for Digital Twins

The application of federated learning to digital twins without the context of manufacturing is observed by [57]. The authors draw the conclusion that federated

learning has "*not yet been broadly applied for digital twins*". For the domain of connected manufacturing or IIoT, the very recent survey of [188] examines the usage of federated learning and digital twins. Overall, their survey could not identify any publications within the IIoT domain that employ both concepts in a single research paper. However, we conducted our own literature review and were able to identify one related work, that is, the article published by [189]. As this article was written before the survey conducted by [188]. According to the search methodology layed out by [188], the article should have been discovered as [189] contains the phrases *federated learning* and *digital twin* in its title and is clearly related to IIoT, which is also a part of the title for [189]. Currently, we cannot provide further reasoning for this occlusion of [189] from [188].

In [189], the authors propose, as mentioned above, an architecture for federated learning with digital twinning technology. Their work furthermore considers edge computing as their underlying communication infrastructure is a network of edge nodes, similar to [152]. Furthermore, the proposed architecture of [189] is used within an industrial context and is evaluated on an industrial use case, that is, predictive maintenance. We point out that our own work related to privacy-preserving edge computing in manufacturing [46] was published prior but did not include a larger network with several edge devices and also federated learning was not yet applied. However, the evaluation of similar industrial use cases (see Section 4.2.2.3 and Section 7.5.1 for details) with our privacy-preserving edge framework was outlined in [46] and evaluated in another of our previous work in [47]. However, digital twins were not considered by us in both these previous works but in another of our previous work that, however, was not concerned with the application of PETs to it. Our work in [47] and the work of [152] appear to be published at the same narrow point in time during the year 2021. In [152], the authors further include no threat models, either for information security or privacy-preservation, and do not provide a control mechanism towards the data subjects. Also, differential privacy is not part of their proposed scheme.

To the best of our knowledge, the work of [189] is the only published research that considers federated learning and digital twins for connected manufacturing.

3.3 Summary

In this section we discussed the current state of the art in respect to the application of digital twins and privacy-preserving technologies within the domain of manufacturing. For digital twins, our focus is on the potential usage of the digital twin technology for information security evaluations. We discussed the body of literature related to digital twins from the perspective of testbeds. Testbeds and similar simulation environments are used for a longer period of time for conducting information security evaluations within (connected) manufacturing. Digital twins are a rather new concept that is experiencing increased attention by researchers active in this area. For privacy-preserving technologies in that area, we examined the application of privacy models and PETs. From the available, comprehensive body of literature, we draw conclusions that show how our own work is categorized within it. Our findings are discussed above in detail and are summarized in this section.

For most of the discussed work on digital twinning frameworks, the evaluation described do not cover the full extend possible with a digital twin. Rather, further conceptualization of the digital twin in manufacturing is proposed by some of the authors examined within this section [163]. Furthermore, a striking number of recent work does not discuss the for digital twins important topic of fidelity within their proposed digital twinning frameworks [162]. In this thesis, we consider fidelity and other relevant design goals [33]. We show how this can culminate within the definition of different levels of fidelity and their realization within the context of an information security testbed for manufacturing [42].

As recent work shows, the study of information security use cases within the context of digital twins is a promising endeavor [28]. Some work reviewed by us within this section does provide a framework that potentially can enable further information security evaluations with digital twins in manufacturing [54, 162]. However, most use cases for information security appear to be not realized at the moment within the context of digital twins. We also do not claim complete coverage of these use cases within our work, however, the amount of possible applications for the digital twin in manufacturing evaluated by us is not found in other related literature at the moment [55, 47].

For most of the discussed work on privacy-preserving technologies in manufacturing, with very few exceptions, e.g., the evaluation described by [186], an evaluation of the proposed schemes on actual machine data with an industrial use case is hard

to come by. Most work is concerned with providing reference implementations with the goal of substantiating formal methods. While formal methods are important to consider in the design of suitable privacy controls, the practical aspects of deploying such controls within manufacturing environments need also to be considered. This is achieved by us in this work as we take requirements from the manufacturing domain into account.

In general, it is our impression that the works related with the protection of information security tend to disregard privacy concerns and vice versa. That is, the published research on the application of privacy-preserving measures within the context of IIoT and connected manufacturing lack the integration of information security controls within their proposed schemes. Also, some privacy controls such as providing control to the data subjects about the usage of their data appears to be not considered by researchers [47].

To the best of our knowledge, which is based on our comprehensive literature review presented in this chapter, no research paper as of compilation of this research exists that combines the following methods and technologies into a unified framework for the use in connected manufacturing: digital twinning, edge computing, differential privacy, and federated learning.

4 Platform Architecture and Attacker Model

In this chapter, we provide our platform architecture and the attacker model we use. Overall we give a brief overview on the target domain our thesis is investigating and the state of information security within it. We continue from the background on connected manufacturing given in Section 2.1. Our discussion of the target domain takes this background into account and extends on it. That considers current best practices and further takes the evolving nature of the manufacturing landscape into account [1, 2, 3]. Specifically, we focus on including digital twins [56, 57]. We start with the attacker model and its underlying assumptions in Section 4.1. Attackers within the manufacturing domain can be distinct from other domains [104, 105, 19]. This needs to be taken into account for any study on information security within manufacturing. We give a detailed discussion on suitable attacker models for manufacturing and discuss their properties. We state what assumptions the underlying attacker models share and how they affect study on information security within (connected) manufacturing. We make a selection of suitable attacker models that are used in the following chapters and throughout this thesis. Attacker models used within other domains, even if they are suitable and reasonable there, cannot be transferred directly to the domain of (connected) manufacturing. Although, the domain of manufacturing might share some similarities with other domains, e.g., that of classical IT such as within office equipment [48]. Thus, it is necessary to also highlight what the differences are and how similarities can be used for our evaluations. From there, we continue by elaborating on the attack scenarios these attackers might take when targeting manufacturing facilities (see Section 4.2). We give a detailed account on possible attack vectors different attackers (in correspondence with our defined attacker model) use to achieve their goals. From there, we define a description attack vector model that can be used by a variety of attack scenarios. These attack scenarios describe specific threats to target one or several protection goals.

From there, we discuss our conceptional system architecture (see Section 4.3). The system architecture takes the developed threat models into account.

4.1 Attacker Model and Assumptions

In this section we discuss the attacker models used within this thesis. Also, we make and discuss assumptions that are related to the attacker model where applicable. An overview on the threat landscape for connected manufacturing is given in Section 2.1.3. This is the background against our discussion of attacker models is done. Before beginning with a discussion on attacker types in (connected) manufacturing, we give an brief introduction to attacker models and their importance within the field of security.

The presence of an attacker or *adaptive adversary* [190] can be considered an important aspect in the science of information security. Thus, the capabilities and the behavior of an attacker are captured within attacker models. A common and widely used attacker model is the Dolev-Yao attacker model [161]. It is borrowed from the study of interactive cryptographic protocols. The attackers within the Dolev-Yao attacker is are omnipotent and control the messages sent within the network. This allows for the study of a wide range of possible attack vectors. This, however, brings some limitations with it as well [104, 105]. First, it can lead to disregarding the possible attack path attackers can follow during an actual attack. This can be a worthwhile field of study by itself [191]. Second, the possible motivations of attackers and the resources available to them are not covered by a Dolev-Yao attacker model. This can, however, be an important piece of information for risk management approaches [29]. Third, the applicability of an attacker borrowed from another domain may not be as high in the new domain. Our discussion in this paragraph hints already in that direction. Typically, attacker models designed for the target domain can offer a higher benefit in information security evaluations (e.g., compare [192] as an example from the automotive domain). Fourth, the heterogeneous nature of components, protocols, and technologies used in manufacturing as well as the diverse landscape of attacks can not be adequately captured by methods for cryptographic protocol verification. The assumptions of the Dolev-Yao attacker model are, thus, quite strong assumptions in that regard as they tend to generalize attack paths and skip over the capabilities of potential attackers. The Dolev-Yao attacker model is, despite its limitations, still frequently used when studying security within manufacturing [104, 19, 193, 194]. Thus, attacker models more suited for manufacturing environments are reasonable to be considered when discussing information security within that domain.

In the following sections, details on attacker models in (connected) manufacturing are discussed. The different types of attackers together with a discussion of them is given in Section 4.1.1. Building up from there, attack scenarios relevant to connected manufacturing are detailed in Section 4.2. The conclusion to this section is given by deriving security goals relevant for this thesis in Section 4.1.2.

4.1.1 Attacker Types in Connected Manufacturing

Attacker models in manufacturing typically contain a set of attacker types [104, 193]. Attacker types try to group attackers, however, their boundary may not always be clearly defined. For better differentiation, attacker types are matched with a set of attacker attributes that represent the properties of an attacker. These attributes can include an attacker's proficiency or equipment. The attacker attributes aim to capture the capabilities and properties associated with a specific attacker type. As discussed in the previous section, the Dolev-Yao attacker model demonstrates that attacker models need to capture the domain of study they are used in. For manufacturing, there is still no unified theory of an suitable attacker model [19]. Such an attacker model needs to encompass the challenges of the manufacturing domain as discussed in Section 1.1 and Section 1.2. In particular, the challenges resulting from the prominent use of CPSs in connected manufacturing need to be considered. For the remainder of this section we present a compilation of attacker types for manufacturing in literature [104, 193]. The attacker types are summarized in Table 4.1. It contains a denominator for each attacker type as well as selected corresponding attributes. The attributes are distinguished by their extend, namely, 'none', 'low', 'medium', or 'high'. For example, an expertise of 'low' extend indicated that this attacker type is not very experienced and only has basic knowledge relevant to the attack. In addition the attacker attributes, the protection goals threatened by each attacker type are given. A letter corresponding to the threatened protection goal indicates in which domain (in the context of CPS, see Section 2.1) the protection goal is mostly located. These are the physical domain (indicated by the letter 'P'), the virtual domain (indicated by the letter 'V') or both domains (indicated by both letters 'V/P'). For the remainder of this section, the different attacker types are discussed and an attacker model is defined based upon this discussion.

The first attacker model is the *basic user*. It describes someone familiar with automated execution of (pre-written) attack code. They have access to basic hard- and software that can be obtained in retail stores or through theft from an employer.

Table 4.1: Attacker types in manufacturing with corresponding attributes.

Attacker type	Expertise	Resources	Threatened protection goals
Basic user	Low	Low	None in particular
Insider	Medium	Medium	Availabilty (V), Confidentiality (V)
Cyber-criminal	Medium	Medium	Availabilty (V), Confidentiality (V)
Hacktivist	Medium	Medium	Availabilty (V), Confidentiality (V)
Nation state	High	High	Availabilty (P), Confidentiality (P)
Terrorist	Unspecified	Unspecified	Availabilty (P)
Competitor	Medium	Medium	Confidentiality (V/P)

The main distinguishing characteristic of the basic user is their lack of knowledge or resources as well as their missing motivation. The lack of motivation here refers to the absence of long or middle-term strategic goals like extortion of money. Synonyms for the basic user include *amateur hacker* or *script kiddie*. Especially under the later synonym, the basic user is often mentioned by popular news stories when trying to emphasize an attack that can be executed with relatively simply methods and little knowledge. Other attacker types are characterized by their larger skill set and their available resources. One type of attacker characterized by at least more resources is the *insider*. They have elevated access to a system as in contrast to the basic user. This access usually is achieved through their role as employee or contractor, i.e., their role inside of an organization. Basic users act outside of an organization and do not have these elevated access. Insiders can, especially in OT systems, cause significant material or personal damage. This is also typically considered their goal. In literature discussing attacks on OT systems, the insider is often synchronously referred to as *disgruntled employee* [13, 18]. One popular example for a disgruntled employee is the Maroochy Shire incident that occurred in 2000 [102, 195, 16] (see Section 2.1.3). Note that insiders typically are considered to also have better knowledge of the attacked system but may have only limited proficiency in information security.

One attacker type with more available resources and proficiency is the so-called *cyber-criminal*. They are individuals or a group of individuals experienced with security. This experience includes the ability to exploit known vulnerabilities and, to some extend, the discovery of zero day vulnerabilities. Cyber-criminals are mo-

tivated by financial gain and aim to achieve this e.g., via blackmail, espionage, or sabotage. The recent trend in ransomware campaigns that also target manufacturing sites is evidence of professional cyber-criminals [13, 18]. The aim of such attacks are virtual components, i.e., the so-called *cyber* component of CPSs. When cyber-criminals successfully attack a system, especially the protection goals of availability and confidentiality are the scope of the attacks. Reducing or completely disabling the availability of a manufacturing component is cost intensive (see Section 4.3) and, thus, attractive to attackers solely focused on monetary benefits. Also, obtained sensitive information from a company or a private individual can be used as leverage to extort money via ransom notes. A well-documented, recent incident is the Norsk Hydro incident from 2019 [18] (see Section 2.1.3). An attacker type similar to the cyber-criminal attacker is the *hacktivist*. They are characterized by a similar skill set and a comparable amount of resources. They differ, however, in the goals they want to achieve. The main goal for an hacktivist attacker is some form of political agenda. To achieve their agenda, hacktivists tend to target the same protection goals, i.e., availability or confidentiality, as cyber-criminals. However, their aim is slow or stop politically unwanted processes or to release sensitive information.

An attacker type belonging to the most advanced types of attackers in manufacturing and OT systems in general are *nation states*. Under this term, organizations either belonging to or sponsored by a nation or state are subsumed. They are characterized by almost unlimited resources and advanced, in-depth knowledge of the targeting systems. Nation state attackers are often presumed to target critical or public infrastructures like transportation, power, or water systems. Presumably the most prominent example for such an attack is the Stuxnet malware campaign from launched from 2009 to 2011 [23] (see Section 2.1.3). It is worth noting that direct evidence of such attacks are rare and controversially discussed. However, the nation state attacker type is useful in the study of security incidents for manufacturing as they allow for modeling powerful attacks with far reaching consequences. This is also the goal for a very closely related attacker type, i.e., the *terrorist*. The main distinguishing dimension is their motivation. Terrorist attackers are motivated by causing a highest possible effect on the psychology of the target, e.g., by spreading fear or uncertainty. Thus, terrorists in the context of attacker profiles for manufacturing are not motivated by financial gain. Finally, we discuss an attacker type that is discussed frequently in information security research for manufacturing systems: the so-called *competitor*. It refers to a market competitor of another OT system

operator, e.g., the operator of a plant. They are, according to literature [193, 194], mostly motivated by attacking the protection goal of confidentiality, i.e., obtaining intellectual property from their target. As we showed in our previous work [46, 47] and discuss in Section 7.2.1, this can include, for example, the geometry of a product being manufacturing by the OT system.

For this thesis, we adopt a subset of the above mentioned attacker types. The first attacker type we adopt is a basic attacker with some additional information on the target. This is best represented by the Insider attacker type. With this attacker type, we study attacks that are comparatively simply to execute but may have a clearly visible and sometimes devastating attack impact. The impact of the attack is studied locally whereas the Insider may use remote and local attack vectors. The Insider aims at causing an immediate, noticeably impact that reflects a monetary cost as high as possible giving its limited time to prepare and execute the attack. The sophistication ot the attack is traded for broad applicability. However, the Insider is mostly focused on a a specific collection of OT devices located within a closely defined area. In our case, such an area includes the premises of manufacturing sites. The second attacker type we adopt for our attacker model is the attacker type of the Nation State. With this attacker type, we model and execute more advanced and complex attack vectors. These attack vector can include elaborate attacks that are carried out over a prolonged period of time. The impact of the attack is usually higher than with the attacks carried out by the Insider. The most striking similarity between the Insider and the Nation State attacker types is their focus on harming or violating the protection goal of availability. Both of the selected attacker types use a combination of local and remote attack vectors. Their targets are located in the physical and the virtual part of the CPS located within the targeted OT systems. Specifically, they mainly threaten the protection goals of availability within our attacker model. The third and final attacker type considered in our attacker model is the Competitor. In contrast to the other attacker types, the Competitor focuses on violating the protection goal of confidentiality and uses remote attack vectors. Consequently, the Competitor threatens only protection goals of the virtual part for CPSs.

To summarize, our attacker model for studying security incidents in connected manufacturing is comprised of three attacker types: Insider, Nation State, and Competitor. This attacker model allows us to cover a variety of attack vectors that are considered to be relevant for connected manufacturing (and OT systems in general).

Furthermore, our attacker models give a varied balance of attackers' capabilities and allow for the study of a broad range of attack vectors.

4.1.2 Information Security Goals

Security goals or protection goals are the desired outcomes or objectives that an organization or user aims to achieve in order to protect its assets, data, and resources from unauthorized access, use, disclosure, modification, or destruction [122]. Information security is concerned with protecting these major protection goals. To achieve this, information security goals for are defined as abstract concepts. These concepts include the so-called CIA-triad, i.e., confidentiality, integrity, and availability. Confidentiality refers to the protection of information from unauthorized disclosure. Confidentiality ensures that sensitive information is only accessible to authorized individuals who have been granted appropriate access permissions. This can be achieved through measures such as encryption, access controls, and secure storage of data. Integrity refers to the protection of information from unauthorized modification or destruction. Integrity ensures that data remains unchanged and accurate throughout its lifecycle, and that only authorized individuals are able to make changes. Measures such as hash functions, digital signatures, and message authentication codes can help ensure data integrity. Availability refers to the protection of information and systems from unauthorized disruption or downtime. Availability ensures that data and systems are accessible to authorized users when needed, and that they are resilient against failures and attacks. Measures such as redundancy, fault tolerance, and disaster recovery planning can help in ensuring availability. Note that other protection goals such as authenticity or non-repudiation are also mentioned in literature. For our purposes, the protection goals of the CIA triad are sufficient and are the basis for further discussions within this work.

The protection goals discussed in the previous paragraph are general and are applied to various domains. However, the application of the protection goals can vary between two different domains. For connected manufacturing and digital twins with their strong reliance on CPSs, particularly the domains of information technology and operational technology and the differences between them are of interest [104, 19, 60]. The IT domain refers to the use of computers, networks, and software in the context of an organization's business operations, while OT refers to the use of technology to monitor and control physical systems and processes, such as those used in manufacturing, energy production, and transportation. In the

context of IT, the CIA triad focuses primarily on the protection of digital assets such as data, networks, and software systems. The goals of confidentiality, integrity, and availability are achieved through measures such as encryption, access control, data backups, and disaster recovery planning. The primary security threats in the IT domain is from cyber attacks, such as data breaches, malware infections, and denial-of-service attacks [13, 25]. On the other hand, in the context of OT, the CIA triad also focuses on the protection of physical assets such as machinery, equipment, and industrial processes. In combination with the property of CPSs that combine digital and physical assets, proper protection of OT equipment offers its own challenges [15, 20]. The primary threat to OT security is from physical attacks or accidents, such as equipment failures, power outages, and environmental disasters [13, 18]. Literature frequently makes mention that information security is lacking behind in (connected) manufacturing in specific and in operational technology in general [58, 14] (the main reasons for this from our perspective are outlined in Chapter 1).

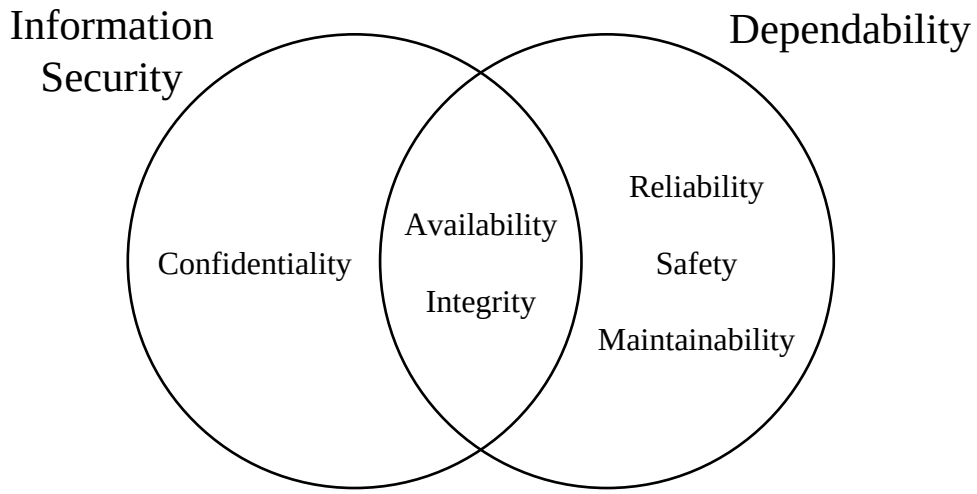


Figure 4.1: Venn diagram showing overlapping concepts of goals between the domains of information security and dependability.

In addition to the CIA goals of IT, OT security in manufacturing also emphasizes the goals of safety, reliability, and maintainability [48]. These are not strictly goals of information security but are also of importance to plant operators or OT operators [20, 9]. The goals of CIA triad are located in the domain of information security whereas the others are located within the domain of *dependability*. There

exists, though, some overlap between the domain of information security and dependability in manufacturing [48]. Availability and integrity are goals related to the design of secure *and* dependable systems. This relationship is shown by the Venn diagram in Figure 4.1. Here, the goals of information security (on the left hand side of Figure 4.1) are shown in relation to those from dependable systems (right hand side). As it can be seen, there are two attributes shared by both domains: availability and integrity (intersection in the middle of Figure 4.1). Availability, from the perspective of dependable systems, encompasses the readiness for authorized and correct service and is often seen as the most relevant protection goal in manufacturing by OT operators [20, 18]. Therefore, the assumed order of relevance of the protection goals for office information technology (IT), i.e., CIA, is often depicted in a reversed manner for OT, i.e., AIC. Integrity is the second shared attribute of information security and dependability. Within the context of dependability, it encompass the absence of improper system alterations. Our definition of *data* integrity given above is more narrow as it focuses on the stored data within a system whereas integrity of dependable systems encompasses the entire system or target of evaluation (TOE). For the remainder of this work, we adopt the broader definition of integrity from dependable systems.

There are three attributes remaining from dependability that are only referred to inside that domain. Reliability describes the continuity of correct service that occurs when the system is ready (or available) for correct service. A manufacturing system must be reliable in the sense that it should perform its intended function correctly and consistently over time. However, this also means that the system must be able to handle variations in input materials, as well as changes in environmental conditions (such as temperature, humidity, etc.). This can be achieved through robust design, effective testing and validation, and ongoing monitoring and maintenance [48, 60]. The goal of reliability is closely related to the goal of availability. Availability is, however, primarily concerned with ensuring that a system is accessible and operational when needed, while reliability is primarily concerned with ensuring that the system performs its intended function correctly and consistently over time. While the two concepts are related (e.g., a highly available system is usually also highly reliable), they are distinct and require different approaches to achieve. Our focus here is, as we argue below, on the availability of manufacturing systems. However, reliability may also be an indirect target of attack [164, 196, 55] (see also Section 4.2). Safety means the absence of harm for the user and the absence of catastrophic events. A

manufacturing system must be safe to use, meaning that it should not cause harm to workers or to the environment. This can be achieved through appropriate safety mechanisms, such as sensors and alarms, as well as through effective training and operational procedures. Safety is sometimes confused with information security within the context of OT environments [15]. Safety does not protect from the *intend* of causing harm or catastrophic events whereas (information) security does. The effects of a safety-critical incident either caused by an information security incident or otherwise may be similar or identical though. Finally, maintainability describes the property of a system to undergo repairs or to be modified. This means that manufacturing systems should be easy to modify or update as needed. This can be achieved through effective documentation, modular design, and other features that make it easy to troubleshoot and repair the system when issues arise. It is worth noting that even if maintainability is given for a manufacturing system, updates to software used within that system are not applied on a regular basis [18].

Processes in manufacturing are tailored to meet the requirements of dependable systems [197]. As we discussed above, the similarities on a conceptual level may lead to some of the protection goals of information security are captured by existing operations in manufacturing. However, even if these overlaps exist, they are limited in their scope, not targeted to address the specifics of information security (in contrast to dependability), and do not cover all protection goals of information security. For those reasons, we argue that an approach focused on protecting the availability of a system is a reasonable basis for studying information security within manufacturing. Thus, availability is the main focus for the attack scenarios we study within these work (see Section 4.2). Other protection goals from information security are also discussed, however, availability is considered the most relevant subject of for studying. The protection goals of integrity and confidentiality are also part of our scenarios, however, they are violated in order to threaten the availability of a system.

4.2 Attack Scenarios for Connected Manufacturing

In this section we discuss attack scenarios for connected manufacturing. This discussion is related to the attacks discussed in Section 2.1.3 but aims at identifying common patterns in cyber attacks. From there, we derive attack scenarios that are used by us for evaluation of this thesis. In Chapter 1 we describe the properties of the domain of manufacturing that make information security challenging within that domain [12, 16, 18]. We give a brief summary on that challenges here, for more details refer to Section 1.1 and Section 1.2 specifically. Manufacturing equipment is subject to high run-times in the field, the exchange rate of systems, components, and protocols is slow. Naturally, manufacturing equipment is operative over 20 years until replacement with new equipment. Also, the machines and devices may be operated 24 hours a day for seven day a week (i.e., in full time). This results in a huge number of legacy devices. These devices might be designed and shipped before the widespread adoption of connected manufacturing in modern plants. Thus, outdated software versions with limited or none possibility for updates or patches are widespread on the shopfloor. This makes OT equipment in general more vulnerable than IT devices. Connecting these OT devices to public networks increases the attack surface even if the OT devices are not connected directly to those networks. Information security controls for mitigating of these risks may also be difficult to implement on the OT devices at risk. This is due to the limited available computational resources on those devices. This is partially the result of soft and hard real-time requirements, which is crucial to oblige in manufacturing [21, 22]. These conditions found in modern manufacturing environments are part of on rapidly evolving threat landscape [13, 18] (Section 2.1.3). New threats and vulnerabilities are emerging constantly, making it challenging for manufacturing companies and plant operators to keep up with the latest risks and developments. Plant operators, however, are usually in competition with each other or their relationship even to their suppliers are marked by distrust [46, 47]. The nature of these relations make a combined effort in regards to information security in manufacturing on the side of the actual users of manufacturing equipment difficult. To this day, there are no unified attacker models or threat catalogs for manufacturing. This lack of standardization in threat modeling approaches in manufacturing makes comparison between threat models and data exchange across organizations difficult. This lack of standardization can lead to a fragmented approach to information security, making it difficult to address

systemic vulnerabilities. This is also observed in the construction of information security testbeds (see Section 3.1) as most authors are not concerned with defining an attacker model for OT systems [104]. This also has consequences on the definition of attack patterns as those attacks are mostly defined on an ad hoc basis. There is, however, some overlap in the description of attack patterns within those testbed related publications that include attack modeling [38, 156].

Attacks on manufacturing systems are dependent on the capabilities of an attacker (see Section 4.1). Thus, different attack patterns can be related to specific attacker models [19]. It is imperative to possess a clear comprehensive view of the distinct attack typologies to construct and evaluate robust defense mechanisms. However, it is also crucial for testbed developers and OT security practitioners to refrain from overestimating the significance of single attacks incorporated within the testbed [156]. Thus, we continue by providing a description of abstract attack vectors relevant for manufacturing. We conclude with a general descriptive framework on how attackers in general behave before and after a breach in the company network has occurred. For this, we distinguish between regular attacks and more sophisticated attack vectors, i.e., advanced persistent threats (APTs). This is the basis for developing our attack scenarios for evaluation of our concepts within this thesis.

4.2.1 Attack Vectors

Attacks in manufacturing equipment can be classified into two categories: network-based attacks and physical attacks [156]. This dualism reflects the properties of CPSs as they too are digital components that can act in the physical world, e.g., by manipulating a robotic arm [105, 28, 55]. For network-based attacks, five different attack vectors can be observed from literature [53, 19, 156].

The first network-based attack is the reconnaissance attack. This attack serves as a preparation point for other subsequent attacks. The reconnaissance attack is not intended to carry out any malicious source code or control commands. Rather, a reconnaissance attack gathers information about a target system or network in order to identify vulnerabilities that can be exploited in later attacks. This can be achieved by a number of different tools and methods, e.g., by scanning for open ports and services or social engineering tactics. The goal of the reconnaissance attack for the attacker is to gain an understanding of the target's IT and OT infrastructure. A breach into the infrastructure can occur after reconnaissance is finished [198]. This results typically in the execution of other attacks. It is worth noting though

that even passive observation or minimal interaction with an OT system can lead to potentially disastrous results. In a report on performing a penetration test for a robotic arm, the authors of [199] report that they executed a ping sweep to identify IP addresses used by the device. However, the controller of the robotic arm was still in standby mode and reacted to the ICMP messages received by the ping sweep by during its arm in a rapid movement by 180 degrees. If human personal were to be standing next to that arm, severe injure may be the result.

A prominent type of attack in IT networks that also occurs in manufacturing is the man-in-the-middle (MITM) attack. In a MITM attack, the attacker positions themselves between two systems that communicate together [122], e.g., between a PLC and the MES that are part of a production process. The MITM attacker intercepts communication between the two systems, which believe they are communicating directly with each other. The attacker then is able to eavesdrop on the conversation or alter the communication without either system noticing. This position can allow the attacker to manipulate or disrupt the production process, access intellectual property, or compromise the safety of the plant. For example, an attacker could intercept communication between sensors and a control system to manipulate the data being collected, causing the control system to make incorrect decisions based in the manipulated data or take destructive actions (this was the case for the Stuxnet malware [23], see Section 2.1.3). A MITM attack can be seriously complicated by applied transport layer encryption such as TLS. However, as discussed above, applying modern encryption to devices with limited resources and hard real-time requirements may not always be feasible or possible at all [21, 22]. The same can be said about other security controls that can counteract some common attack patterns.

The sending of malicious data as described above is referred to as an injection attack. A MITM attack in manufacturing can be a pre-requisite for injection attacks. More general, an injection attack in manufacturing is an attack in which an attacker injects malicious code or commands into a target system. Often, this exploits vulnerabilities in the system's input validation or data handling processes. An injection attack in manufacturing has wide range of potential targets. These potential targets include, for example, PLCs, SCADA systems, MES, or human-machine interfaces (HMIs). To continue the Stuxnet related example from above, an attacker could also inject commands into an HMI to manipulate the information shown on the display so a human operator is unaware of changes by the malware to the OT system. As

seen, injection attacks can have serious consequences, including a disruption to the production process and even compromising the safety of human personal [200].

An attack of arguably lesser sophistication but with similar damage potential in manufacturing is the replay attack. It is a type of attack where an attacker intercepts and re-transmits data, e.g., as a MITM, in an attempt to manipulate the system into accepting it as valid [122]. The attacker replays previously captured data packages to perform unauthorized actions or gain access to restricted systems or data. In a manufacturing environment, any manipulated package, either if altered or not, can have drastic results on the production process. A replayed control command can have the same effects as an injected package if sent to a device. For example, an attacker could record communication between a MES and a PLC that controls a conveyor belt. The stop and start signals sent between the two devices can be recorded by the attacker and replay at a later point in time. That replayed communication can cause the conveyor belt to stop or start unexpectedly, potentially damaging equipment.

The final network-based attack for OT systems frequently discussed in literature is the Denial-of-Service (DoS) attack [156]. By a DoS attack in manufacturing an attacker attempts to disrupt the normal operation of an OT system, network, or device. This is achieved by sending a large volume of traffic or other requests, more then it can handle, to the device. This results in the system or device being unavailable to users or other devices, i.e., a denial of service. As discussed in Section 4.1.2, the resulting loss in availability is considered a bad outcome by plant operators [20]. For example, in a manufacturing environment a DoS attack could involve an attacker flooding a production system with a large volume of traffic or requests. In an environment with soft and hard real-time requirements, an unexpected amount of messages can result in the system becoming overloaded and unable to continue to function properly. An unexpected shutdown of a system can cause severe damages and disruptions to the manufacturing process [9]. This concludes the discussion of the network-based attacks.

As seen by the examples given above, an attack resulting from the network (i.e., the digital part of a CPS) can affect the real world rather tremendously. Therefore, it has become an established procedure in the study of information security for manufacturing to distinguish more specifically between the networked part of a production environment and the physical manifestations of the concrete manufacturing process [19, 156]. For this, a set of three physical attack vectors are identified. Their goal is to describe how OT systems can be attacked in order to alter the phys-

ical process controlled by the OT systems (the physical process here corresponds to Layer 0 of common industrial automation model [82, 83, 81], see Section 2.1).

The first of these physical attacks is the direct damage attack. How the name of this attack suggests, its intention is to cause immediate damage to manufacturing equipment or infrastructure with the result of production downtime, equipment damage, and potential injury of humans involved in the manufacturing process. Direct damage attacks are intended to cause a maximum of damage within a short period of time. These attacks are carried out through significant alterations in the physical process causing the industrial systems to enter an undefined or unsafe state. An example of a direct damage attack is downtime resulting from ransomware like in the incident at the Norwegian manufacturing company Norsk Hydro [18, 111] (see Section 2.1.3). In this example, an attacker encrypts critical manufacturing systems, either as targeted attack on OT systems or by accident, and demands payment in exchange for the decryption key. This can result in production delays or even a complete halt of production if critical systems are affected as it was the case with Norsk Hydro.

The second physical attack is called device manumission. This is the only attack vector discussed here that also required physical access to the attacked OT system. That means, that the attacker or its malware needs to be located inside the target facility, can move within the system or facility, and access the system undetected. When this is possible to the attacker, they then tamper with the target device to manipulate the data stored on the device. The attack aims to induce incorrect values for measurements taken by the device, e.g., for a sensor. The results of the attack can be the same as discussed so far. An example for such an attack is the Maroochy Shire attack in Australia [102, 13, 18] (see Section 2.1.3). Here, the former employee of the water treatment company also required physical access in case of physical proximity to the OT systems that were attacked by radio waves. Counterparts to device manumission that do not require psychical access to the facility are, for example, network-based attacks like injection or replay attacks.

The above described attacks, both networked and physical attacks, are all, at least to some degree, available to the third of the three physical attacks discussed by us: the stealth attack. A stealth attack in manufacturing is an attack that is designed to stay undetected within the targeted system over an extended period of time. Those attacks can be difficult to detect because they are developed with capabilities for evading security controls as well as other controls such as safety systems or supervi-

sion by human operators. The most prominent example for a stealth attack where the attackers remained in the targeted systems for a prolonged period of time is arguably the aforementioned Stuxnet malware [23] (see Section 2.1.3). Stealth attack like Stuxnet are also considered to be advanced persistent threat (APTs) [198]. APTs often involve a series of stealth attacks that are carefully planned and executed to achieve a specific objective. An APT in a manufacturing environment can have serious consequences, as witnessed by the large scale destruction of OT systems over several months by Stuxnet. Attacks like Stuxnet and APTs in general are often carried out by well-funded and highly skilled attackers with access to advanced tools and techniques.

To summarize, manufacturing industry is a critical sector of industry that is vulnerable to various attack vectors. As discussed in this section, these attack vectors include networked-based and physical attacks. The impact a successful attack might have can be severe. They include downtime in the production process, compromise of sensitive information, and the potential to cause physical harm to equipment as well as humans. In the following section, we discuss how the attack vectors can be expressed within a generic model for attack vectors within manufacturing.

4.2.1.1 Generic Attack Vector Model

In this section, we take our discussions on attack vectors in manufacturing into account and develop a generic model for attack vectors in manufacturing. This model sets the frame for our subsequent introduction of our attack scenarios used for evaluation within this thesis (see Section 4.2.2 as well as Section 6.4 and Section 7.5.2). The generic attack vector model allows for comparison of different attack vectors. This may support a standardized approach to identifying and categorizing potential attack vectors. The model itself is based on generic attack vector models such as so-called 'cyber kill chains' [53, 18] and OODA (observe, orient, decide, act) loops [198].

Figure 4.2 shows the generic attack vector model we use to describe cyber attacks in manufacturing [53, 198, 18]. The figure is best viewed from top to bottom as this reflects the chronological sequence of events. These are the individual steps an attacker takes during execution of an attack vector. They are represented by rectangles and connected by arrows indicating the transition between the single steps. Furthermore, dashed lines indicate logical boundaries in the figure. Beginning from the top of Figure 4.2, the initial step in the attack chain is the planing step. This step

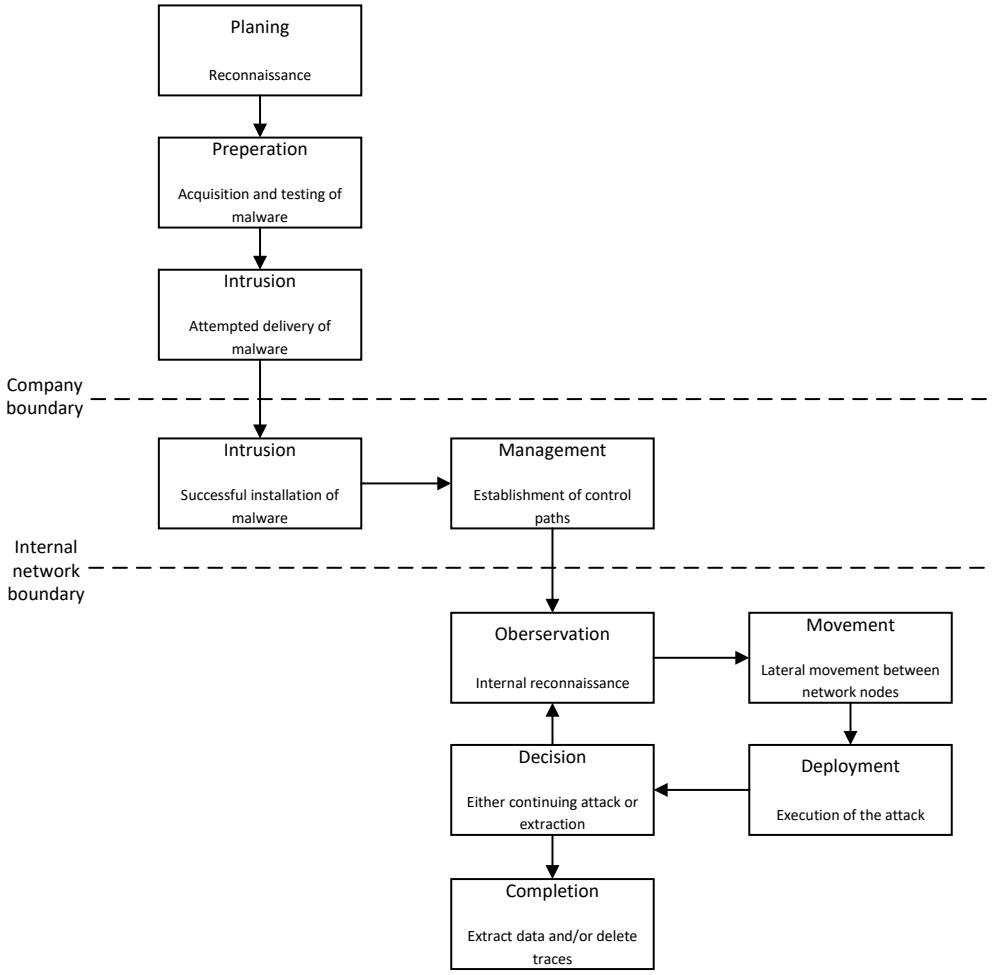


Figure 4.2: Generic attack vector for attacks in manufacturing.

is closely related to reconnaissance as discussed above. The planing for attacks on manufacturing is the process of gathering information about a manufacturing facility and its computer systems in order to identify vulnerabilities and potential targets for a cyber attack. This can include a variety of techniques and methods, e.g., port or IP scanning. Overall, the planing step and the connected reconnaissance activity is a critical step for an attack on manufacturing systems, as it allows attackers to better plan and execute an attack on the facility. After the initial reconnaissance step, the preparation of the actual malware begins. This can involve a number of different activities. These can range from acquisition of a malware sample, e.g., via

malware-as-a-service [201], or the independent development of the malware up to testing of the malware. For example, the Stuxnet malware was targeted to cause a specific effect within a certain type of OT equipment which might be difficult to achieve without testing the malware beforehand in a lab setup [23, 107, 108]. Then malware is ready for deployment, the actual breach is performed. In Figure 4.2, this is represented by two subsequent steps as a two-step approach. During transition between these steps, the company boundary is crossed. This does per se not require physically crossing the border as a large number of cyber attacks can be carried out remotely. However, as seen in the Maroochy Shire incident, physical proximity may be required for some constellations [13]. Also, the Stuxnet malware required presumably transportation via an infected USB device to reach its target facility [23]. In most cyber in the recent years, however, the attempted delivery of the malware occurred automated and remotely. Specifically, phishing techniques were used targeting employees to interact with malicious links sent to them via email [25]. Generally speaking, it appears that human error is one or arguably the most relevant enabling factor within a cyber attack in manufacturing [15, 202, 9, 203]. After an successful intrusion attempt, the malware is installed on a system inside the company network. This system does not necessarily need to be the final target system but may serve as a beachhead into the facility. This is especially true for APTs. APTs are the main focus for the remaining steps within the generic attack vector model. When an attacker or APT successfully infiltrates a system, they will typically attempt to establish control paths. These are actions taken by the attackers to ensure their access to the system and to carry out their objectives. Control paths are the communication channels or methods through which the attacker can interact with the compromised system. This can be achieved through a variety of means, e.g., backdoors or command and control (C2) servers. C2 servers are remote servers or domains outside the company network and in control of the attackers, which use C2 servers to exchange messages with the compromised system and send control commands for the malware. Note that APTs may establish several control paths to ensure uninterrupted connectivity to the malware. Once control paths are established, the attacker can enter a loop within the generic attack vector model [198]. For this, the malware is considered to be active within the internal network, thus, crossing the logical boundary to that network. There, the malware initially starts with an observation step to plan its next actions within the infiltrated network. The observation step can be seen as a phase where reconnaissance is conducted inter-

nally within the network of the infiltrated facility. The methods used for this can be similar to those described above within the context of the initial planing step, e.g., network mapping, network scanning, or ICMP sweeps [199]. When a sufficient amount of information is gathered, the malware starts to laterally move within the network. This lateral movement describes a set of techniques used by attackers to copy their malware to other systems within the infiltrated network. This is done in order to gain access or move closer ot the targeted component. There are several methods that attackers can use for lateral movement. For example, vulnerabilities in other components such as protocols, software, or network services can be exploited. A popular target for lateral movement is the Active Directory (AD) infrastructure. This infrastructure, if it exists within the infiltrated network, connects a majority of personal computers via its AD servers. Compromising one of the servers that manages PCs located within the IT as well as OT network is what enabled the successful ransomware attack on the Norwegian company Norsk Hydro [18, 111] (see Section 2.1.3). Once an attacker has successfully moved laterally through a network, they find themselves within a position that enables them to execute the actual attack. This is done during the deployment step shown in Figure 4.2. Here in this step, attacks like the ones discussed in the previous section are executed. These can include a combinations of different attacks. For example, an injection of manipulated control commands onto the bus in order to cause a production system to cause damage within the manufacturing device [60]. The goal for attacks carried out during the deployment step are manifold and can range from physical destruction of manufacturing equipment to exfiltration of intellectual property [19, 46, 47]. After a goal is achieved, the attackers, supported by their C2 servers, reach a decision whether or not to continue with the attack. Some APTs may remain within the attacked networks for several months without detection and cause a significant amount of damage especially when they remain undetected [23, 107, 108]. If the decision for completion of the attack is reached, the malware takes actions in order to extract data (if this is a goal of the attackers). Finally, the malware may also take steps of self-destruction, i.e., deletion of the malware and log files [103]. This way, potential victims may not even realize they were the target of an cyber attack.

4.2.2 Attack Scenarios

In this section, we present and discuss the attack scenarios for evaluation within this thesis. The attack scenarios follow the generic model of attack vectors in manufac-

turing as discussed in the previous section. However, not all attack scenarios cover all aspects of that model. Rather, the attack scenarios focus on specific steps and actions taken by the attackers. The attack scenarios are located in a discrete assembly setup. This refers to our discussion on a discrete-event dynamic system (DEDS) and continuous-variable dynamic systems (CVDS) in Section 3.1.1. To summarize, industrial assembly processes can be categorized by their flow of material [164]. In a continuous process ,i.e., in a CVDS, material is constantly flowing through the manufacturing environment, e.g., in water treatment or other chemical processes [165]. In discrete processes, i.e., in a DEDS, the flow of material is quantifiable. Examples are automotive assembly, furniture manufacturing, or sorting operations performed by robots within these domains [9, 55]. Stop-and-wait states are typical to occur in discrete processes. A combination of both continuous and discrete processes, can be encountered in manufacturing as well. These are characterized by an interruption of the continuous material flow, where discrete operations need to be performed. This is the case for, e.g., pharmaceutical or metal-alloy assembly. Correctly addressing these kind of processes requires the usage of simulators specific to this domain [30].

As mentioned, we consider discrete assembly processes for our attack scenarios as they are can be found in a DEDS. Studying these type of systems has advantages when focusing on information security evaluations with digital twins. Those advantages are discussed for the remainder of this section before describing the attack scenarios individually. Discrete manufacturing systems process individual units or events that can be simulated more easily [204]. In contrast, continuous manufacturing processes involve the flow of materials in a continuous manner [36]. Modeling this type of manufacturing processes typically requires more effort. The reason for this is that within a simulation of discrete processes, individual components can be modeled separately and studied individually. Simulations of discrete manufacturing processes can predict the behavior of the system under different scenarios more easily as they are usually not depending on complex physical or chemical interactions. This improves the scalability as well as the control of a simulated discrete manufacturing process. Discrete manufacturing processes are often more scalable than continuous processes. That is because discrete processes can be more easily expanded or modified without disrupting the entire process. Adding more or less distinct units into the system results in queuing related challenges but does in general not threaten the stability of the entire manufacturing process. As a result, discrete processes tend to be more flexible in accommodating changes to the process

improvements. They can be modified or reconfigured with lesser effort as adjustments to the experimental setup do not require a re-calibration of the parameters for a continuous process. Furthermore, discrete manufacturing processes are also more controllable than continuous processes. In a discrete process, each step can be precisely controlled, and any deviations from the desired outcome can be quickly identified within a simulation environment. From there, discrete manufacturing processes can be observed better by the experimenter. This improves the possibilities and quality of the data analysis. Simulations of discrete manufacturing processes generate more structured data than continuous processes. Each manufacturing step interacts with a discrete, distinct unit and each simulated manufacturing step, thus, generates data that is directly associated with that unit. This data can be analyzed more easily to identify patterns and anomalies [49, 109]. With simulation of continuous manufacturing processes, it can be challenging to associate data with specific materials and steps within the process. The ability to generate and analyze structured data allows for data-driven applications such as differential privacy to be studied more effectively [130]. As discrete manufacturing processes are typically composed of distinct steps, also eases studying the impact of attacks as it can be associated directly to the individual step [60]. Also, security controls can be attached more precise, which improves the possibility for studying their effectiveness [21].

For those reasons, the attack scenarios studied by us benefit from the simulations of discrete manufacturing processes. The attack scenarios are discussed in detail in the following subsections. A brief overview about the attack scenarios is given in Table 4.2. Here, the attack scenarios are compared by their used attacker model, their main attack vector, and the threatened protection goal within the OT systems. As seen, the attack scenarios cover the range of attacker models discussed in Section 4.1.1. Except for the basic user with low expertise and resources, the attacker models in the attack scenarios represent those attackers with medium and high expertise and resources. This allows for the study of more sophisticated attacks as those represent the attacks that are most relevant to manufacturing at the moment [12, 18]. The attack vector is a summary of the main attack vector used within the scenario (cf. Section 4.2.1.1). Typically, we focus on a specific step within the attack vector that we consider most promising for new findings. Finally, the threatened protection goals for the manufacturing system(s) targeted are listed. Note that typically, a successfully executed attack requires the violation of more than one protection goal. It is, however, worth considering the attempted goal of the

attacker [108]. Thus, the threatened protection goals in Table 4.2 are those related to the ultimate goal of the attacker. Each of the described attributes is discussed in the chapter corresponding to the individual attack scenario in the following.

Table 4.2: Attack scenarios.

Attack scenario name	**Attacker model**	**Attack vector**	**Threatened protection goals**
Attack Scenario 1: Sorting	Insider (disgruntled employee)	Direct damage	Availability (of manufacturing equipment)
Attack Scenario 2: Sawing	Nation state	APT	Integrity (of the production process)
Attack Scenario 3: Data Sharing	Competitor	Data Leakage	Confidentiality (of intellectual property)

4.2.2.1 Attack Scenario 1: Sorting

The first attack scenario takes aspects from the Maroochy Shire incident into account [102, 13] (see Section 2.1.3). Furthermore, this scenario is motivated from a real-life case study on the security of manufacturing parameters located within furniture manufacturing of kitchens [9, 69, 10]. The attacker profile is the insider [19], specifically an insider that can be considered a 'disgruntled employee' [193, 194] (see Section 4.2.1.1). The goal of the attackers is to cause as much damage as possible. They are not interested in launching a long threat campaign against their target (which is their former or current employer in this scenario). Furthermore, remaining undetected is not their primary concern. The initial steps of the generic attack vector model for manufacturing as seen in Figure 4.2 are, therefore, considered as already accomplished. The attacks are assumed to have conducted a sufficient amount of reconnaissance during their time of employment at the targeted facility. Also, an intrusion attempt is trivial as the attackers are still in possession of the required credentials and other methods for authentication. Also the sophistication of the attack is not given as no prolonged threat campaign is planed and avoiding detection is not of the uttermost importance to them. Therefore, the OODA (observe, orient, decide, act) steps are simplified to the actual deployment of the attack. For de-

ployment of the attack, no sophisticated malware is used by the attackers. Rather, they misuse their access privileges and detailed knowledge of the target facility for causing damage, e.g., by sending manipulated control commands [60].

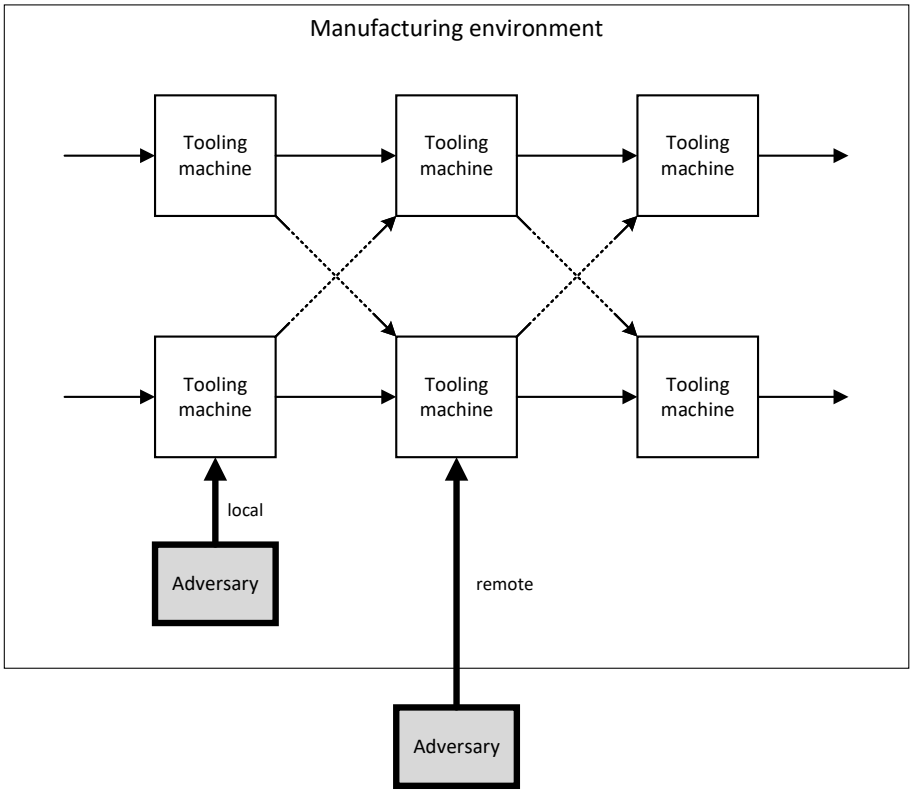


Figure 4.3: Attack on manufacturing environment with the intend on causing direct damage.

The principle method of the adversary is shown by Figure 4.3. The tooling machines of the targeted manufacturing environment are setup as production lines with transitions between the individual tooling machines as well as the production lines. The adversary is located either within the facility or outside it. The remote attack vector requires, however, some proximity to the manufacturing environment.

The attacks carried out in this scenario directly target the availability of the assembly line making a successful attempt of the attack highly risky for plant operators [20, 191] (see Section 4.1.2). A service failure of one of the tooling machine in a production line causes the entire production line cease operation. Flexible manufacturing setups do exist but are not used in every environment [60]; furthermore, man-

ual re-organization of the production process is costly and usually infeasible [18, 111]. The threat to the availability of the assembly line is realized by manipulation of a pick-and-place task, i.e., sorting of raw material (e.g., Figure 5.2 provides a model for robotic arm performing a pick-and-place task). Sorting tasks like pick-and-place are common place in manufacturing processes, for example, also in furniture manufacturing [9]. As a piece of furniture typically consists of individual wooden parts, the individual parts need be sorted and prepared accordingly for processing. This is illustrated exemplary by the manufacturing processes modeled by Figure 2.4 and by Figure 5.1. The attackers aim at minimizing their effort and maximizing the impact of the attack at the same time. Therefore, they attack the integrity of the communication between the control devices [21]. The attackers finally achieve their goals by direct manipulation of the sorting parameters within the MES and, thus, influencing the devices interacting with the assembly process [60, 55]. Specifically, the sorting portion is targeted by the attackers. This is a viable attack strategy for the attackers as pick-and-place operations are useful for almost all production flows and their unavailability being a severe threat to continuation of service [205].

This scenario serves also as an initial testing use case for the implementation of our testbed as the complexity of the attack is low and the impact of the attack takes place rather immediately.

4.2.2.2 Attack Scenario 2: Sawing

Attack Scenario 2 is influenced by the detailed descriptions available on the Stuxnet malware [23, 107, 108] (see Section 2.1.3). The attacker profile is, thus, that of a nation state [19] (see Section 4.2.1.1). The goal of the attackers is to cause a significant amount of damage to the production process. In contrast to the attacker in the previously discussed Attack Scenario 1, the attacker in Attack Scenario 2 is launching a prolonged threat campaign with the intention of remaining undetected. Furthermore, the attackers here possess a large amount of expertise and resources allowing for more sophisticated attacks on the target facility. The steps related to planing and intrusion, as discussed in generic attack vector model for manufacturing in Section 4.2.1.1, are considered to be finished before the start of this scenario. The attackers accumulated a large amount of knowledge on their target and are in possession of a malware suited for their goals. The scenario starts with the actual intrusion into the target facility. The intrusion here is conducted by the standards means of phishing, which is the most common breach point for attacks on

manufacturing [203, 25]. From there, the APT, which is the center of this scenario, is infiltrating the facility and positioning itself in order to manipulate the industrial process of the facility.

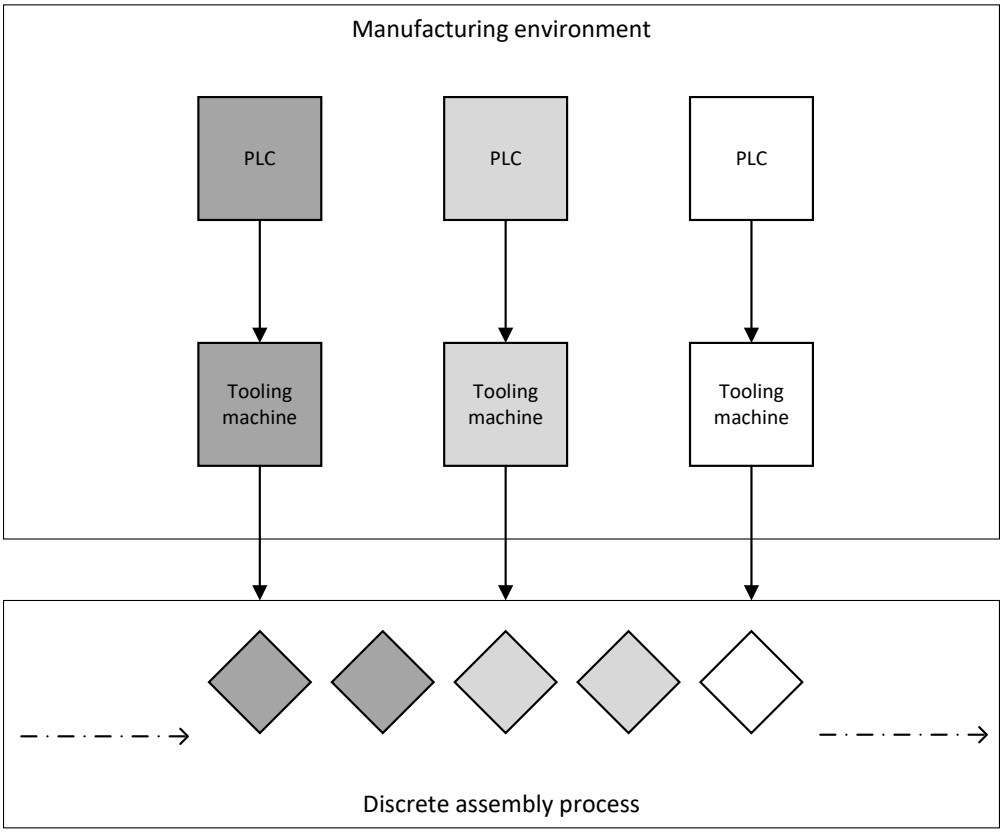


Figure 4.4: Attack on manufacturing environment with the intend on causing damage over time undetected.

The general idea for this attack vector is shown by Figure 4.3. The APT or malware is installed on the PLC devices and gradually affects the production quality over time, e.g., by altering production parameters. The darker the shading of an element in Figure 4.3 is, the further the production parameters deviate from the expected to malicious values. The PLCs control tooling machine and those tooling machines then alter the distinct units in the discrete assembly process [80, 81] (see also Section 2.1 for an introduction to the different layers of automation). This is also represented by the tooling machine and the produced units being shown in the same shades of gray than the infected PLCs.

In this scenario, the attackers aim at threatening the integrity of the product manufactured by the discrete assembly process [191]. A successful attack is expressed by a decline in product quality and an increase in product defect rate [164, 196]. The basic approach remains similar to Attack Scenario 1 (see Section 4.2.2.1). However, the attackers put much more effort into the design of the attack vector for this scenario. The aim of Attack Scenario 2 is to replicate an advanced persistent threat (APT) on the assembly line via subtle alterations of parameters; in this scenario, sawing operations are performed. These sawing operation are the process the attacker intends to sabotage. Sawing is an operation typically executed in wood-based manufacturing for all types of products [205]. The generalization of sawing is *cutting*, which is a frequent operation performed within subtractive manufacturing (SM). Therein, cutting refers to the removal of material of any kind, e.g., the cutting of metal through welding or via laser technology. When sawing, the machine tool conducts a series of movements as defined within its computer numerical control (CNC) program. This can relate to the intellectual property implicitly encoded within the CNC program, e.g., to product geometry. Sawing or cutting operations are typically not performed as the first or last step within a production process. These are usually pick-and-place or sorting operations as discussed in Section 4.2.2.1 (see Figure 2.4 and Figure 5.1 for a simple and more complex manufacturing process involving cutting). The attackers take a stealthy approach and try to remain undetected for a prolonged period of time to maximize their impact, i.e., affecting the largest possible amount of products assembled. Attack targets are the sawing robots or machine tools involved in the process.

The attack discussed above is one possible attack for affecting the product manufactured by an industrial process. However, possible attack vectors for this are plenty and it is currently not well understood by OT security research how they limit the product quality [53, 191, 55]. By studying this scenario, security-relevant interactions can be considered in addition to quality evaluations. Thus, implementation of proper security controls can be supportive in increasing overall product quality. The metrics chosen for evaluation for this scenario relate to ensuring the quality of the manufactured product [164, 196]. This is clear in the case of Attack Scenario 2, where product quality and product defect rate directly correspond to the quality of operation. Thus, studying the impact of attacks can be supportive in increasing overall product quality.

4.2.2.3 Attack Scenario 3: Data Sharing

In Attack Scenario 3, we discuss a scenario reported to us by manufacturers and users of tooling machines [183, 46, 47]. The attacker profile for this scenario is that of the competitor [193] (see Section 4.2.1.1). Competitors in this specific scenario are interested in acquiring intellectual property (IP) from another company. They have the knowledge of assessing the relevance of another company's IP. From the knowledge contained within the IP, a competitor could, for example, extract the type and volume of products manufactured by another company. This could further provide the competitors with an advantage as they could adjust their product line accordingly. IP may also hint at the quality of the discrete assembly process implemented by another company. However, accessing these information is for a competitor typically not possible as their expertise in information security is limited [15, 12, 18]. Therefore, Attack Scenario 3 introduces a setup for collaborative data sharing for optimization of manufacturing environments. This is a possible business use case for connected manufacturing [1, 70]. The platform where the collaborative data sharing is setup acquires data from different manufacturing environments and uses that data as input to optimization algorithms that require a large amount of training data. Several algorithms are possible, e.g., such related to condition monitoring or predictive maintenance.

An abstract architecture depiction for a platform that enables collaborative data sharing is seen in Figure 4.5. Here, different competing manufacturing environments allow the extraction of data sets from their tooling machines. The data sets are then stored within the platform. A malicious competitor may have the goal to gain illegitimate access to the data. This can be achieved by them either via eavesdropping on the data transmission or accessing the data at the platform's storage. The data sets in the platform's storage, however, are required to enable the analysis conducted at the platform as the analysis algorithm is depended on different large and diverse data sets.

One of the use cases of condition monitoring or predictive maintenance is chatter detection. Chatter detection is concerned with the detection of chatter vibrations that occurs within machine tools [206, 47]. Chatter vibrations can be caused by interaction of the machine tool with the workpiece or by the construction properties of the tooling machine. Also, further environmental causes can also influence or create chatter vibrations. Chatter vibrations can influence to product quality in an undesirable fashion, e.g., the product's geometry or its surface finishing. This

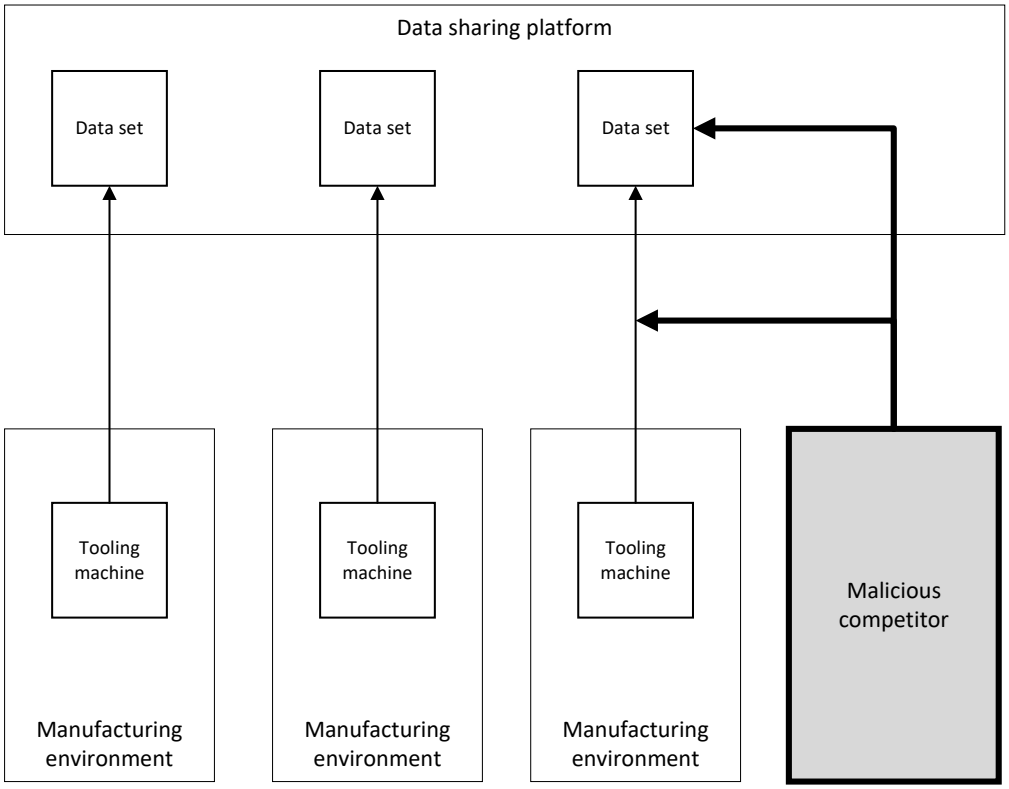


Figure 4.5: Platform for collaborative data sharing among competitors with malicious competitor.

can lead to an increase in cost due to frequent re-calibration or the machine tool or can also increase the rejection rate of products with insufficient quality. Chatter vibrations can be reduced by proper configuration of the tooling machine. For environments that produce the same or similar products, a one-time configuration of the tooling machine can mitigate chatter vibrations to a manageable degree. However, when we consider customer-individual production [1, 11], this is not easily achievable as the range of possible products can change more often. Each of these changes in product manufactured then requires a period of trial-and-error to deduce a proper configuration of the machine tool. This is not always economically reasonable. For the reasons stated above, different chatter detection methods are discussed in literature [206, 207]. The methods have in common that they require data from the tooling machine in order for automated derivation of proper machine configurations. The more data available, the better some of these methods can perform. Aggregated

data from different sites can further increase the outcome of some methods. The data can be processed by an external service provider with the proper methodological knowledge. The major threat here is that the data extracted from the manufacturing site is accessed by an unauthorized party. This unauthorized third party is in the context of Attack Scenario 3 one or several competitors. They can access data by accident or intentionally, especially if the external provider is (partially) under their control or does not implement sufficient protection mechanisms. This is a relevant threat considering successful attacks on platform providers that also led to data leakage [208].

4.2.2.4 Attack Scenarios Specific to Digital Twins

In this section, we briefly highlight the threat landscape related directly to digital twins. That is, we introduce some of the security-related challenges in regard to the exploitation of the digital twin concept. The discussion provided in this section by us is complementary to the general discussion of information security threats in (connected) manufacturing given by Section 2.1.3. No reported attacks on digital twins are known to us at the moment. One reason for this is that the concept of digital twins is still being development and true digital twins are not used widely or at all in productive use cases [28, 42]. However, a discussion on digital twins and how they can provide value to the overall state of information security within connected manufacturing, as provided by us in this thesis, needs to at least discuss the potential security issues that might arise from the usage of digital twins.

Digital twins themselves can, as any component within a network, pose a security risk [209, 210]. In particular, attackers might see digital twins and their connected systems as a valuable target within the future [149]. The attacks on digital twins can be directed at the twin itself, the physical twin, or the data connection between them [160]. A variety of different attack scenarios are possible, for instance, see the work of [209] for an overview. For our understanding in this thesis, we provide some examples that illustrate the potential threat that might occur from digital twins. A straightforward attack for an attacker is to achieve unauthorized access to the digital twin. Depending on the use case the digital twin is employed in, this could mean that also the physical twin can be manipulated or even controlled by the attacker. Within such an attack scenario, disruptions to service quality, damage to physical objects, or even harm to individuals can occur [60]. For example, a robotic arm could be manipulated to execute abrupt and unexpected behavior towards indus-

trial equipment or human operators [199]. Also, digital twins generate and contain vast amounts of data and sensitive information [28]. This sensitive information can include such information about the physical object or system they represent. If the data processed and stored by the digital twin is not protected adequately, it can be leveraged or extracted by an attacker. For instance, compromising a digital twin of a production line could expose intellectual property of the company operating the line [66]. Furthermore, successful attackers that achieve access to the digital twin and its related data can tamper with that data or the algorithms that are applied to it. This could be used within attack scenarios that that follow a covert approach as described, for example, by the APT of Attack Scenario 2 [23, 107, 108] (see Section 4.2.2.2). By inducing false information into a digital twin, operators could be deceived or led to perform tasks that are unintentionally harmful to the manufacturing environment. Another topic worth considering is that of supply chain security. Digital twins are often complex systems that might be composed themselves of various sub-components and -systems. These systems may not all be produced and developed by the company that uses the digital twin. Such third party suppliers can be the victim of an attack or may be malicious by themselves. Offering a compromised sub-component can be an attack vector for the initial breach into a target system [198].

As can be seen, a compromised digital twin can have severe impacts on the security and safety of its connected systems. This in turn means that digital twins themselves should be protected adequately, which includes the model and simulation of the twin as well as the data connection between digital and physical twin [160]. Some potential security measures are discussed in literature and may offer protection to a digital twin within a larger network with connections to OT security [209, 160]. However, as hinted at in Section 1.1, securing legacy systems introduces challenges for security controls that need to be considered carefully [15]. For the context of our thesis, we consider the security of the digital twin when regarded as another network node out of scope [210]. We assume, thus, that the digital twin is executed within an isolated environment that is protected against attacks [28].

4.3 Conceptual System Architecture

This section provides an overview of the conceptional testbed architecture. This architecture is detailed further in the subsequent chapters. We begin by discussing the overall concept of the architecture here in this section. Our proposed conceptual system architecture is shown in Figure 4.6 on Page 128. From top to bottom, the figure is separated into three major areas: the digital twin framework, the manufacturing environment, and the cloud infrastructure. Additionally, on the bottom of Figure 4.6, other manufacturing environments are sketched.

The digital twin framework is the conceptual structure for the digital twin used for information security evaluations in manufacturing. It is a software abstraction layer that combines different software tools, libraries, and interfaces. Furthermore, design principles and guidelines for realization of the digital twin are included. With this, it is possible to provide a standard setup for digital twins to be built and deployed for manufacturing environments. The framework provides a structure for its user and ensures that the digital twin is provided in a thorough manner. As outlined in Section 2.1.2, a digital twin is considered to require at least three components to fulfill its intended purpose [39, 42]. These components are the digital twin itself (i.e., a highly accurate simulation), its physical counterpart (or physical twin), and a connection between the twins. The physical twin is located in the physical environment seen on the upper right in Figure 4.6. Here, the physical twin is for most use cases located in a real-world setup, e.g., a production line [28]. It is connected to a hardware controller, e.g., a PLC or a device provided by the manufacturer of the physical twin. On the opposite side of Figure 4.6, the corresponding digital environment is located. As with the physical environment, the digital environment entails the digital twin, the virtual replica of the physical twin. The digital twin is executed within a high-accuracy simulation environment (see Chapter 6). Also, the digital twin is connected to a virtual controller. This virtual controller serves the main purpose for the digital twin as the hardware controller does for the physical twin [55]. It may be also realized by digital twinning of the hardware controller. Both controllers are connected via a switch. The switch serves two distinct purposes. It allows for data transfer between both controllers and, consequently, between the twins. That is, it realizes the data connection digital-physical and the data connection physical-digital [40] (see also Figure 2.2). Furthermore, it allows for real-time enabled swapping between both twins allowing to conduct experiments

in the digital environment and confirming them within the physical environment. This results in a software that is specific to the use of digital twins for information security evaluations in manufacturing. Critical security evaluation can be conducted in a safe, digital environment before further evaluations take place in the field. This aids in the implementation fo several use cases for digital twins in information security evaluations [28]. The digital twin and the corresponding digital twin framework are embedded within a larger software platform to facilitate information security evaluations.

The twins share a data connection between themselves and are furthermore connected ot the manufacturing environment. The manufacturing environment is a simulation of a plant or facility used for the production of goods. This simulation is built upon a model of the system (center of Figure 4.6). The model of the manufacturing system needs to fulfill several requirements. The model must represent the industrial processes that is controlled by the manufacturing system. This includes the flow of materials and units through the system. Each of these units is subject to quality control within a typical manufacturing setup [55]. Therefore, the model incorporates fitting quality control measures in order to determine the overall quality of the products and goods being produced. For ensuring the availability of the production line, maintenance and downtime planning is required [60]. The model should, therefore, be able to accommodate for planned downtime for maintenance, e.g., for repairs of equipment or changing machine tools. Also, unplanned downtime, e.g., as the result of a cyber attack, needs to be included into the model to enable reasonable information security evaluations. In order to better study the impact of cyber attacks on manufacturing and industrial automation equipment, the model should be able to incorporate safety measures. This allows for a realistic study of information security in manufacturing as safety measures are implemented in almost all devices located on the shopfloor. Business models related to connected manufacturing are reliant on data, preferable large amounts of data [46, 47]. A model of a connected manufacturing environment, thus, must allow to extract, share, and analyze data from various sources, such as sensors, machine logs, and quality control records. This is especially important also when including digital twins within the manufacturing environment [42] as discussed in Section 2.1.2. Consequently, the model must be able to incorporate modern manufacturing technologies such as robotics or machine learning (see also Section 2.1.1). Finally, the model should allow for scalability. This is a basic requirement for models but is also relevant to

the models used for connected manufacturing. Within a modern production environment, changes to the production process, such as the addition of new products (for example, customer-individual manufacturing [10]), adjustments to the demand for the produced goods, or expansion of the manufacturing facility. From there, the model is received as input by a simulation framework resulting in the actual simulation of the manufacturing system. This simulation framework may share a similar basis as the digital twin framework. In fact, the simulation of the manufacturing environment may very well be an extension to the digital twin provided by its framework. This can be used to simulate various scenarios for the twin and to visualize them in real-time. Benefits from this are help identifying potential problems and improve decision-making for plant operators [9, 55]. The final component in the manufacturing environment is the edge device. An edge device in manufacturing refers to a specialized computer that is located at or near the edge of an industrial network. Typically they are located on the shopfloor close to the tooling machine [46, 47]. In manufacturing, edge devices can be connected to a variety of sensors, PLCs, and other OT equipment. Particularly modern PLCs, e.g., such implementing current standards like OPC UA can provide large amounts of data to the edge device [211, 212]. It is worth noting, that the edge device in our conceptional system architecture may be a real computer connected to the simulation or a real device. Edge devices can be assigned several tasks including acquisition of (raw) data, initial processing of that data, or performing some preliminary analyses before the data is sent to another device. These devices can be central servers within the IT network of the company or cloud-based platforms for further analysis operated by external providers.

Such a cloud-based platform is the final component in our conceptional system architecture. It is located on the bottom of Figure 4.6 and labeled as cloud infrastructure. The purpose of the cloud infrastructure is to collect and analyze data from the manufacturing environment. The infrastructure runs a virtual server hosted on an external platform [46]. It is optimized to handle the specific workload that results from extracting large amounts of data in real-time form a manufacturing environment. The data transmission to the cloud infrastructure must be protected by state-of-the-art secure network protocols, e.g., TLS, to protect the extracted data. The data is securely transferred to an instance of the virtual server. Here, steps like indexing of the data are performed in order to prepare the data for the actual analysis. The cloud infrastructure should provide a visualization of the data as well

as of the results of the analysis, preferably reachable via a web-based interface. This allows interaction with the platform and control of the data [47]. Both, the user-facing components and the virtual server, can be virtualized using containerization technology. Using containerization technology provides greater flexibility and scalability, as additional resources can be added to the virtual server as needed to handle increasing data volumes. However, data persistence must be ensured to prevent loss of the data in case of server reboots or re-deployment by the use of volumes that store the data collected from the manufacturing environment. This ensures that data is not lost if a container is restarted or re-deployed. By leveraging the provided data, the cloud infrastructure allows for better data analysis especially when combining several additional sources from different manufacturing environments. This is depicted by the additional manufacturing environments connected to the cloud infrastructure shown on the very bottom of Figure 4.6.

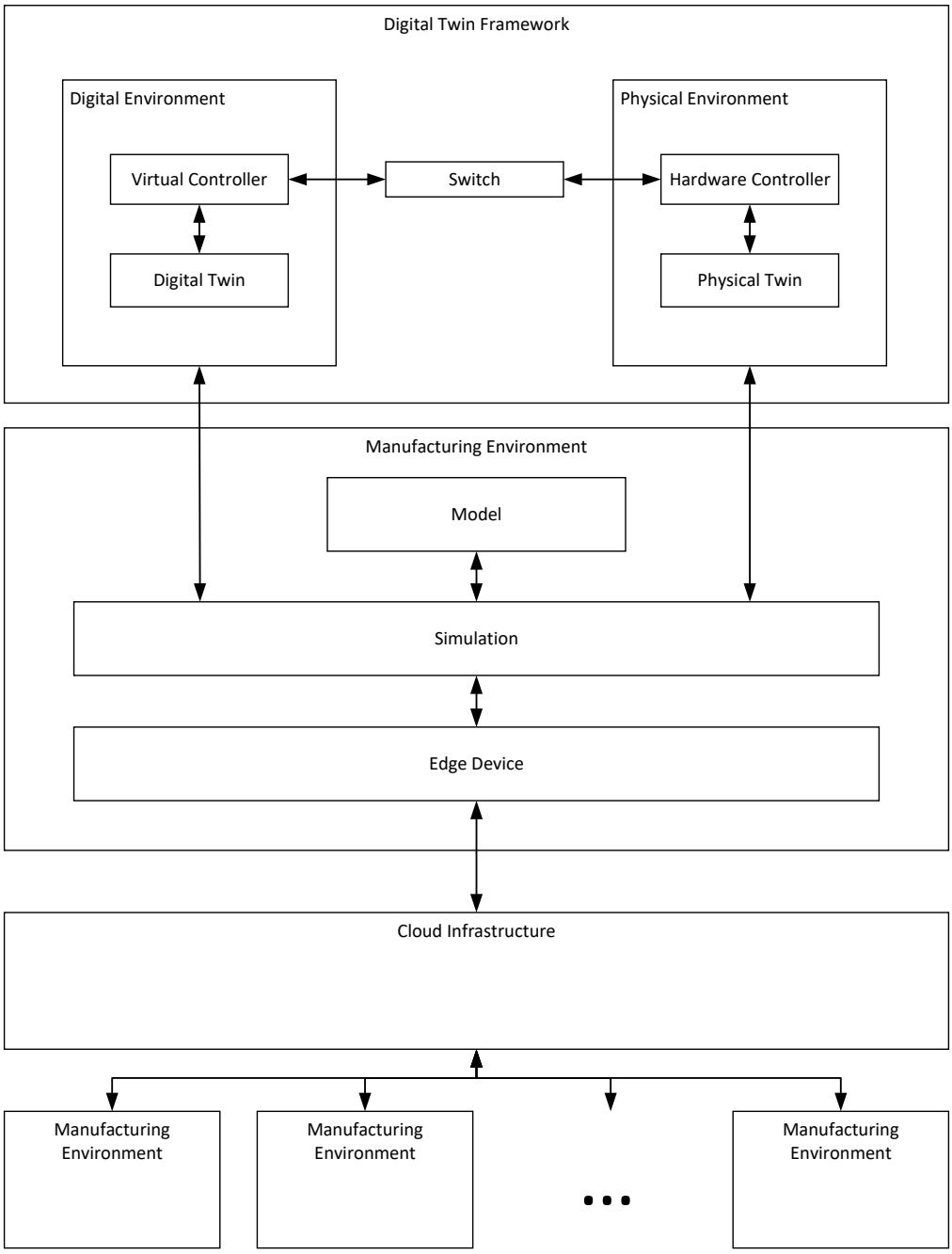


Figure 4.6: Conceptional system architecture with digital twin framework, manufacturing environment, and cloud infrastructure.

4.4 Discussion

In this chapter we discussed the conceptual system architecture with the underlying threat model. That threat model consists of two main parts: our attacker model and corresponding attack paths the attackers might take. The attacker types for studying information security evaluations are defined by us in Section 4.2.1.1. As there is currently no unified attacker model for manufacturing [19], we take attacker models from different sources into account [104, 105, 19] and extend based on the methodology provided by additional attacker models that are reasonable within the domain of manufacturing [193, 194]. This allows us build upon a well-balanced catalog of different attacker models. To summarize, the attacker models are: basic user, insider, cyber-criminal, hacktivist, nation state, terrorist, and competitor. Each of the attacker models is explained by us and the threatened protection goals are given for each attacker. The protection goals in manufacturing are given by us and put into context by a comparative analysis with the domain of classical IT [48]. From there, we continue with an exploration of attack scenarios for connected manufacturing in Section 4.2. We discuss attack vectors typical in manufacturing. The discussed attack vectors emphasize the nature of cyber-physical systems (CPS), i.e., their digital nature and their ability to act within the real world as well [53, 156]. The digital part of CPS is represented by the following attack vectors: reconnaissance attack, man-in-the-middle (MITM) attack, injection attack, replay attack, and denial of service (DoS) attack. The physical part is addressed by additional attack vectors, namely: device manumission, direct damage attack, and stealth attack. All of the attack vectors, for the digital as well as for the physical part of CPS, allow us to gain a better understanding of how attacks are executed within manufacturing environments. We provide a generic attack vector model that allows us to express these attack vectors in a unified manner. This supports us in defining our own attack scenarios. These attack scenarios defined by us are the specific attack use cases that are investigated for the remainder of this thesis [46, 55, 47]. The attack scenarios are named sorting, sawing, and data sharing. Each of them uses a different attacker model, an attack vector specific to the threat posed by the attacker, and a different protection goal threatened by the attacker. The evaluation of the attack vectors is conducted using our conceptional system architecture. The system architecture itself is comprised of three major components. the digital twin framework, the manufacturing environment, and a cloud infrastructure. The digital twin frame-

work serves for implementation of the digital twin. Furthermore, it allows for data exchange with its physical twin and for switching between both twins. The manufacturing environment provides a complementing simulation of industrial processes and equipment for the digital twin framework. Thus, the manufacturing environment serves as the context in which the digital twin is used. Furthermore, is allows connection to a cloud infrastructure. To this infrastructure, different manufacturing environments can be concreted. All of the connected manufacturing environment provide anonymized data to the platform. The platform itself performs information security evaluations that benefit all connected manufacturing environments. The conceptual system architecture is implemented in the following chapters, where also the attack scenarios are investigated further and evaluated.

5 Modeling of Digital Twins in Manufacturing for Security Evaluations

In this chapter, we discuss the modeling of manufacturing environments. For this, we present our modeling approach that is the basis for our work in this thesis. As discussed previously, our modeling approach is based on UML state machine diagrams (see Section 3.1.1). We show how these type of diagrams can be used for modeling digital twins in manufacturing for the purpose of studying information security. For this, we focus on how attacks can be modeled and the consequences of those attacks can be quantified. Furthermore, we discuss the integration of the digital twin in a modeled manufacturing environment.

The presented concepts in this chapter are based on our previous work [9, 60] but extend on it. Modeling of manufacturing environments is discussed by us initially by our previous publication in [9]. In this publication, we focus on modeling of tooling machines based on their functionality rather their physical construction. As a modern tooling machine is offering a variety of different services, it is reasonable to focus on this service view [9, 46]. This means, that services (or functionality) of a tooling machine is modeled as an individual state. The alternative to this service-oriented view on tooling machines is a monolithic view. Here, the machine is seen as a single component offering a variety of functions, i.e., one state with several methods. This monolithic view, does not capture the flexibility required by modern manufacturing paradigms, e.g., customer-individual manufacturing, that is considered as an important part of connected manufacturing [1]. We next extend upon this scheme in another of our publications [60]. In this work we apply more meaning to the individual states especially in regards to information security and the measurement of related parameters.

In this chapter now, we extend upon our previous work on modeling of manufacturing environments [9] and the measurement of parameters [60] by introducing several new concepts. The first extension is the introduction of concurrency. With

this, we are able to express the complex reality of large plants consisting of several production lines. Another addition to our previous work is the integration of the digital twin. This extensions are discussed in Section 5.1. In particular, we note that the usage of UML state machine charts is not conducted in this context previously, either by us or other authors (see also Section 3.1.1).

5.1 Modeling of Manufacturing Sites

Our modeling approach is designed to address the peculiarities of connected manufacturing (cf. also Section 1.1 and Section 2.1). These distinguishing characteristics are, however, manifold [1, 19, 28, 18]. Thus, our focus is on those related to the study of security and digital twins within manufacturing. We begin by giving an overview of how our modeling approach maps to the manufacturing domain. From there, we discuss characteristics related to that domain and give a detailed description of how the elements of a UML state machine diagram are used for modeling. This is complemented by illustrative examples.

UML state machine diagrams are composed of a variety of elements that describe its semantics (see Section 2.2.1). These semantics represent the basic concepts encoded within UML state machine diagrams, e.g., states, transition, or events, as well as more advanced concepts, e.g., orthogonal regions and hierarchically nested states [74]. Table 5.1 shows how the semantics of UML state machine diagrams are related to connected manufacturing. Note that not all semantics described by [74] are listed in the overview given by Table 5.1. The reason for this is that not all semantics are suited to represent a concept of a connected plant.

Table 5.1: Semantics for UML state machine diagrams and their mapping to connected manufacturing.

UML state machine semantic	**Representation in connected manufacturing**
Objects	Parts or products that are processed or manufactured within the plant
States	Operation or service provided by a tooling machine for an object
Transitions	Movement of an object through the plant
Events	Indication that an operation is finished
Initial State	Entry point for the object into the plant or a specific area of the plant
Final State	Exit point of the object from the plant or a specific area of the plant
Orthogonal Region	Parallel manufacturing and processing of objects
Hierarchically nested states	Degree of detail for the manufacturing process

An example on how modeling of manufacturing processes with our modeling approach can be conducted is given by the state machine in Figure 5.1. Here, a piece of wood is processed by an area of the plant where it is cut, drilled, and finished [205]. The first state entered by the object is the *placing* state located in the upper left of Figure 5.1. Here, a change in the orientation of the object is performed so it can be processed correctly by the following states. Once the part is placed accordingly, the transition to the *cutting* state is performed as response to the event *part_placed*. The service provided by the tooling machine by the state *cutting* is the dimensional reduction of the wooden plate, i.e., the object. Once this is achieved by the tooling machine, a sensor located in that machine checks whether a specific action, i.e., *drilling*, is required. A decision is made based on the check performed upon the object exiting the *cutting* state. If the check returns *true*, a drill point is placed by the *drilling* state. In that state, different types of material, e.g., wood or glass, can be drilled. The type of material is checked upon entry of the object in the *drilling* state. Depending on the result of the evaluation, the corresponding drill head is set by the tooling machine. Then, the material is drilled according to the specification of the material [9]. Once this operation is performed or if the previous evaluation of the variable *drill_required* concludes with *false*, the object is transitioned to the *border_banding* state. The operation of border banding is the process of attaching a border to a wooden plate for protection and for aesthetic purposes. Finally, the border is finished and the part leaves this area of the plant for further processing.

Typically, the services provided by the individual states that comprise the manufacturing process shown in Figure 5.1 can be performed by a variety of different tooling machines. That is, because different tooling machines from different vendors are capable of the same or similar services. One characteristic of manufacturing that is relevant for the study of security is the heterogeneous technological landscape of a modern plant. A modern, digitized plant consists of a multitude of different components and systems provided by various vendors [213]. This is true for industrial controls systems (ICSs), machine tools, and for the communication infrastructure used within these modern plants [21]. In addition to this, home-grown solutions can be present. This is frequently the case for MESs acting as a connecting layer between enterprise systems and devices on the shop floor [214, 80]. These factors contribute to the heterogeneous landscape within a modern plant. In order to address this heterogeneous landscape, our approach to modeling based on UML state machine diagrams is generic. UML state machine diagrams are a presentation of FSM and

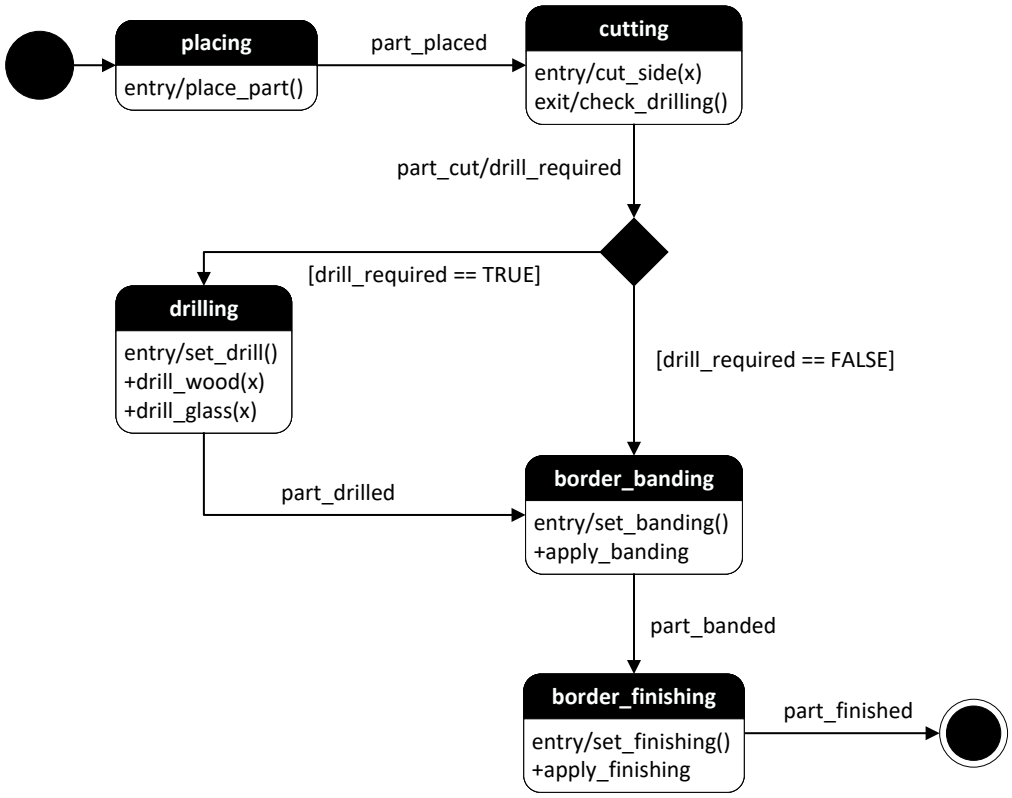


Figure 5.1: Representation of a manufacturing system as UML state machine chart.

are, therefore, well-suited to model different types of DEVS (see Section 3.1.1). A manufacturing process contains multiple interconnected devices and systems that communicate with each other in order to produce a desired outcome, i.e., the manufacturing of products. With UML state machine charts, the individual devices and systems that are part of a manufacturing process can be modeled. This includes the different states the device(s) can be in, the events that occur within the system and trigger transitions to other states, as well as the actions that take place when such transitions occur. The description of the service performed by a tooling machine modeled by a UML state machine diagram allows the replacement of the underlying technology without adopting the model.

For example, in a manufacturing process, the *placing* service of Figure 5.1 can be performed by a robotic arm. This is detailed by Figure 5.2 showing the *placing* state in a detailed description. This description models the placing operation performed

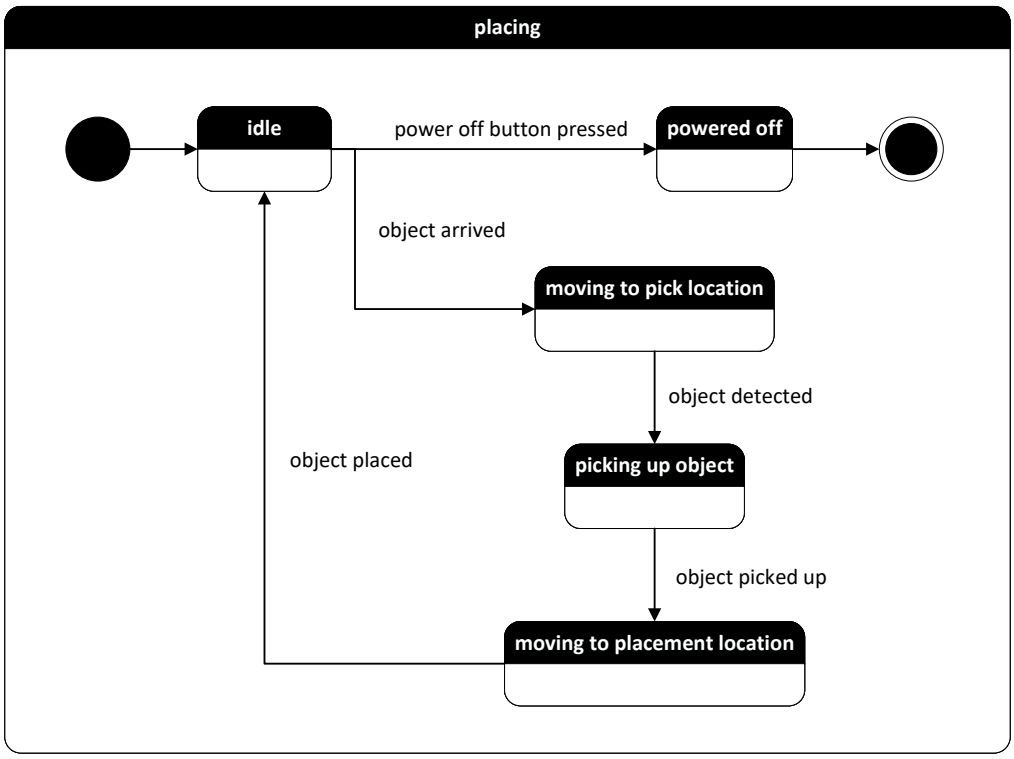


Figure 5.2: Representation of a placing operation as UML state machine chart with hierarchically nested state.

by a robotic arm as hierarchically nested state. The state machine starts in the *idle* state. In it, the robotic arm is waiting for a signal to start the placing operation. The signal can come from a higher-level system, such as the MES, or from a sensor that detects the presence of an object to be placed. When the arm receives the signal to start the placing operation, it transitions to the *moving to pickup location* state. In this state, the arm activates its locomotion system and moves to the location where the object to be placed is located. Once it reaches the pickup location, it transitions to the *picking up object* state. In this state, the arm activates its gripper to grasp the object. After picking up the object, the arm transitions to the *moving to placement location* state. In this state, the arm moves to the location where the object is to be placed. Once it reaches the placement location, it transitions to the *placing object* state. In this state, the arm opens its gripper to release the object. After placing the object, the arm transitions back to the *idle* state, ready to receive a new signal to start the placing operation. Each state in the state machine can have actions

associated with it that occur when the arm enters or exits that state. Also, actions can occur during state activity. For example, when the arm enters the *moving to placement location* state, it might activate its sensors to detect obstacles and avoid collisions, when it enters the *picking up object* state, it might activate its gripper motor to close the gripper around the object. Using UML state machine diagrams, we can model the behavior of the entire manufacturing system, including individual devices and their unique interactions. With this, the behavior of the entire system can be observed by us at once. This makes is possible for us to identify problems that can occur within the manufacturing process and that may be part of an attack or security-related incident.

In addition to representing the technological variety, the variety in manufacturing layouts can also be represented by UML state machine diagrams. This is necessary to encompass technological progress in modern manufacturing environments and to include existing, heterogeneous manufacturing landscapes. Furthermore, manufacturing environments have varying requirements. For example, an assembly line of a furniture manufacturing plant uses a different set of tooling machines as food packaging lines [215]. Also, tooling machines from a variety of vendors are typically employed by plant operators (see Chapter 7). These machines vary in their range of operation and the included tools. Additionally, a theoretically large range and variety of different products can be produced on an assembly line given a fixed set of tooling machines. This results in a high complexity for manufacturing setups as they are typically found within a modern plant. The complexity of these modern manufacturing environments can be addressed in variety of ways when attempting to model them [35]. Models are always an abstraction of the real device they attempt to model and are, thus, not the real device. However, a model can still be useful in drawing conclusions from a simulation. Thus, a common approach is to reduce the complexity of the model itself in an attempt to make the model more easily manageable [54]. This can greatly reduce the complexity of the model itself and the computational resources required for simulation of the model. Another approach for making models of complex devices, such as a modern manufacturing environment, more manageable is compartmentalization. Compartmentalization is a modeling technique that involves breaking down a complex system into components or compartments of that system. These compartments are more manageable by themselves as the entire system as they, naturally, do not include the entire system model. Each compartment represents a distinct aspect of the system and is modeled separately,

which can be achieved also by using different modeling techniques or modeling notations. This isolation of compartments can make it easier to understand and analyze the respective compartment. In practice, compartmentalization can be achieved by two mayor approaches. The first of these approaches is division of a complex system into subsystems. Here, each of the subsystems is modeled separately. Alternatively, a complex system can also be divided into different layers. Then, each of those layers represents a different level of abstraction or view on the system. The division in different layers can be seen, for example, by the model for automation as seen in Figure 2.1.

5.1.1 Modeling of Digital Twins within Production Facilities

The inclusion of digital twins within a model requires, however, specific modeling techniques [59]. As mentioned in Section 2.1.2, a digital twin is a more detailed representation of a real device than a traditional model [28]. Thus, reducing the complexity of the model for the digital twin appears to be not suitable when attempting to include a model of a digital twin within a larger model. In return, that indicates that compartmentalization appears to be a more viable approach instead. The model of the digital twin can be represented as a compartment within the larger model. Suitable modeling techniques can be used to ensure that the digital twin is aptly modeled. Compartmentalization can allow for increased granularity and accuracy in the model, as each compartment can be modeled with a higher level of detail. For digital twins, this means that model for the model can offer a higher fidelity as other compartments of the overall model. Compartmentalization in UML state machines is realized by the notion of hierarchically nested states (see also Section 2.2.1). Hierarchically nested states are used to aid in the effort of modeling complex systems and behaviors within those systems. They are broken down into smaller, more manageable parts or compartments, where a state can contain other states. The contained states are referred to as sub-states, the containing states as super-states. Thus, hierarchically nested states form a hierarchical structure within a UML state machine diagram. Sub-states, further, can contain their own sub-states and, consequently, act as both, sub-state and super-state within the UML state machine model. An example for the usage of hierarchically nested states is given by Figure 5.2. Here, as described above, a placing operation is modeled. Typically, several placing operations can be performed in series within an automated manufacturing process [9]. Several individual placing operation then result in a positioning

operation. The positioning operation itself is part of the overall manufacturing operation. This is depicted by Figure 5.3. Here, the manufacturing operation is modeled with hierarchically nested states with a total of three layers. The manufacturing state is the super-state located on the outer layer. This outer super-state contains two sub-states, i.e., the positioning state and the processing state. The positioning itself contains the aforementioned placing states (denoted as *placing_1* and *placing_2*). Thus, the positioning state acts as sub-state of the manufacturing state as well as super-state for the placing states.

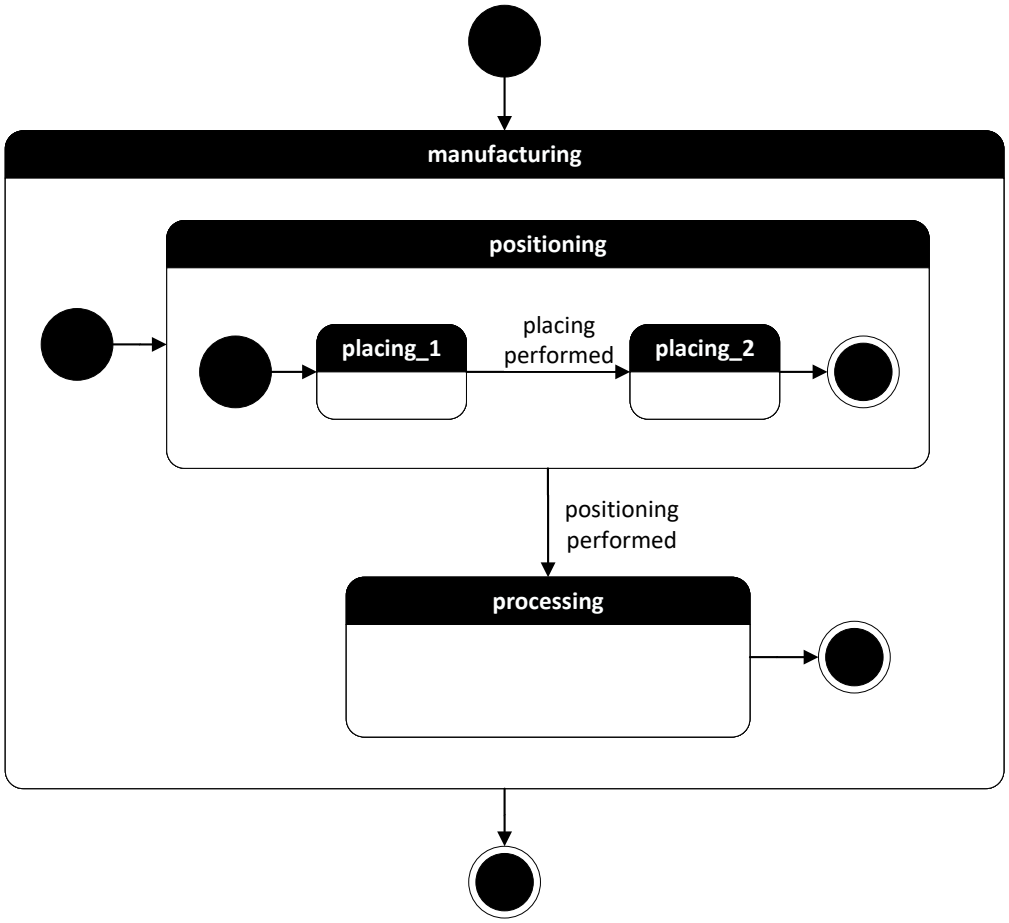


Figure 5.3: UML state machine chart of a simplified manufacturing operation represented with as a three-layered model with hierarchically nested states.

A digital twin can be contained within a larger system model as a sub-state. When attempting to include a digital twin model within a larger model, it is important to ensure that all relevant parts of the model of a digital twin are included. A digital twin is a virtual representation of a real-world system (e.g., a machine tool like a robotic arm) [40, 55]. In particular, a digital twin has increased requirements on the accuracy provided by its virtual representation of the real device [28, 54]. A digital twin can be evaluated on how well it captures and, thus, represents its physical counterpart. If relevant components are left out of the digital twin model, it may not accurately represent the behavior and performance of the physical twin. This can introduce a variety of problems, ranging from incorrect predictions over erroneous analyses to overall inefficient product development. Therefore, including all relevant parts within the model of a digital twin is of paramount interest. In the context of our discussion that means for the model to provide at least a complete functional representation of the twin (the accuracy of the simulation is a topic discussed separately in the subsequent chapters).

For this, the key components or elements that constitute the physical system need to be identified initially. This can include a variety of systems such as sensors, actuators, or PLCs. Identification of the key components of a system can be a challenging task by itself alone [41, 42]. It involves to understand the purpose of the digital twin. This includes the goals and objectives of the digital twin as well as what problem or process it is meant to address [33, 156]. This can support the modeler to determine which components are the most important to the twin and, therefore, to include within the model. This can be followed up by a thorough review of the architecture of the physical components. For this, design documentation can provide relevant information. It may also be the single source of information for the architecture review when the physical twin is not accessible, e.g., in the early stages of prototyping. The interdependence of the individual architecture components is important to understand and consider within the model of the digital twin. This is especially important for UML state machine diagrams as these are strongly focused on providing an expression of the modeled system's behavior (see Section 2.2). Finally, for identification of the key components of a physical system, the environmental conditions and their potential impact they might have on the system need to be included in the analyses. In the context of information security for connected manufacturing, the physical environment and the interactions of with it must be included in order to provide meaningful evaluations [60].

Once the key components are identified and understood, the next step is the creation of the actual model. In our case, that means the creation of a UML state machine diagram. The diagram represents the behavior of the physical system and their interactions with other systems. A state machine diagram for a digital twin must aim to capture all of the relevant states the system can be in as well as the transitions between these states. For example, the model of the placing operation seen in Figure 5.3 can represent a model of a digital twin during this stage of modeling. After modeling the states and the transitions between them, the behavior of the individual components of the modeled system is added. That means including functions and entry as well as exit events to the states. Finally, all further relevant events are added to the UML state machine diagram. All events that can trigger a transition between states within the diagram need to be considered for this. This can include events that are initiated by the physical system or product itself, as well as external events that can affect its behavior, such as changes in environmental conditions or user input. Naturally, validation and testing of the UML state machine is required to ensure that it accurately represents the behavior of the physical system or product. The model should be revised and adjusted based on the results of testing.

Modern tooling machines offer a variety of functions for the manufacturing environment they are located in [9]. Thus, a model for such a tooling machine needs to include corresponding states for all those functions or operations. For instance., the operations *border_banding* and *border_finishing* of the manufacturing environment seen in Figure 5.1 can be part of one physical tooling machine specializing in border processing. Furthermore, the communication interfaces included within the model need to be the same as for the physical twin [46]. Otherwise, the interoperability between the twins is not given [47]. This lack of interoperability can, consequently, lead to varying results when comparing the behavior of the physical twin to that of the digital twin. Furthermore, this would violate the important property of a proper data link between both twins [39, 42]. Considering these two aspects, i.e., inclusion of all relevant states and usage of identical interfaces, the model of a digital twin becomes a sub-state with properly defined communication interfaces. This allows for the digital twin to be integrated easily in existing models of manufacturing environments through compartmentalization via hierarchically nested states. This way, digital twin models can be validated and tested within different production contexts. Furthermore, it is ensured that the digital twin is capable of receiving the same data as the physical twin. This way, a hardware-in-the-loop integration of

the physical twin within the simulation of the manufacturing can be envisioned [55] (see also Chapter 6). Also, the accuracy of the model is of sufficient granularity as key components for the digital twin are identified and included within the model. Also, compartments other than the digital twin compartment can be included with a reduced granularity allowing for more flexible and potentially cost-effective modeling [54].

Once the UML state machine diagram with the model of the digital twin is created and tested for accuracy and fidelity, further modeling aspects can be taken into account. Those include for the most part a model for the attackers and their capabilities. The key consideration to be kept in mind when modeling attackers is the topic of the following section.

5.2 Modeling of Information Security within Connected Manufacturing

In this section we discuss the modeling of attackers and overall information security within the domain of connected manufacturing. Information security evaluations require sufficient modeling of information security in order to yield useful results [26, 34, 33]. Furthermore, the emerging and dynamic nature of connected manufacturing is in need of information security being implemented by design (i.e., security by design) [16]. This can be supported by building proper system models. As modern manufacturing systems are becoming increasingly more heavy on their reliance or need of software, modeling techniques need to reflect upon this. As discussed in Section 3.1.1, UML can provide this functionality. In the context of UML, modeling of information security can be incorporated into state machine charts to create a comprehensive view of the system's security architecture. With state machine charts, the reaction of a system to attacks can be understood. High-level modeling approaches, such as UML state machine charts, make it possible to study the security architecture (or the lack of it) of a manufacturing system. This includes the behavior of the manufacturing system when facing potential threats. Furthermore, UML state machine charts provide a visualization of the modeled system, which makes it easier to observe and study the impact an attack has on the system.

For modeling of information security, we consider the following security-related entities or concepts into the system: attackers with their capabilities, the malware and other tools used by these attackers, security controls present within the system, potential vulnerabilities, and (semi-)automated incident response procedures. Due to the flexibility provided by the modeling language of UML state machines (see also Chapter 2.2.1), the overall modeling of these entities can be conducted in different ways. We discuss different approaches and include suitable methods for each in the following. One approach is to include a state labeled as *secure* within a given UML state machine diagram. The system can enter this secure state indicating that some level of security is reached or maintained by the system. This is an abstract way to model security within UML state machine diagrams. It does not provide any specifics what exactly entails this state of security. Though useful to gather a rough understanding on when a system is operating securely, detailed studies of the security for the modeled system are not possible with this abstract approach. Entities like malware or specific security controls would not be explicitly represented

by this secure state. Therefore, we include a more detailed model of security within our models.

5.2.1 Attackers and Their Capabilities

First, we consider the attacker model. The attacker model is a representation of the behavior and capabilities of potential attackers [104] (see Section 4.1). The capabilities of an attacker can be represented either directly or indirectly within the state machine chart. A direct representation of an attacker means to include several attacker states, e.g., attacker states labeled *scanning* or *exploiting* representing the reconnaissance phase of an attack and the actual execution of the attack [53, 18]. The attacker can transition between these states and interact with other states, e.g., states representing a tooling machine, via dedicated attack events. To use an attacker model in UML state machine charts, potential attacker states need to be identified. The attacker states can be those from the generic attack vector model we discussed in Section 4.2.1.1. Next is to consider how an attacker can transition via these different states of its own attacker model in order to achieve their goals. This is represented by events and actions within the UML state machine model. For example, an attacker may move from the *scanning* state to the *exploiting* state in order to gain illicit access to a vulnerable system. When attacker states, events, and actions are identified, they can be incorporated into a UML state machine chart, e.g., by providing an attacker with its own orthogonal region (see Section 2.2.1). In such a region, for instance, only identified attacker states and other corresponding elements may be included. Modeling the attacker and their states as orthogonal regions considers the independent nature of both, attackers and their targets, and the concurrent nature of cyber attacks. Overall, the state machine for the attacker needs to include the different states that an attacker can be in, the events that can trigger a state transition, and the actions that an attacker can perform in each state.

The actions an attacker can take are also related to the representation of the attacker. As mentioned above, direct and indirect representation of an attacker is possible within a UML state machine diagram. So far, we discussed the direct representation of an attacker. The indirect representation reflects the actions the attacker can perform on their target. Indirectly modeling an attacker with UML state machine charts involves modeling the system's response to potential attacks rather than explicitly modeling the attacker's behavior. For this, the state machine needs to be designed to capture potential attack scenarios and the system's

response to these scenarios (for potential scenarios, see Section 4.2.2). This can involve modeling potential attack vectors and the system's response to each vector, e.g., a network-based attack or an APT. A state machine diagram can be used to indirectly model an attacker by focusing on the system's behavior as it encounters and reacts to potential malicious activities. This approach highlights potential vulnerabilities within the system and demonstrates the impact of an attack. To begin creating an indirect attacker model, it is crucial to identify potential attack vectors by analyzing the system for possible security weaknesses or vulnerabilities. For tooling machines or other components used in industrial automation, a substantial amount of unpatched software components are present and can be assumed by the model [15, 156]. Examples for such vulnerabilities can include SQL injection, cross-site scripting, or buffer overflow attacks. For each identified attack vector that is relevant given a specific attack scenario, the system's behavior in response to the attack need to be modeled. This can be done as transitions between states that are triggered by specific attack events, e.g., *SQL Injection Attempt* or *Buffer Overflow Attack*. To separate the behavior of the system under attack from its normal, intended operation, guard conditions with corresponding actions can be added to each transition. Guard conditions are Boolean expressions that must evaluate to true for a transition to occur. For example, a guard condition might involve checking whether input validation has failed before a corresponding action like logging the failed attempt and blocking the attacker's IP address occurs. Actions can also represent the execution of an attack. These *encoded actions* are distinguished from regular actions that are labeled with the prefix '+' by including the prefix '-' [1] . For example, the encoded action *-run_exploit* can indicate that a particular vulnerability is present on the system and can be exploited by executing this encoded action. Encoded actions represent attacker behavior in an implicit way as the attacks are encoded within the non-attacker states of the model. The individual attack steps or techniques are included within the system without requiring the presence of dedicated attacker states. In each regular system state, the chart can, thus, include encoded actions that represent the attacker's activities and the impact these actions have on the system.

So far, we discussed modeling of the attacker and its capabilities. We showed how explicit and implicit modeling can be used within that context. Our focus was on

[1] Note that prefixes '+' and '-' are used in software engineering to distinguish between public and private methods respectively. This represents the idea of intended and illicit operations within our models.

the attacker by describing states and other chart elements related to them. The manufacturing system itself is influenced by these attacks. Attackers may install malware on the system and, thus, altering the system and its intended service or purpose. The adjustments in service that result from a cyber attack can be severe. In fact, the results can be so severe that the physical system itself is altered, for example, when destruction of components occurs or the system is rendered unavailable otherwise. In order to reflect such alterations to the modeled manufacturing system, it can be reasonable to include dedicated states that indicate insecure or unsafe operation of the system. This can be as abstract as a state labeled *insecure* or *vulnerable* indicating that the system has transitioned to an insecure or vulnerable state (analogously to the *secure* state discussing on Page 145). Such an approach can be reasonable when modeling attack scenarios that result in the physical destruction of equipment or render manufacturing equipment unavailable, e.g., until an unscheduled maintenance is performed. For example, consider a manufacturing system consisting of several, concurrent production lines. If that is system is modeled via orthogonal regions where each region represents a line, a failure in one of the lines can result that the state machine in that particular region enters the *insecure* state. This way, the overall functionality of the system is still given while a part of the system is rendered unavailable.

Another possible approach for modeling attacker capabilities for manufacturing are the aforementioned encoded actions. These actions are included in states that represent vulnerable manufacturing systems. Typically, at least one vulnerability is present within the system for a successful attempt on attacking the system. The execution of the encoded action represents system usage that is not intended originally by the system's designer. For example, consider a manufacturing system that is programmed and setup to produce electronic devices. The system consists of several industrial tooling machines, including a circuit board assembly machine, a soldering machine, and a quality control station (typically a tooling machine with various sensors such as video cameras or lasers). One potential vulnerability in this system is the injection of malicious code into the circuit board assembly machine. The circuit assembly machine has a vulnerability that allows for the execution of this attack vector, e.g., a lack of code signing or sender authentication. If an attacker were to inject malicious code into the machine, they could potentially compromise the integrity of the electronic devices being produced, e.g., by reprogramming the machine's operation or by altering of parameters. This can be represented by the

encoded action -*InjectMaliciousCode()* within the *assemble circuit board* state. Note that an encoded action can also trigger an event and, thus, cause a state transition. This can be used to bring the system in a state that is not intended by its control logic (encoded within the MES for manufacturing). The model can, furthermore, include states that represent ongoing attacks. When considering the example of malicious code injection from before, instead of executing an encoded action without triggering a transition, the system can transition in an additional state labeled *injecting malicious code*. Within this state, the injection of the malicious code can be modeled and examined.

5.2.2 Malware and Other Attack Tooling

Second, we discuss modeling of the tooling used by the attackers. The main point of focus for us is on malware, which is the main tooling used by an adversary for conducting an attack [25, 13]. Different types of programs can be considered as malware, for example, viruses, Trojans, worms, or ransomware, among others. For definitions and differentiation between these programs, the interested reader is referred to the appropriate literature [122]. Adversaries use these malware tools for different purposes, for example, to exploit vulnerabilities in systems, to infiltrate and move within networks, to access disclosed information, or in order to cause damage and disruption to the targeted system or organization. Modeling malware can be as complex as modeling any other part of a manufacturing environment and cyber security. For instance, APTs are characterized by complex behavior and are operated for a longer period of time in the field than most other malware [23, 198]. Similar to our discussion about modeling attackers, malware can in principle be included as direct model or as a stateless state machine chart. A directly modeled malware is a stateful representation of the malware entity. Stateful in this context refers to the explicit representation of an entity by states within a state machine chart. For example, consider a Trojan malware. This type of malware can start with a state named *connecting to command and control server* (assuming the malware is already being installed and has become active). Once the connection to a remote server is established, the Trojan issues the *connected* event and transitions into the *listening to commands* state in which the Trojan waits for instructions. Once the command for downloading a payload is received via the *download* event, the Trojan transitions into the *downloading payload* state and so on. A model of a stateful malware like the exemplary Trojan can also be included in a larger model by orthogonal regions as

discussed above. A stateless model of a malware is a representation of the malware's behavior without explicitly modeling its states. In a stateless UML state machine chart, the transitions between the malware's states are represented as events that trigger (encoded) actions, rather than explicitly defining the states themselves. It is, thus, a more event-driven view on the system, where the focus is more on the impact a malware can have on the overall system rather than a detailed observation of the malware's internal workings. The events can be represented by triggers that cause the malware to execute whereas the actions represent the specific behavior of the malware. For example, for a Trojan, a trigger events can be that the user opens a malicious email attachment, downloads a malicious file, or visits a compromised website. The actions taken by the Trojan can be installation of the Trojan, monitoring of user activities, collection of sensitive data like passwords, and so on. By using a stateless UML state machine chart, the focus is on the flow of such events and actions rather than the explicit modeling of states for the malware. This approach can be useful for modeling malware that is highly dynamic and reactive, as it allows for a more flexible and event-driven representation of its behavior. Furthermore, the processes of the manufacturing environment and the impact the malware has on it are more clear to observe.

5.2.3 Information Security Controls

Third, we discuss modeling of security controls capable of mitigating attacks on manufacturing environments. There are several different high-level approaches to modeling security controls that can be used depending on the use cases being studied [28]. The first of these high-level approaches is threat modeling. Threat modeling is a structured approach to identifying and assessing potential threats to the protection goals of a system. A threat is realized when an adversary executes an attack. Threat modeling involves, among others, creating a diagram of the system architecture, identifying potential threats, and then determining which security controls are necessary to mitigate the identified threats. UML state machine charts can be used to represent the behavior of security controls in response to potential threats. Here, the state machine can include states for different attack scenarios. For example, an anomaly detection system can react differently to distinct patterns in the monitored network traffic in an industrial control system [49, 216, 37]. Risk analysis involves identifying potential risks to a system [31, 29]. The final score for a potential risk can be computed, for example, as the product of the likelihood and

the potential impact of the risk. This approach can be used to identify and prioritize which security controls are necessary based on the level of risk associated with each potential threat and overall target security level for the system. Risk analysis can be supplemented by state charts to model the behavior of security controls in response to different risk scenarios in a similar fashion as with threat modeling. For example, an intrusion detection system can react differently in response to different levels of risk. A state machine can include states for low-risk scenarios, medium-risk scenarios, and high-risk scenarios, as well as transitions between states representing the response of the intrusion detection system to each level of risk. Both of the approaches mentioned above, that is, threat modeling and risk analysis, may appear to be similar to each other. However, each of the two approaches serves distinct purposes when examined closer. On the one hand, threat modeling focuses on identifying and mitigating specific threats and vulnerabilities. On the other hand, risk analysis has its focus on assessing overall risks to a system and typically includes some way of prioritizing risks so that the most critical risks can be addressed first. Both approaches support each other and a combination of threat modeling and risk analysis in a unified methodology can be reasonable.

Security requirements engineering involves gathering and analyzing security requirements for a system. Following these activity, corresponding security controls meeting those requirements are designed. Requirements engineering is a common task in engineering is aided by modeling and simulation with increasing tendency. UML state machine charts can be used to represent the security requirements for a system. With state machine, the behavior of a security control that meets a certain set of security requirements can be modeled. For example, a state machine can model the behavior of an access control system that accepts certain ways of authentication like passwords. The state machine can include states for different levels of access. Transitions between these states represent the behavior of the access control system in response to each requested level of access. Security testing, on the other hand, involves testing the effectiveness of security controls of a system. This approach can be used to identify potential vulnerabilities and weaknesses in security controls. In addition, security testing can determine if the tested security controls are effective at mitigating potential threats. UML state machine charts can be used to model the behavior of security controls in response to different types of attacks. This way, test cases for testing those controls can be specified. For example, a state machine can model the behavior of a password authentication system

in response to different types of password attacks such as dictionary attacks. The state chart could include states for different types of attacks, the transitions between states that represent the response of the authentication system to each attack. Both of the approaches discussed above, that is, security requirements engineering and security testing, are distinct approaches. Security requirements engineering, on the one hand, focuses mostly on the the design and implementation of security controls that adhere to a set of security requirements. Security testing, on the other hand, focuses on evaluating the effectiveness of existing security controls and on identifying potential vulnerabilities within them.

All of the high-level approaches described above require a proper model for information security controls. The examples provided for the approaches already give some indications on how modeling of security controls can be realized with UML state machine diagrams. We conclude this section with a discussion on how to model information security controls for manufacturing with UML state machine charts. In order to model information security controls with UML state machine charts, the necessary information security controls need to be identified. This can be controls already present within the manufacturing environment or it can be components that are identified to be necessary to include into the system, e.g., as the result of a risk analysis. While those security controls can be described on a high level, e.g., use access control, encryption, or authentication, it is more reasonable to name specific security controls. For example, a security control for authentication in manufacturing environments can be role-based access control, for encryption Transport Layer Security (TLS) can be used, and authentication can be implemented via password authentication. This allows for a more precise modeling of the system, which in turn enables informative impact analysis [60] or better evaluation of the security control [54, 55]. Once the security controls are identified and specified, a similar modeling approach is used as described above. The different states that the security controls can be in are defined. This is followed by the definition of all events that can trigger a transition between those states. For example, a user logging in, a file being encrypted, or a network connection being established can trigger transitions between states. The step is definition of the actions that are performed when a transition occurs. For example, at user login, access control checks can be performed in order to determine if the user has the required permissions in order to access the system.

5.2.4 Vulnerabilities Present within the Modeled System

In this section we cover the modeling of potential vulnerabilities present within the system. Modeling of vulnerabilities is introduced in Section 5.2.1, we provide a detailed discussing in this section. In the context of IT systems, which includes modern manufacturing systems, a vulnerability refers to any weakness or flaw in the system that can be exploited by an attacker to compromise the protection goals of that system. That weakness or flaw is inherent to the system. Therefore, a vulnerability can be considered as a dormant aspect of a system as long as it remains undiscovered. The longer a vulnerability remains dormant, the greater the likelihood that it is discovered and exploited by an attacker. The vulnerability then becomes active and must be considered from now on as a potential threat to the system. Thus, modeling of vulnerabilities means including a vulnerability model within the system model. This vulnerability model can be realized by different model approaches. As described in Section 5.2.1, encoded actions can be used to represent unintended functionality within a manufacturing system. Such functionality that is not intended by its designer can result in illicit and malicious usage of the manufacturing system. For example, the successful Stuxnet or Industroyer malware campaigns that caused large amounts of damage to industrial equipment was enables due to vulnerabilities present within the attacked systems [23, 13, 103] (see also Section 2.1.3 for additional examples). Encoded actions as an element belong to a specific state, i.e., a state from the manufacturing environment. In addition to this, depending on the type of vulnerability and the purpose of the model, it can be beneficial to include vulnerable states into the model, for example, a state labeled *vulnerable*. This and the addition of the vulnerability model to the overall system model is discussed by us for the remainder of this section.

Initially, potential vulnerabilities need to be identified prior to inclusion in another model. The vulnerabilities included into a model of a manufacturing environment should be reasonable considering the architecture of the system. That means, using vulnerabilities matching to the type of equipment used within (the model) of the manufacturing system. For example, some PLCs are known for certain types of vulnerabilities to be present in them given a certain patch level. Such vulnerabilities for manufacturing equipment can be identified by distinct methods. One such method is research within vulnerability databases such as the Common Vulnerability Enumeration database (CVE) or product Computer Emergency Response Teams (CERTs). However, for manufacturing systems, public disclosure of vulnerabilities

is lower when compared to public disclosure of vulnerabilities of IT systems [217]. One reason for this is that most companies that produce automation equipment are concerned about the confidentiality of their proprietary information. Publicly disclosing vulnerabilities can potentially reveal information about their products, processes, or technologies that are feared to be exploited by competitors. Also, fear of image loss could be part of the reason as well. Other methods for identification of vulnerabilities can involve following a structured process, such as code analysis, penetration testing, or risk assessment [29].

After the vulnerabilities are identified and understood, the next step is to describe the behavior of the vulnerability by encoded actions. This involves mapping the actions in the system to the corresponding states representing the vulnerable automation equipment. If several actions need to be encoded within the system, for example, when a vulnerability enables complex attack vectors, it can be necessary to create the encoded action sequence. This sequence of actions then describes the behavior of the system. Such a sequence of actions needs to be structured so that it can represent the behavior of the system experiencing the potential attack vector in an accurate manner. At this point, it may be necessary to create and additional states to the system model that are specifically meant for modeling vulnerabilities. The introduction of additional states and transitions here represent potential attack scenarios. The new states are included into the sequence of actions in a way to still reflect the behavior of the system. For the new states modeling the vulnerability, transitions and corresponding events also need to defined and included within the model.

5.2.5 Incident Response Procedures

In this section we discuss the modeling of incident response procedures within UML state machine diagrams. There are several incident response procedures known, such as redundancy, real-time monitoring, disaster recovery, or continuous backup and restoration. Furthermore, fault tolerance techniques can be used to enhance the security of manufacturing systems by reducing the impact of faults or errors that could be exploited by attackers to compromise the system. They can, for instance, allow for continued operation of a manufacturing system even in the presence of an active adversary. However, it is still reasonable to immediately start removing the malware form the affected systems even though this requires a costly shutdown [111]. The cost of the consequences from a cyber attack, e.g., destruction of equipment or

leakage of intellectual property, can be higher than the cost of the shutdown. Even so, if the shutdown follows pre-defined procedures and occurs, thus, in a planned manner. Such pre-defined procedures for a planned shutdown are present within manufacturing and industrial automation for a long period of time [48, 60]. Here, the area of fault handling offers incident response procedures that are well-established. Originally designed for handling safety-related issues, they can also be used within the context of information security in connected manufacturing. We discuss the four major techniques used in this domain and how they relate to information security as well as to modeling with UML state machines diagrams. These techniques are diagnosis, isolation, reconfiguration, and reinitialization. This enables us, to use these techniques from the domain of manufacturing in our models. It is worth noting that, although the techniques from fault tolerance can mitigate some threats, they are by themselves no substitute for a comprehensive information security strategy in manufacturing [203, 218, 25].

Diagnosis, isolation, reconfiguration, and reinitialization are the major components of fault handling in manufacturing systems, and can also act as partial and supplementing incident response procedures within the context of information security [48]. Diagnosis involves identifying the cause and scope of the (ongoing) incident. This can be accomplished through automated systems such as intrusion detection systems (IDSs) up to via manual labor [109, 111]. Diagnosis can be represented as a transition from a manufacturing state to a *diagnosing* state in response to an attack event, e.g., the detection of a security incident by an IDS. Also, a concurrent, permanent diagnosis can be modeled via an orthogonal region. Once the cause of the incident is identified and its scope is known, appropriate incident response procedures can be initiated. One such procedure is isolation. It involves separating the affected system components from the unaffected components of the system. The intention behind this is to prevent the spread of the malware before it can affect other systems. This way, further damage or unauthorized access can be prevented. For example, if a security incident is detected in one production line or a single manufacturing cell, that line or cell can be isolated from the rest of the manufacturing environment. Again, similar to diagnosis, isolation can be represented by an *isolated* state where the (sub-)system transitions to in response to an event or condition, such as the detection of a security breach. When transitioning to the *isolated* state, actions and events such as *disabling machine tool*, *isolating network segments*, or *shut down system* can be triggered for instance. Once the incident is resolved,

the system can transition back to its regular states with corresponding events, e.g., *rebooting* or *enable machine tool*, can be triggered. The next procedure is reconfiguration and involves changing the manufacturing system's configuration or layout in response to the security incident. This can include modifying system settings, replacing hardware components, or even changing the layout of the manufacturing environment. For example, reconfiguration can involve updating access control policies, blocking malicious IP addresses, or deploying security patches to affected systems. As described above, reconfiguration can be represented as a transition to a *reconfiguring* state in response to an security-related event. The reconfigured state can include actions such as *+updateAccessControlPolicies*, *+blockMaliciousIPAddresses*, or *+deploySecurityPatches*. Once the reconfiguration is complete, the system can transition back to its regular states again. The final fault tolerance procedure is reinitialization. Reinitialization involves restoring the system to a known good state, such as by restoring from a backup or resetting the system to its initial configuration. This can be necessary after a serious security incident that cannot be resolved through other means. Again, the transition to a *reinitializing* state with corresponding actions can be a way to model this type of fault tolerance procedure.

5.2.6 Summary

In the previous section, we discussed several modeling techniques that allow for integration of information security in manufacturing environments. To summarize, attacker models can be modeled either directly or indirectly. Directly modeled attackers have explicit attacker elements such as states, events, transitions included within the system. On the other hand, indirectly modeled attackers are represented within the system by encoded actions assigned to corresponding states of the model for the manufacturing environment. Indirect attacker models can, however, be supported by states representing a vulnerable system configuration. Vulnerabilities present within an industrial automation system can be represented by encoded actions. With this modeling techniques, dormant properties of a system that are not intended by the system's designers can be represented. These encoded actions can be called upon by malware that is introduced to the system by the attackers. Malware, and other tooling used by the adversaries, can be modeled with dedicated states or stateless. Stateless modeling here reflects more on the event-driven nature of software programs, such as malware. From there, we discussed how information security controls can be included within the model of a manufacturing system using

UML state machine charts. We discussed how high-level approaches to information security, i.e., threat modeling, risk analysis, security requirement engineering, and security testing, can be supported by state machine charts. This shows how information security use cases can be aided by modeling the behavior of the system. The support for information security use cases, especially in the context of digital twins in manufacturing [28], is further elaborated on in the subsequent chapters. Also, we briefly showed how specific information security can be expressed within the model. From there, we conclude our discussion with incident response procedures as additional information security controls. Incident response procedures are implement in some manufacturing systems and are related to fault tolerance [48]. In particular, we discussed diagnosis, isolation, reconfiguration, and reinitialization as the major techniques in fault tolerance and how to include in an UML state machine chart of a modern manufacturing environment. For each of the concepts of information security discussed above, we discussed several distinct approaches on how to represent them within a model of a manufacturing environment using UML state machine diagrams as modeling language. Modeling of information security in manufacturing is strongly depended on the use case studied and the research interest in these use cases. Therefore, different approaches to expression of a certain concept can be used in different attack scenarios and for different use cases. In the subsequent sections, we apply the modeling techniques discussed in this section to practical examples of manufacturing environments. We discuss the decisions involved in providing system models that support conducting information security evaluations within manufacturing.

5.3 Reference Scenario

In this section, we apply our modeling approach based on UML state machine charts to develop a reference scenario. The reference scenario is based on the attack scenarios discussed on Section 4.2. The intended purpose of the reference scenario is to provide a model for a manufacturing environment where different attack scenarios can be mapped upon. For this, we use a generic model of a manufacturing environment that included enough nodes for conducting the analysis of network-based attack vectors but still remains clear enough in order to stay flexible and usable [18]. The reference scenario itself is based on our previous work but uses the modeling technique developed by us so far in this chapter [60]. This way, we can establish a baseline where the performance of the model can be evaluated against. With a well-defined reference scenario, we can measure the performance of the model under varying or similar assumptions and, thus, evaluate if the model is credible. In other words, our reference scenario serves a benchmark for comparison. With it, the performance of the model can be evaluated objectively. For this, we first discuss the background of our reference scenario before we provide a refined model of the reference scenario. From there, we integrate the attack scenarios discussed on Section 4.2 and discuss our reasoning on modeling them. We include modeling of the manufacturing environment as described in Section 5.1 and modeling of information security as described in Section 5.2 for model development. Then, we use this model for a comparative analysis with another model in order to verify its validity and discuss evaluation of the model further.

Our reference scenario is based on production lines in manufacturing [1, 2, 3]. A production line, also referred to as assembly line, is a form of organizing a manufacturing environment. In a production line, the production process is organized as a linear sequence of individual workstations that are traversed one after another. All workstations within a production line are connected to each other. Each of these workstation is tasked to perform a specific operation within the production of goods. The input material to the production process is processed along the production line from one workstation to the next. The input materials, such as raw materials or unfinished products, undergo sequential operations along the production line until it leaves the production line, either as finished product or as intermediate good for the next step in production. The goods processed by the production line follow, thus, follow a sequential flow through the production line, that is, a predefined sequence of

operations, where each workstation performs a specific task in a fixed order. Each workstation contains either machine tools for automated operation but there can also be workstations involving some manual labor performed to the goods. The workstations of a production line typically include quality control measures for determining and ensuring the product's quality [55]. Production lines are widely used in modern manufacturing across various industries. However, as the manufacturing landscape is evolving, manufacturing processes and organization also evolve with new technologies, e.g., in automation or flexible production systems. For example, production lines are intended for mass production of a specific, standardized product and may not handle frequent changes in product configuration or design well like it is the case with customer-individual production [9, 1]. Consider, as a more elaborate example, furniture manufacturing, in particular the production and assembly of kitchens [219, 220]. Kitchens are a piece of furniture that is often demanded by customers with several customization options. Kitchen manufacturing can have a relevant role for a customer-individual business model within connected manufacturing [1, 69]. They are, however, also standardized kitchens, so when a large amount of kitchens is ordered for installment in large housing blocks. Therefore, operators of a manufacturing environment dedicated to the production of kitchens use multiple production lines within their manufacturing layout. A subset of these lines is used to manufacture standardized kitchen models while the remaining lines are used for custom designs. This a realistic setup for a manufacturing environment producing kitchens as frequent reconfiguration of tooling machines is costly and can additionally lead to undesirable side effects, e.g., machine chatter [46, 47].

Our reference scenario adopts the concept of production lines as described above. We consider a manufacturing system where several production lines are operated concurrently. Thus, hierarchically nested states and concurrent regions are a reasonable approach to modeling such an environment (see Section 2.2.1). Figure 5.4 shows such a production environment. The entire manufacturing environment is subsumed within the state *Manufacture products*, which is the top-level state within the hierarchy of the nested states. On the second level of hierarchy the production lines for sequential processing of the products and goods are located. We already include our definitions determining the size of the manufacturing environment so the visualization of the model in Figure 5.4 shows the final iteration of the model. As can be seen, the production setup consist of three production lines (PL) labeled *PL1-PL3*. These states are executed concurrently within the model as indicated by

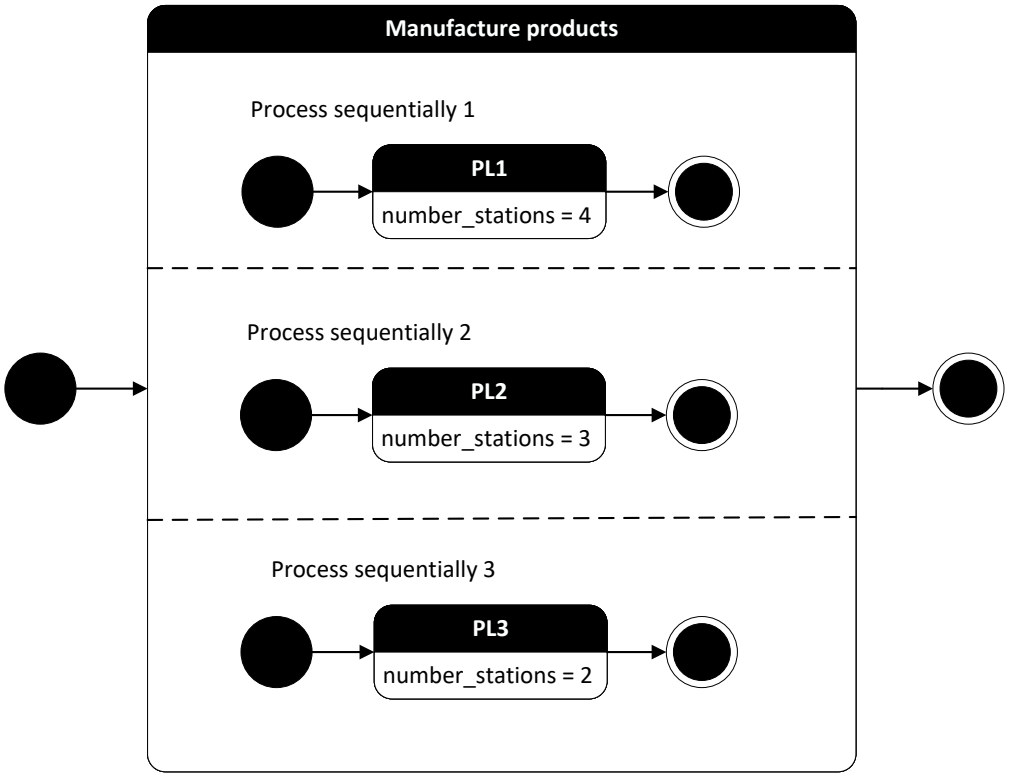


Figure 5.4: UML state machine model of a production system that consists of three concurrently executed production lines.

the two dashed horizontal lines indicating orthogonal regions. The inputs to the model arrive at the starting state to the left of Figure 5.4. These can inputs include raw materials, components, or partially assembled systems. The inputs arrive either from the outside of the manufacturing environment, e.g., as delivery from suppliers, or from other internal sources, e.g., a previously concluded production step. Feeding the inputs into the model indicated the starting point for the manufacturing process. From there, the inputs are distributed by a production planing system [9]. The production planing system is responsible for a variety of tasks including determining the production schedule and establishing the sequence of operations. The production planing system is not included as an explicit state within the model but as programmed logic that is executed as an entry event when the transition from the main input to the *Manufacture products* state is triggered. This state represents the main production facilities that contain the tooling machines and other equipment

necessary to carry out the production process. The layout and organization of the production facility is designed for efficiency and minimizing bottlenecks. The production process is the steps and operations involved in transforming the inputs into products or goods. Note that a finished product may not necessarily be the result of a production process as also partially finished products or systems can be produced and serve as an input to s subsequent production step or process outside the scope of our model. The output of the manufacturing system is then distributed further.

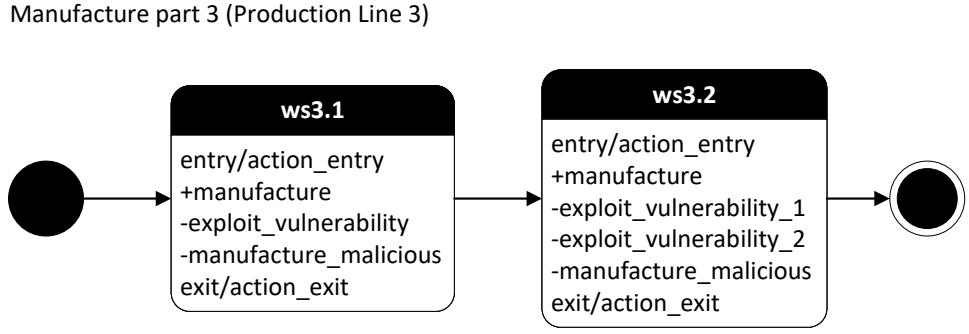


Figure 5.5: UML state machine model of a production line that consists of two sequential production steps.

Each of the three production lines has a specific number of workstation assigned to it that are connected in a sequential fashion. The number of assigned workstations is indicated by the variable *number_stations* belonging to each of the *PL* states. So, *PL1* consists of a total of four workstations, *PL2* of three workstations, and *PL3* of two workstations. The number of workstations is assigned in order to stay compatible with our baseline scenario [60]. Figure 5.5 shows the model for the third production line, *PL3*, with its two workstations. The models for the other two production lines, *PL1* and *PL2*, are modeled analogously with varying number of workstations. All our models of a production line start, again, with the input of raw materials or components required for the manufacturing process. The input received is based on the decisions derived by the production planing system. In some production processes, these inputs may be stored in an internal warehouse or at a local storage space close to the workstations. This is omitted in our model so we can focus on the production process itself and to reduce complexity of the model. The inputs are distributed from the entry point to the first workstation labeled *ws3.1*. This transition to workstation as well es the transition between workstations

in general are conducted via a conveyor belt. A conveyor belt is a fundamental component of a production line. It serves as the main means of transporting inputs or partially processed products workstation. The conveyor belts are represented by the transitions between states in our model and are triggered by events (not shown in Figure 5.5). The events typically indicate that the production step has finished and the next step in production can be started at the next workstation.

The models for the workstations are visualized by the states *ws3.1* and *ws3.2* in Figure 5.5. At a workstation specific tasks are performed. Each workstation is assigned a particular operation or set of operations summarized by the action *+manufacture* in our model. This action can include a variety of operations such as assembly, cutting, banding, or quality control. Furthermore, actions can be executed on entry or exit of the workstation state (*entry/action_entry* and *entry/action_exit* respectively), e.g., proper placing of components before processing or subsequent transportation. These actions are carried, for the most, by automated machine tools such as laser cutting tools or robotic arms. Also, quality control can be performed at the workstations, e.g., camera-assisted inspection of the parts produced. The models for a workstation include further security relevant elements. In the states representing the workstations, *ws3.1* and *ws3.2*, these are vulnerabilities present within the software of the machine tools used but also the possible misuse of the machine tool. The vulnerabilities are included as a separate action labeled *-exploit_vulnerability*. The '-' prefix indicated that this action is hidden and typically not executed by the machine (see Section 5.2.4). In addition to this, different tooling machines can have a different number of vulnerabilities. For example, *ws3.2* is a different tooling machine then the one used by *ws3.1* and has two vulnerabilities instead of one vulnerability present in its system (*-exploit_vulnerability_1* and *-exploit_vulnerability_2*). The misuse of the machine tool is included in the model by the action *-manufacture_malicious*. This action indicated the usage of the machine tool in a way that is not intended by the production planing system. For example, when parameters are used that can lead to a deterioration of expected product quality [55]. This is, for example, what the attackers in the Stuxnet incident aspired to in order to sabotage the industrial process of their target facility [23] (see Section 2.1.3). Thus, the action *-manufacture_malicious* can be reasonably used to model the behavior of an APT within the model. Note that malware such as APTs usually also require the presence of vulnerabilities at some point within their attack vector (see Section 4.2.1). Again, the '-' prefix of the action *-manufacture_malicious* indicated

that this action is not used under normal working conditions of the production line but rather in the presence of an active adversary.

We make some simplifying assumptions in the model of the production line described. Some we already discussed like the omission of local storage facilities and the assumption that the finished goods are transported somewhere outside the production facility. This extends to the entire area of inventory management within the production facility. Also, we assume that the production lines follow a strictly sequential order of processing, that is, no feedback loops like seen in Figure 2.3 are considered within the layout of the production facility. Also, we include not dedicated quality control step at the end of the assembly process. Rather, we assume that quality control procedures are strategically placed within the respective production lines. Also, we consider the modeled production facility fully automated meaning that the process is expected to be executed without minimal manual intervention as possible.

5.4 Discussion

Our presented modeling approach helps us to derive a basis for representation of manufacturing environments for the remainder of this thesis. The evaluation of the models, which are used to model the attack scenarios described by Section 4.2, is given in the respective sections. Specifically, Attack Scenario 1 (see Section 4.2.2.1) and Attack Scenario 2 (see Section 4.2.2.2) make use of the modeling approach developed in this chapter. Their evaluation is given by Section 6.1.3 within the context of information security testbeds with digital twins. For the remainder of this chapter, we discuss the advantages and potentials for future work of our modeling approach within the context of connected manufacturing, digital twins, and information security for both domains.

First of all, our modeling approach is not only compatible with the paradigm of connected manufacturing (see Section 2.1.1) but also further facilities its continued extension. This is because the integration with connected manufacturing is supported by UML state machine charts. UML state machine charts can model the complex interactions of automated systems that include, for example, CPS [49]. Furthermore, as discussed by us in Section 5.2, UML state machine charts can express the additional interaction that occur in the presence of an adversary. The methods discussed in this section are based on our previous work [9, 60], which were developed and tested by us in the research project IUNO (see Section 1.4). We implemented a proof of concept, which was later implemented by a IUNO project partner within an extensive plant as part of their production planning system [10, 11]. This system solved a problem related to manipulated design files that could potentially damage tooling machines and other production equipment. In particular, see our previous work of [9] for a detailed discussion of this attack and the associated damage scenarios. The modeling technique evaluated at this time was not based on UML state machine charts. UML state machine charts, however, offer a significant improvement of our previous work, both in terms of including more potential use cases in manufacturing and by providing support to digital twins.

In terms of digital twinning, UML state machine charts can support the integration of accurate digital twins into the simulation of a manufacturing environment. This is due to the technique of compartmentalization, which we used for including digital twins into our larger models (see Section 5.1.1). Compartmentalization can help with increasing the fidelity in digital twin models. A complex model, which can be

the manufacturing environment but also the digital twin itself [42], can be broken down into smaller models or compartments. Each compartment can be modeled with greater accuracy, e.g., for using different techniques or by implementing support for a specific simulation. This approach to compartmentalization can improve the overall fidelity of the digital twin model, which makes the digital twin model more effective for analyzing the behavior and performance of the physical system. Also, the data collection for the digital twin can be improved with UML state machine charts. This is an important aspect, as digital twins rely on data, either live or historical, which is discussed by us in more detail in Section 2.1.2 and Section 6.1.2.1. For the context which this chapter is concerned with, data collection can also benefit from compartmentalization of the system model, as data collection can be detailed for each relevant part of the system. Furthermore, by using compartmentalization, the verification and validation of your model is improved. Comparing individual parts of the digital twins with the physical twin helps in verifying that the model was built right and that the right model is built. By testing each compartment separately, it can be easier to identify and correct any discrepancies within the model. This also does again improve the fidelity of the digital twin as its representation can be improved upon successively. We discuss this in the context of unit and system testing as laid out within our testing methodology in Section 6.1.3 further. Overall, compartmentalization can be a useful technique for increasing the fidelity of a digital twin model. By breaking down a complex system into smaller compartments, the digital twin model's fidelity can be improved upon and testing of the model with its implementation can also benefit from it.

5.4.1 Future Work

With our new modeling approach based on UML state machine charts described in this chapter, several improvements to our previous work [9, 60] are added. However, there are still some more potentials for future improvements to our approach left, which is what we discuss in this section.

In our evaluations, we used processes and manufacturing setup that are related to subtractive manufacturing (see Section 3.1.1). While this is a reasonable decision for reducing complexity and still does provide support for a large variety of manufacturing use cases, it does exclude additive manufacturing processes. However, it appears possible to us that models for additive manufacturing can be included within UML state machine charts. A combined approach [221, 222] can be realized by providing

functions for adding of material and by adopting the model with corresponding elements. The effort for integration of these models cannot be reasonable estimated by us, however, as integration of continuous processes within UML state machine charts seems possible [74], the overall effort should be located within a manageable period of time.

The knowledge on the subtractive manufacturing processes encoded within our models is currently contained within them in an explicit fashion. That means that the knowledge is included internally in each new model and not explicitly formalized. This means, that for any change in processes or general knowledge on the production, the models need to be updated and a new version of the entire model needs to be rolled out. This is impractical, especially when dealing with a large number of compartmentalized system submodels for complex systems. Externalization of this implicitly encoded knowledge could allow for better maintainability. That includes the knowledge of the manufacturing process but also about security. Knowledge systems that can provide an externalized knowledge of information security exist within the contest of ontologies [223]. Such ontologies can be used and extended for manufacturing [191], which does allow for an integration within UML state machines. This can be achieved rather straightforward once the ontology is implemented, as UML state machine charts do provide support for a large number of programming languages [74] (see Section 2.2.1).

As outlined above, UML state machine charts are well-suited for the the integration of digital twins into a larger model of a manufacturing environment. However, UML state machine diagrams are not a reasonable choice for modeling digital twins themselves. They may be supportive within the development of the model, e.g., as an abstract representation of the behavior for the physical twin. But such models also need to cover other aspects of the twin, which cannot be represented in a dynamic modeling language such as UML state machine charts. Static modeling is also required for covering all aspects of the digital twin, e.g., UML class or component diagrams as outlined by Section 2.2. Nevertheless, UML state machine charts offer a valuable contribution to the modeling of digital twins in the light of information for manufacturing, which is of benefit to us for implementation. Implementation of a simulation environment for digital twins in manufacturing in order to conduct information security evaluation is the topic of the next chapter.

6 Information Security Testbed for Digital Twins in Manufacturing

In this chapter, we discuss the realization of our information security testbed for digital twins in manufacturing. As outlined in Section 3.1, our starting point for the construction of a digital twin that can be used for information security evaluation in manufacturing is that of related OT security testbeds. As the construction of a real digital twin appears not to be achievable at the moment [28, 42, 18] (cf. our discussion on this topic in Section 1.1 and in Section 6.1.4), we start by building a credible information security testbed [34, 27]. From there, we show in this chapter how to realize a sufficient digital twin model and simulation that can be used for information security evaluations in manufacturing.

We start by discussing the design of information security testbeds for their intended usage within the domain of manufacturing in Section 6.1. For the construction of a credible information security testbed, several aspects need to be considered by us [33, 32, 26]. We detail our assumptions and design goals for the testbed within Section 6.1, discuss its construction with the integration of digital twins into it, and give a detailed methodology for evaluation of our testbed and its digital twin.

From there, we continue in Section 6.2 with the preparation for implementation of our testbed. That is, we develop a thorough baseline architecture for our testbed. This baseline architecture details the conceptual architecture discussed by us in Section 4.3 and gives the required insights into how our testbed is supposed to function. With the baseline architecture we can demonstrate how our design approach is realized by us. Also, the attack scenarios and their specification are given by us within Section 6.2.

In Section 6.3 we discuss how the testbed is implemented. This implementation of the testbed is in accordance to our baseline architecture and provides a sound technological basis for us. We explain what technologies are used by us and how these technologies are chosen based on their possibilities to be used within the near future. Technologies are bound to change but we strive to provide a technical sound

but also modern foundation for our testbed as this is especially important as we consider a key future technologies in connected manufacturing in our work: the digital twin. The construction of the digital twin is discussed as well by Section 6.3 in addition to the some initial evaluations that need to be performed before the actual experimentation can start.

The experiments conducted by us are the topic of Section 6.4. Our experiments are conducted within a laboratory setup with real-life industrial machinery. We detail how this setup is arranged and how our testbed is integrated into it. This serves for understanding the results of our experiments that are also part of this section. In particular, we discuss the realization of Attack Scenario 1 and Attack Scenario 2 (see Section 4.2.2.1 and Section 4.2.2.2 respectively) and the results we generate by using our testbed.

This culminates in the discussion of the overall process of designing, specifying, implementing, and experimenting with our testbed in Section 6.5. In that section, we show our contributions and what they imply for our research goals. We sketch concepts and ideas for future work and subsequent updates for our testbed. Also, we show the limitations of our work, which may support other researchers in using and extending our testbed. This is a partial look ahead on the overall discussion of our work, which is the topic of Section 8.

6.1 Testbed Design for Information Security in Manufacturing

In this section we discuss our design approach for realization of a digital twin testbed for information security evaluation in manufacturing. Our reasoning for design decision are given and discussed. This continues from the examination on the state-of-the-art given in Section 3.1 and details the conceptional system architecture presented in Section 4.3. In order to create a digital twin testbed for information security evaluations in manufacturing, we first discuss testbed design and realization. From there, we discuss important aspects to consider when including digital twins into a manufacturing security testbed. The design and construction of these testbeds is discussed in literature [32, 24, 156, 27]. We follow a general design approach summarized by [33] and adopted where appropriate to include digital twinning technology. As a result, we start by discussing the intended usage of the testbed, followed by the scope of the testbed. The scope of the testbed includes the testbed's major design principles as well as the aspired coverage of the testbed. From there, the evaluation methodology is discussed next. The evaluation methodology describes the criteria and methods for the testbed's evaluation.

6.1.1 Intended Usage and Purpose for the Testbed

For the design of reasonable testbeds for conducting information security evaluations in manufacturing, the intended usage of the testbed needs to be defined before beginning with the conceptualization of the testbed. Understanding the intended usage of the testbed is essential for designing a testbed that meets the requirements for achieving its purpose. A testbed is usually considered to be a tool to achieve a certain purpose. In our case, that is, to conduct scientific studies on information security within manufacturing while employing digital twinning technology. Other purposes most likely have a set of requirements that differ from our purpose even if only in details. By understanding the particular needs of the testbed, the testbed can be designed to provide the appropriate features and capabilities. In most cases, a testbed serves several purposes and aims to integrate these purposes into its design and realization. According to [33] and the examined body of literature, the most common purposes in testbed design are the analysis of attacks and the analysis or testing of security controls. Both are closely related as typically a corresponding defense mechanism is evaluated in response to a certain attack vector. This is fol-

lowed typically by a binary distinction if the security control was either able to stop the attack and mitigate its effects or if the security control was insufficient for mitigating the attack. The analysis of attacks can, however, also be performed to study the impact of the attack, i.e., impact analysis. With corresponding metrics, the attack impact can be estimated and quantified [60, 55]. Impact analysis can also be studied in the presence of active security controls. This way, the effectiveness of the security controls under test can be observed, which is especially useful for controls that cannot be evaluated in a Boolean fashion. For example, a testbed for anomaly detection in manufacturing environments can be evaluated based on the number of false positives detected in the presence of ongoing attacks [216]. Another testbed purpose is the analysis of vulnerabilities. For the most part, vulnerability analysis is concerned with the study of a particular system or sub-system, e.g., a PLC or a robotic arm [199]. Testbeds with such a scope can be well-suited for security testing of single devices, e.g., for penetration testing [28]. Another often stated purpose for testbed is education and training of information security in manufacturing [32, 33]. This can include manufacturing personal working in security-critical areas of the plant, information security professionals being educated for the specific challenges in manufacturing, or students at higher education institutions like universities [95]. Other, less common testbed purposes can include threat analysis for strategic decision making, general performance analysis, or the creation of policies and standards.

6.1.2 Scope Definition for Information Security Testbeds

By defining the usage objectives of the testbed, it is possible to establish the scope of the testbed. This includes identifying the testbed's design principles and it's coverage. We start by discussing the design principles in the order provided by [33], which identified often used design principles for the construction of testbeds. We do, however, omit some of the design principles stated by other authors. We omit such design principles that we do not consider relevant to our study of digital twins. For example, reduction of complexity, as a design principle, potentially may hinder the study of digital twins as they require large amounts of data and are conceived as complex systems. Also, we omit those pairs of design principles that appear to be to closely related to each other as to provide a clear differentiation between them. For instance, modularity is a property that enables adaptability; therefore, modularity is not discussed separately but rather within the context of adaptability.

6.1.2.1 Design Principles for the Testbed

The first design principle we elaborate on is fidelity. We already discussed fidelity in the chapters above (for instance, see Section 2.1.2 or Section 3.1) by referring to the accuracy of the digital twin and other simulation models. Fidelity, thus, refers to the degree to which a testbed accurately represents the real-world system or environment it is supposed to represent [28]. A testbed that offers high fidelity is a testbed that can reproduce key characteristics and behavior of the system under test in a realistic and accurate manner [27]. Fidelity is also related to the credibility of the results obtained from testing. A testbed with low fidelity may not accurately reproduce the characteristics or behavior of the real-world system. While this may not necessarily lead to erroneous results, the resolution of the results can be insufficient to draw conclusions for some applications. The quality and quantity of data used by the simulation can impact the fidelity of a testbed. A fact that is especially true within the context of digital twins as discussed further below [42].

Another design principle discussed by us is testbed flexibility or testbed adaptability. In the context of testbeds, flexibility or adaptability refers to the degree to which a testbed can be modified, reconfigured, or otherwise extended [224]. This way, support for different types of experiments or scenarios that can be executed with the testbed is ensured. A flexible or adaptable testbed can be customized with less effort to meet different requirements, e.g., requirements introduced by different attack scenarios, without requiring significant changes to the testbed's architecture and its underlying implementation. This way, a wide range of different scenarios and use cases can be studied. Also, a flexible and adaptable testbed is in principle more future-proof as adjustments over time can be done more easily as new technologies or requirements emerge. Factors influencing flexibility and adaptability of a testbed include how easily new components and modules can be added or removed, the choice of APIs and interfaces for integration with external systems or tools, and the range of different technologies such as protocols supported.

The next design principle is scalability. Scalability refers to the ability of a testbed to manage an increasing number of components, data flows, and events within its simulation while at the same time maintaining performance and function. A scalable testbed, thus, is a testbed that can handle the growth in simulation elements without requiring significant changes to the underlying architecture [54]. While the design principle of flexibility or adaptability is focused on the customization properties of

the testbed , scalability is rather focused on enabling the testbed to be changed in regards to usage patterns over time.

Another design principle that is discussed within the context of information security testbeds for manufacturing is repeatability. Repeatability refers to the ability to reproduce the same results in different instances of the testbed. In other words, a repeatable testbed must, given the same setup parameters and conditions, be able to generate the same results among different periods of time or place. Repeatability is important in testbeds because it allows others to verify the results and to ensure that the experiments can be replicated by other researchers in different environments [34]. This helps to ensure that the results of an experiment are not coincidental, e.g., via a variation in environmental conditions. For repeatability to be given for a testbed, documentation of the testing procedure as well as a comprehensive list of tools used with their setup environment is required.

The next design principle discussed is the safe execution of the tests or experiments performed within this testbed. Safe execution of tests means the ability of a testbed to conduct experiments and simulations in a controlled manner without posing a risk to the testbed itself, to other connected systems like networks, and to the persons using the testbed or that are in close proximity to the testbed. Safety in general is a critical consideration when designing and using any system including testbeds. Experiments conducted in a testbed can potentially have unintended consequences or cause disruptions to real-world systems or networks. This is especially true when conducting security experiments. For example, penetration testing of components can have unforeseen consequences such as the activation and rapid movement of a robotic arm in an unexpected manner [199]. Appropriate safeguards and mitigation strategies need to be in place in case of such unforeseen issues or also in the case of externally occurring emergencies without direct relation to experiments conducted with the testbed such as the outbreak of a fire.

Another design principle we discuss is measurement accuracy. Measurement accuracy in the context of testbeds means to the degree to which the testbed is able to accurately measure and quantify the behavior of the system or manufacturing environment being tested with the testbed [60]. The usefulness and the degree of trust that can be placed in the experimental results depends on the accuracy of the obtained measurements. A low margin of error and a high resolution of the measurements are preferred properties for testbeds [27]. Measurement accuracy can be influenced by a number of factors. These include but are not limited to calibra-

tion of sensors used for obtaining the measurements, the level of interference in the testbed's environment, or the accuracy of data collection and processing methods. Furthermore, it is important to validate the accuracy of the measurement results through repeated simulation runs or by comparison with ground truth measurements if such ground truth measurements available.

The next design principle discussed is interoperability. In the context of testbeds, interoperability refers to the capability of integrating different components or systems within a testbed. The integrated entities are supposed to function together effectively, regardless of their underlying technology, architecture, or implementation. Examples for such different components can be an emulation of a PLC, a simulation of a physical process, or the integration of hardware components into the testbed [55]. Interoperability needs to be considered in a testbed especially when different tools, platforms, or systems are used in a combined fashion. It can be necessary to include these different components, for example, when studying consequences of complex attack vectors [60]. Interoperability of a testbed can be achieved, for instance, by the use of standardized protocols and formats for data exchange and communication, the availability of common APIs between different components, or compliance to industry or other standards.

6.1.2.2 Coverage Provided by the Testbed

In this section, we discuss the coverage of the testbed as a part of general testbed design. Coverage refers to the degree of which different parts of an industrial automation system are represented within the testbed. Furthermore, coverage also entails with what technologies these parts of an industrial automation system are implemented within the testbed. For this, we briefly recur on the architecture of industrial automation systems before we provide an overview on the different simulation approaches used for implementation of testbeds. Then, we conclude by discussing coverage for testbeds.

As discussed in Chapter 2, industrial automation systems are visualized by a layered model (cf. also Figure 2.1). Each of the layers represents different degrees of automation that can be achieved in a production process by the automation system. From top to bottom, the layers develop from management and business functionality to the actual physical process realized within the plant [13, 76]. Most published testbeds leave the topmost layer, i.e., the Enterprise Layer responsible for business planing and logistics, out of scope for their testbed's realization [33]. Therefore,

testbeds for information security in manufacturing include some sort of Supervisory Control and Data Acquisition (SCADA) system or another type control center as the highest management component. A control center in manufacturing testbeds serves as the central point responsible for managing and coordinating the different components and systems included in the testbed. The control center is responsible for supervising the operation of the testbed while monitoring the testbed's different components. In addition to its operational role, the control center in a manufacturing testbed can also be responsible for administration and configuration of the testbed [30]. The control center is followed by the field devices. Field device here is a general term and can include any number of different sensors, actors, and other devices such as PLCs or Human Machine Interfaces (HMIs). It entails, thus, all devices or nodes that are included within the testbed. In manufacturing testbeds field devices are tasked to provide data on the performance of the equipment and processes simulated. This data can include information on physical measurements, e.g., temperature, pressure, or flow rate, as well as other parameters of the equipment and processes being tested [47]. Also, information on the performance of the equipment and the quality of the products being produced [164, 196]. All layers of industrial automation require a communication infrastructure. The communication infrastructure connects the control center and all field devices among each other. It is largely comprised of the communication protocols and the network architecture [33]. Communication protocols define the rules and procedures for exchanging data and information. They are also a target for attacks on the manufacturing systems [103]. In addition to information security aspects, communication protocols also facilitate interoperability between different components and systems within the testbed environment. The communication protocols included in the testbed need to be considered carefully in order to select those protocols that are appropriate for the specific requirements of the testbed environment [225]. This can involve selecting such protocols that are used within real-world facilities and also implementing additional security measures for legacy protocols [21]. Lastly, testbed's also need to include a representation of the physical process being executed by the automation system. The physical processes can include a large variety of different processes such as chemical processing, oil refining, or power generation (for continuous manufacturing processes) as well as the production of goods that are individual, distinct units (in discrete manufacturing processes) [56, 57]. The simulation of the physical process in manufacturing testbeds provides a realistic and controlled environment for test-

ing and evaluating information security. Without including the physical process in manufacturing testbeds it is difficult to obtain meaningful data on the performance of the equipment.

Once the architecture of the industrial system represented within the testbed is known, simulation approaches for the testbed can be decided upon [224, 54, 68]. We categorize the simulation approaches on their perceived level of fidelity they provide. This type of categorization is reasonable for our discussions on digital twin as fidelity of the digital twin is an important aspect in digital twin realization [28] (see also Section 6.1.4). Our categorization of simulation approaches for digital twins is given by Figure 6.1. It gives an abstract, high-level overview of different simulation approaches that can be considered for the design of our testbed. We group the different simulation approaches on a one-dimensional axis ranging from virtual simulation approaches in the left to physical simulation approaches to the right of Figure 6.1. Virtual simulation approaches are those that rely on software for implementing the testbed. A purely virtual simulation approach is one that is only comprised of software components combined together (obviously, devices required for execution of the simulation such as a PC are not part of that definition). In contrast, a purely physical simulation approach is one where the testbed is completely realized with actual hardware components - just like it was the case with the early twins constructed for NASA's Apollo space program or the so-called Iron Bird [39] (cf. also Section 2.1.2). In between both poles, purely virtual and purely physical simulation approaches, the hybrid simulation approaches are located. These include any number of different simulation techniques such as emulation and virtualization as well as hardware-in-the-loop (HIL) [226, 227, 55]. Both, emulation and virtualization, simulation techniques used to create virtual environments that replicate the behavior of other systems or devices. Both approaches, emulation and virtualization, create virtual environments within their methodology. Emulation, on the one hand, creates a virtual environment on one system that imitates or mimics the functions of another different system. Emulation is often used to execute software on a system that would otherwise be not executable on that system as it was developed for another target platform. For example, in the context of manufacturing emulation is often used to integrate legacy software and systems that were developed on or for a hardware platform that was taken out of service at a later time. Virtualization, on the other hand, also creates a virtual environment but in the context of virtualization that virtual environment behaves like a separate computer system. Virtualization now

is used to create several virtual machines that are executed on one single physical device. This way, several operating systems can be executed concurrently. HIL is an approach to testbed realization that connects a physical system to a simulation environment. Real manufacturing components, i.e., actual hardware that executes the software included by the vendor prior to shipping, are integrated with a virtual environment. That environment generates simulated inputs to the hardware, the output of the hardware under test is then fed back to the virtual environment. This way, the hardware under test can be analyzed and tested in a variety of different scenarios without the need to change a physical setup. Purely virtual and hybrid simulation approaches constitute the vast majority of published testbeds [28, 33].

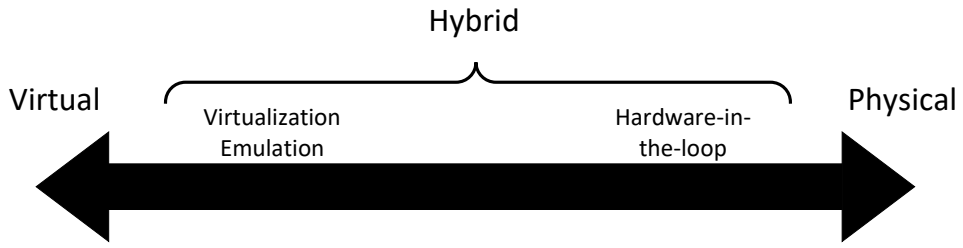


Figure 6.1: High-level overview and categorization of simulation approaches for testbed realization.

So far we discussed the common architecture for industrial equipment and the simulation approaches used for manufacturing testbeds. A testbed for manufacturing includes different equipment, communication protocols, and information security scenarios that are comprised within the testbed's library [225]. From that library, the different experiments conducted with the testbed are compiled [30, 55]. The coverage of a testbed is the type and amount of industrial equipment included within the library of the testbed as well as the simulation approach used for each of the library's items [33]. For example, a testbed's library can contain a HMI that is represented by the commercial-off-the-shelf (COTS) software and a legacy PLC system that is realized via software emulation among others. The choice which coverage to apply can be depended on the intended purpose of the testbed. For example, a testbed that is conceived for enabling penetration testing of industrial equipment can choose to include the system under test (SUT) via HIL while other components used for feeding input signals to the SUT are realized as software simulations. Furthermore, the design principles can introduce their own requirements that can reflect

in the composition of the testbed's library. For example, consider a testbed that is designed with the principle of cost-effectiveness (not discussed in Section 6.1.2.1) in mind [33]. Then, a reasonable approach for testbed implementation can be to use software simulations for as many library items as possible as virtual simulation approaches tend to have a lower cost as physical simulation approaches [54]. Finally, the information security scenarios studied within the testbed can introduce their own requirements on the testbed's coverage. For example, a testbed that includes a scenario with an APT and associated security controls requires a security control that can process data with a minimum level of fidelity [37]. Overall, the choice of coverage for a testbed is influenced by a range of factors as highlighted above. For the designers of a testbed, these factors need to be considered when including architectural items into the testbed's library and selecting a simulation approach.

6.1.3 Evaluation of Information Security Testbeds

Understanding the intended usage of the testbed and defining its scope is essential for establishing the evaluation methodology for the testbed. The evaluation of testbeds is the final step of testbed design. Evaluation is a relevant process that intends to ensure that the evaluated testbed is suitable for its intended purpose, that it produces accurate and reliable results, and that it can be trusted by the testbed's stakeholders [33]. An evaluation process involves collecting data and analyzing it with various metrics or other performance indicators [52]. In order to ensure that the testbed is reliable, accurate, and suitable for its intended purposes, a rigorous and systematic approach is suitable. This process, or rather processes, are known as Verification, Validation, and Accreditation (VV&A). VV&A provides a comprehensive methodology for evaluating the testbed and is performed in that order, i.e., first verification, then validation, and, finally, accreditation. The VV&A process, thus, involves initial verification of the testbed that it operates correctly and meets its design requirements. This is followed by validation of the testbed that it accurately represents real-world scenarios. Lastly, accreditation of the testbed is the last part in that process.

Verification of a testbed is the process of ensuring that the behavior of the testbed is that which is expected from it. Thus, the testbed is checked against its design requirements. This involves to ensure that the testbed is performing the functions that it is supposed or expected to perform. Also, verification involves ensuring whether or not a testbed is not performing any functions that it is not supposed to

perform. During testbed verification, the requirements for the testbed are compared to the behavior of the testbed. Verification can be conducted with a combination of manual and automated testing. A finished verification process aids the overall performance of the testbed, as design flaws of the testbed can be identified and corrected before the testbed is being used. By ensuring that the testbed operates correctly and meets its design requirements, verification helps to build confidence in the testbed's performance and results. In short, verification tries to answer the question, whether the testbed was realized right [9].

Validation of testbeds refers to the process of ensuring that the testbed accurately represents the real-world systems and scenarios it is intended to represent. This can involve to examine the testbed in realistic working conditions and comparing its performance to real-world data. During validation, the testbed can be tested against real-world data to ensure that the system accurately represents the intended real-world scenario. As data is not always available within manufacturing [49] (see Section 1.2), a description of the expected outcome of an attack scenario can be used to check whether the testbed produces the expected results [60]. The performance of the testbed is compared to its expected performance in order to ensure that it produces accurate results. Validation can be performed through a combination of simulation and testing. It can involve the use of real-world data if such data is available. In short, validation tries to answer the question, whether the right testbed was realized.

The final step in VV&A for testbeds is accreditation. It refers to the process of formally recognizing that a testbed is suitable for its intended purposes. For the most part, this involves ensuring that the testbed meets relevant standards and guidelines. For example, this can include such standards concerned with information security in manufacturing such as IEC 62443 [228]. Accreditation can include a formal assessment process by a third party that includes assessment of the testbed's design with external audits or reviews. During accreditation, the testbed is evaluated against a given set of standards or guidelines. It is established whether or not the testbed adheres to the requirements or specifications contained within those standards or guidelines. This is done in order to receive some sort of formal recognition that states that the testbed does adhere to the standards or guidelines in question. That recognition is issued by a third party and states that the testbed was evaluated accordingly. A formal proof of accreditation can aid in establishing

trust with stakeholders as a it shows that a certain level of conformity can be realized by the testbed.

6.1.4 Digital Twins in Manufacturing Security Testbeds

In this section we discuss the integration of digital twins in testbeds for manufacturing security. For this, we initially give a definition of fidelity and the digital twin in manufacturing. From there, we continue by reflecting the testbed design criteria outlined in Sections 6.1.1- 6.1.3 with the requirements of a digital twin and discuss how an information security testbed for manufacturing employing digital twinning technology can be designed.

In our previous discussions on digital twins (for instance, cf. Section 2.1.2 and Section 3.1), we did not provide a strict definition of the term digital twins per se. We rather referred to the digital twin as a virtual replica or counterpart of a real-world device [42, 39]. A precise, universal definition of a digital twin is currently not available [40]. Literature points out that there are numerous definition of digital twins but a definition that is agreed upon by the research community has not arrived yet [56]. One reason for this that digital twins are a comparatively new technology. Though the concept of twinning was originally employed during NASA's Apollo program, academic discussions on digital twins powered by simulations did not occur until 2002/3 [39, 42]. Furthermore, the scope and application of digital twins can vary widely depending on the context in which they are used [28]. This context can be related to the digital twin's area of application and to the intended use cases of the twin. For example, digital twins used within automotive domain, for maritime applications, or within the context of critical infrastructure all share different requirements [229, 230, 152]. This is further complicated when examining digital twins for specific use cases. Digital twins can be used for a variety of use cases, such as performance optimization, simulation of maintenance scenarios, or, within our case, information security in connected manufacturing. Depending on the intended purpose and the use cases of a digital twin, it can be defined and realized quite differently. As a result, digital twins are still evolving in different paths, which explains why there is no unified definition available.

In addition to this, digital twins can have different levels of fidelity. So far, we discussed fidelity in the context of accuracy or resolution of the data fed into the digital twin (see, for instance, Section 6.1.2.1). More precisely, fidelity in the context of digital twins refers to the degree to which a digital twin accurately represents

the behavior and characteristics of its physical counterpart [42]. This includes the accuracy of the geometry and physics of the system being simulated, as well as the precision and completeness of the data used to create and update the digital twin. Digital twins can be realized with varying degrees of fidelity. A digital twin with high fidelity is one that closely resembles the behavior of its physical counterpart. On the other, a digital twin with low fidelity can show significant differences in its representation of the physical system, e.g., as a result of simplifying assumptions. For example, a low-fidelity digital twin can be a simple 2D model of its twin, whereas a high-fidelity digital twin can be highly detailed 3D simulations that incorporates real-time data and has a thorough physics model embedded. The level of fidelity for a digital twin depends on its intended use and the underlying design principles [28, 33]. For example, consider cost-effectiveness as an important design principle for the construction of a digital twin testbed. Creating a high-fidelity digital twin can be expensive, as it requires the use of sophisticated sensors, data processing systems, and other advanced technologies. Therefore, building a low-fidelity digital twin most likely helps in reducing costs for the testbed realization [54, 227]. Also, the realization of digital twins is also largely influenced by the available technology. As discussed in Section 1.1, current technology in manufacturing is not sufficient to built a high-fidelity digital twin [42, 18, 28].

Fidelity is an important aspect of digital twins, as it determines the degree to which a digital twin accurately represents the behavior and characteristics of its physical counterpart. Also, a unified definition is not available with one of the reasons for this being the multitude of application areas and use cases digital twins are used for. Therefore, we adopt working definition for a digital twin used in the manufacturing domain for information security evaluations. The working definition is based on previous work published within our field of study [28] (see also Section 3.1) and is intended to derive a reasonable definition for our usage within this thesis that still covers the basic concepts of digital twinning technology. Our working definition closely follows [28], where the authors compiled a definition from related work. Thus, our definition: A digital twin for information security evaluations in manufacturing is a virtual representation of a physical manufacturing system. The digital twin here is designed to accompany its physical twin throughout its entire product lifecycle. The digital twins, furthermore, can consume real-time and historical data depending on the scenario studied. Additionally, the digital twin possesses a level of fidelity that allows for implementing and testing attacks and security controls.

The definition on a digital twin for information security evaluations in manufacturing given above consists of three major parts. The first part is the ability of the digital twin to accompany its physical twin throughout its entire product lifecycle. It refers to the fact that the digital twin is not a one-time creation that is only used for a short period of time. Rather, a digital twins is a continuous representation of its physical counterpart and evolves over time. This fact can be overlooked when discussing digital twins in the context of testbed design [41]. Testbeds, particularly those with an academic background, tend to discarded after an initial set of experiments is performed. However, for a digital twin, it needs to be ensured that the twin can be used over a prolonged period of time - especially in manufacturing with its long product life times [16, 18]. Integrating the digital twin in established processes and procedures that are well-established in manufacturing and industrial automation can ensure that the twin is setup to accompany its physical counterpart during its lifecycle [55]. The lifecycle of a product can range from the early design stages to the end-of-life decommissioning [28, 92, 93]. The digital twin is intended to be an ever-present virtual replica of the physical system that it represents. Ideally, this means that as the physical system evolves, the digital twin, correspondingly, is updated in real-time to reflect any changes and updates to the physical system.

This evolution of the digital twin alongside its physical twin is the topic of the second part of our working definition. It describes the ability of the digital twin to consume real-time as well as historical data depending on the scenario examined. This means that the digital twin is capable of accessing and processing data from various sources [55]. This can range from engineering prototype descriptions to live data collected during the operation in the field by the physical twin [231]. Real-time data refers to data that is collected and processed by the digital twin in real-time, for example, data collected from sensors on the shopfloor. This data can provide immediate feedback on the current status of the physical system, allowing for fast analysis of the system. This can, for instance, be useful in anomaly detection use cases where the behavior of a physical device potentially infected with malware can be compared against one that is known to be free of malware, i.e., its digital twin [232]. In addition to the consumption of real-time data, digital twins can also consume historical data. This describes the ability of the digital twin to process data that was collected and stored previously. This data is then played back into the twin in order to replicate the behavior at that previous point in time. Original conceptions of digital twins were only intended and built to consume real-time data [39].

However, consumption of historical data offers benefits to testbeds, e.g., for anomaly detection, trends and patterns can be analyzed if historical data is stored and used.

The transfer of data, either real-time or historical, is enabled by the data connection between physical and virtual twin [39] (see also Figure 2.2). Based on that data connection, it is possible to define levels of fidelity for the digital twin [42]. According to [42], levels of fidelity for digital twins can be defined by including the number of parameters transferred via the twins' data connection as well as the fidelity of each individual parameter. For example, parameter sets can contain variables representing measured quantities such as memory usage or network traffic and behavior that can provide useful context for the study of attacks and malware [38]. The fidelity of the parameters, for example, for network traffic, can be defined by a basic network topology and inclusion of only security-relevant events for a lower level of fidelity but can be increased to a detailed representation of the network, including all of the devices and connections, as well as environmental factors such as network latency and bandwidth for a higher level of fidelity. The total number of fidelity levels can be defined according to the application at hand and the use cases. This approach to multi-level fidelity can be used in testbeds for information security evaluations that contain digital twins [54, 28, 42]. By defining these different levels of fidelity, we can construct an information security testbed for security evaluations within manufacturing that includes a digital twin and is also open for future applications. To support digital twins with a high or maximum level of fidelity in the future, interfaces and APIs need to be included within the testbed in order to allow the usage of evolving tools and technology. However, some technological components such as network cables or switches may require replacement as the communication infrastructure in manufacturing environments evolves.

6.1.5 Design Considerations for Testbed Realization

In the context of our working definition for digital twin testbeds for studying information security in manufacturing, we conclude this section by discussing our design choices for testbed construction. For this, we start by discussing the intended usage of our testbed, following with by the scope of our testbed, and conclude with the evaluation methodology used for our testbed [33]. Each of this steps in the testbed design process is reflected on our working definition given in the previous section [28, 42]. Following a testbed design methodology allows for a systematic approach to designing and implementing the testbed, which can significantly en-

hance the quality of the testbed. This allows for a better evaluation of the testbed especially when it includes a digital twin as a well-designed testbed can simulate real-world conditions and scenarios to a better degree than other testbeds [18].

6.1.5.1 Intended Usage and Purpose

The authors of [59] discuss three potential categories for the use of testbeds in digital twinning research. These categories are attack simulation, penetration testing, and system security testing. Attack simulation is the common intended usage for most testbeds and corresponds to the analysis of attacks [33]. Penetration testing is in some aspects similar to impact analysis as both aim to identify and assess potential vulnerabilities and the risk they pose in a system or network [199, 60]. Penetration testing is the process where a human penetration tester is actively trying to exploit vulnerabilities within a target system or network. This way, the tester can determine how potential attackers might get unauthorized access to a system and, to some degree, what the results of this could be. Impact analysis, on the other hand, is a process that aims to determine and assess the potential consequences a breach in security with a subsequent cyber attack can have on a device or system. We also aim to understand the consequences an attack can have on a larger system, i.e., a connected manufacturing environment, and are not concerned with identifying vulnerabilities within a specific industrial component. Although, penetration testing per se can be a viable use case for digital twins on its own [28, 59], we focus on the aspect of impact analysis. Impact analysis can very well be conducted by the use of computer simulations, which allows for the specialization of relevant attack scenarios within our testbed [233]. System security testing refers to testing the security of a system including already existing and planned security controls.

These three purposes for a information security testbed in manufacturing, i.e., simulation of attacks, analysis of attack impact, and security control tests are relevant to the construction of digital twinning enabled information security testbeds. For designing and realizing a digital twin testbed for information security evaluations in manufacturing, requirements from all three of those purposes are needed. A thorough understanding of potential attack vectors is essential for testing effective security controls with digital twins. By analyzing potential attacks in a testbed, we can identify weaknesses in the simulated manufacturing environment. By simulating various security scenarios (see also Section 4.2), it is possible for us to determine whether the security controls can detect and respond to potential security threats.

Furthermore, understanding the potential impact of a successfully executed attack is essential for developing effective security measures. Digital twin testbeds can be used to simulate different types of attack vectors and assess the potential impact on the digital twin system and the production processes.

6.1.5.2 Scope

After discussing the intended purposes of our testbed, we continue with narrowing down the scope of our testbed. This includes defining the testbed's design principles as well as the coverage the testbed should provide. We start by discussing the design principles before expanding on the coverage of the testbed.

Design Principles For digital twins, fidelity must be considered as an important design principle [28, 42, 18]. Fidelity is a critical factor in digital twin testbeds as it determines the accuracy and realism with which the digital twin replicates the physical system it is modeling. The accurate representation of the physical twin is one of the defining characteristics of the digital twin concept. However, as pointed out previously (cf., for instance, Section 1.1), true high-fidelity digital twins are currently not achievable or are not published yet. However, if it is expected that the technological gap is closed and the underlying engineering issues are resolved, a testbed must include the capabilities to enable high-fidelity digital twins for future applications. This means, that the integration of high-fidelity needs to be possible within the architecture and infrastructure of out testbed. Also, the studied attack scenarios should yield more useful results when digital twins with different levels of fidelity are employed. For this, we define different levels of fidelity for our digital twin testbed (see next section).

Fidelity is not only related to the construction of credible and realistic digital twins or testbeds in general, it is also an important aspect in scientific experimentation [26]. The authors of [34, 27] discuss this in the context of constructing scientifically credible testbeds for manufacturing. Information security is not considered broadly by the their works but valuable insights in the design of the testbed can be gained. The definition of scientific rigorous experimentation used in [26, 34, 27] covers four parts: fidelity, repeatability, measurement accuracy, and safe execution of tests. Fidelity is discussed above by us. The three remaining parts also need to implemented in our testbed in order to allow reasonable experimentation. Specifically, repeatability introduces the requirement to our testbed that executed simulations given the same

set of starting parameters should produce identical or similar results. Measurement accuracy is related to this as it involves acquiring accurate and correct measurements from the testbed. This is, among others, supported by a reasonable definition of metrics [60]. Also, safe execution of experiments means that a safeguard, either by design or if necessary via physical means, needs to be in place during operation of the testbed [199]. It is worth noting that only about 10% of published information security testbeds for manufacturing *at best* consider all four required principles for scientific experimentation [33]. In fact, it appears that only [27] comprehensively follows the rigorously scientific approach outlined by themselves according to the survey of [33].

Continuing from the discussion of scientific experimentation, which captures the design principles of fidelity, repeatability, safe execution of tests, and measurement accuracy, we now discuss the remaining design principles, i.e., adaptability, scalability, and interoperability. In order to construct a testbed that can be considered adaptable, different design choices can be made. First, a modular approach to design of the testbed and its library supports adaptability of testbeds. Testbeds can be designed with a modular architecture where parts of the architecture, e.g., components or communication protocols, can be modified or replaced with ease. The choice of APIs and interfaces used within the testbed for connecting the different components together as well as for integrating external systems or tools is a worthwhile consideration for this [47, 38]. Second, an open architecture can further support adaptability of testbeds. Testbeds that rely on open source components and provide documentation of their architecture in addition to scientific publications enable the adaption of the testbed for different purposes. However, most components used in manufacturing are of a proprietary nature and it is likely that this remains the case for the foreseeable future [51, 32]. Therefore, open architecture should be promoted in the testbed to a degree that still allows usage of proprietary systems within the testbed. Third, the usage of virtualization or containerization as simulation technique can make a testbed more rapidly accessible as creation and transfer time can be reduced by the application of transparent layers. To conclude, testbed adaptability is important in order to provide a future-proof testbed architecture that is well-suited for research on different digital twin use cases.

Scalability is worthwhile considering for digital twin testbeds as they are expected to process large volumes of data [42]. Also, a high-fidelity implementation of a digital twin can potentially contain a large number of different (sub-)systems and

components [54]. A scalable testbed can be realized by different means. First, a distributed testbed architecture can aid in scaling up testbed components. Testbeds can be designed with a distributed architecture such that it enables the testbed to be distributed across multiple nodes or devices. For example, the emulation of PLCs within the testbed can be implemented on a device with higher processing power whereas a low-fidelity simulation of the physical manufacturing process can be executed on a standard office PC. By using such an approach, the testbed can scale up to larger systems by adding more nodes or devices into the distributed architecture. Second, as already outlined above in the context of adaptability, virtualization techniques can be used for the efficient execution of several instances of components within the testbed, e.g., PLCs or HMIs. Third, the deployment of testbeds on a cloud infrastructure, which can be easily scaled to meet changing requirements, is another supporting factor for scalability [46, 47]. Cloud computing is, in turn, supported by virtualization technology and is supportive of distributing computing paradigms.

Interoperability as a design principle is also relevant for the inclusion of digital twinning technology. The digital twin is a testbed component that is distinguished by particularly high fidelity [28, 42]. The fidelity of the digital twin is usually much higher than that of other components included in current testbeds [54]. The integration of such a component needs to be possible in order to make use of a digital twin in our testbed. Interoperability can enable this and is itself enabled by a number of approaches. First, the use of standardized protocols, data formats, and interfaces can enable interoperability between different testbeds. Usage of standardized components ensure that different testbeds can exchange data and, if necessary, communicate with each other, e.g., as a co-simulation [30, 49]. However, standardization within manufacturing in the field of information security is still evolving and is likely to change over time. Therefore, standardized components should be used where possible, e.g., the security standardization provided by the IEC 62443 [228] can be considered. Where this is not possible, inspiration from related domains such as automotive or critical infrastructure can be drawn [229, 230] or it can be resorted to best practices within information security [122]. Second, as already outlined above in the context of scalability, an open architecture for the testbed can enable the integration of new components and the modification of existing components. This, in turn, enables interoperability within the testbed and between different testbeds. Third, as discussed for adaptability, the integration of an API

and common interfaces allows testbeds to interact with other systems using the same API or interface. Furthermore, some newly developed APIs and interfaces in manufacturing can be expected to stay relevant for the next years and should be included within the testbed where possible [47, 46]. Fourth, the chosen methodology for evaluation of the testbed can promote its interoperability [33]. Testing and validation of interoperability can be ensured by including it within the testbed's evaluation. This way, it can be asserted with a certain degree of confidence that interoperability is given within the testbed.

Coverage From discussing the design principles we continue on to the next topic within testbed scope, i.e., the coverage of the testbed. The coverage of a testbed is itself divided into two aspects: the architectural items included within the testbeds library and the choice of simulation approach. We start by discussing compilation of the testbed library. The library for our digital twin testbed should contain components and communication protocols from all areas of industrial automation as high up as the SCADA systems (cf. Section 6.1.2.2). This allows, first of all, a holistic representation of a connected manufacturing environment. These environments with their industrial automation systems integrate multiple interconnected components and such as sensors, actuators, control systems, robotics, and data acquisition systems [55, 17, 196]. By including components from all areas of industrial automation, the digital twin testbed can provide a holistic representation of the entire system. This ensures that the interactions and dependencies among different components are contained within the overall simulation of testbed. Thus, providing a context for the digital twin to ensure that the digital twin is not regarded as an isolated system but rather a device integrated within the manufacturing workflow. Including components from all areas of industrial automation within the testbed library, furthermore, allows for a comprehensive understanding of the system as a whole, i.e., a system-level understanding. In addition to the study of the digital twin, it also enables the evaluation of the impact changes within the system have on the overall system, e.g., in the event of a cyber attack [60, 234]. Also, a library that includes components from the entire stack of automation layers (in our case: Layers 0-3, cf. Section 6.1.2.2) is better suited for providing a future-proof testbed realization. Industrial automation systems are most likely subject to change and evolution over time within the next years and decades [1]. By including components from all areas of industrial automation in the digital twin testbed, we can, at least to some degree,

future-proof our simulation environment. Most likely, existing technology and components are updated and enhanced over time instead of being replaced entirely due to the long operating times of industrial equipment in the field and the need for interoperability with legacy systems [18, 15, 16]. Thus, components of the library can be gradually updated to address these continuous but comparatively small changes to the devices and protocols. Therefore, we should include the following components within our testbed's library: the physical process executed by the manufacturing environment, sensors, actuators, PLCs or devices with similar functionality providing connectivity and control, HMIs as way to interact with the system, industrial communication protocols, and a data historian for data collection [51, 224]. In addition to this, the following information security items should be included for enabling information security evaluations: attacker nodes, representation of malware programs, and security controls or countermeasures (see Section 4.1). Note that we only speak of a representation of malware as the usage of real malware can be difficult to control and can also potentially be harmful to the physical twin [199].

Not all of the items in a testbed's library require the same approach to realizing their simulation. It can be reasonable to apply different simulation approaches to different items depending on the examined use case and the design goals of the testbed [224, 54, 33]. For example, for a testbed that considers cost effectiveness as their most important design principle may not consider a high-fidelity simulation worth achieving [227]. We also advocate the usage of different simulation approaches combined in our testbed. This way, different levels of fidelity can be achieved [27, 42]. We focus in our discussion on fidelity as this is one of the defining characters for digital twins [28]. The digital twin within the testbed should, thus, provide a sufficiently high level of fidelity. Preferably, the level of fidelity provided by the digital twin is at least equal or higher to the level of fidelity the other components implement. In order to determine a suitable level of fidelity for each item from the testbed's library for a specific scenario, different aspects need to be considered. First, the items used for the study of an attack scenario should be selected from the testbed's library. Not all items within the testbed's library need to necessarily be included within any simulation of an attack scenario. The items should be selected on the benefit they provide for each scenario. For example, the addition of a HMI is reasonable for a scenario where an APT plays back false data to a human operator in order to divert personal from the APTs presence or try to assert a certain response but may only introduce simulation overhead in scenarios where this is not part of

the studied attack vector [55]. Then, the characteristics of the manufacturing environment in focus need to be considered. For this, the characteristics and behavior of each system component within that industrial automation system should be assessed. For the assessment, factors such as complexity, dynamics, interdependencies, data availability, and computational requirements of the system's component need to be taken into account. This characterization can help in identifying the types of simulation techniques that are suitable for each component. Also, the availability and quality of data is important to evaluate within manufacturing. Data may not be available at all or only in insufficient quality [49]. It should be determined if a sufficient amount of data exists to support the implementation of certain simulation techniques. This is particularly important for digital twins. If the available data is not sufficient, efforts can be required to enhance the accuracy and reliability of simulations. For instance, the model digital twin can be validated with its physical counterpart, which is integrated as a HIL simulation.

6.1.5.3 Evaluation Methodology

In this section we discuss different evaluation methods within the VV&A area and how they apply to information security testbeds for manufacturing. We provide an overview over relevant methods, a selection of a suitable evaluation methodology for our testbed is met in the following sections. As discussed in Section 6.1.5.3, verification methods are techniques or processes used to assess whether a system meets its requirements or specifications [9, 33]. Verification ensures that a system has been built correctly, adheres to design specifications, and functions as intended. For our testbed that means that the testbed follows its scope, that is, implements its design specifications correctly and provides its intended coverage. Testing is a widely used verification method that involves executing the system under controlled conditions to identify deviations from expected behavior. In our context, that means, among others, that the testbed is executed within a safe environment. Various testing techniques are available, also for information security in embedded systems [235]. Here, unit or component testing focuses on testing individual components of the system in isolation from the main system. This can involve testing independent pieces of code with a small number of lines per code, e.g., functions or methods, to verify their correctness. Component testing can be automated easily and can help to identify some flaws within the testbed's implementation. Integration testing verifies the interactions and interfaces between different components or modules of the testbed.

It ensures that the individual components of the testbed function properly together and that data flows between them are established. This type of verification testing technique can be used to verify that digital twins interface well with the remaining testbed and that the data connection between the digital and the physical twin is established properly. Lastly, system testing is testing the entire system with all modules and components connected together that are required for simulation of a specific attack scenario.

Validation methods, on the other hand, are techniques used to assess whether a system conforms with its intended usage. Unlike verification methods that focus on ensuring the system is built correctly, validation methods focus on evaluating whether the right system has been built. That means, validation methods ensure that the testbed satisfies its intended purpose for usage of the testbed. One such validation method is prototyping, where a working model or a partial implementation of the testbed is created in order to validate its design and functionality. Prototyping can help in eliciting requirements, uncovering design flaws, and refining the system based on the prototyping test's results. Also, another promising method for scientific testbed validation is field testing. Field testing involves deploying the testbed in a real-world environment, e.g., by interfacing with a manufacturing environment and observing its performance under more authentic conditions than that available within an experimental environment. Field testing can aid in uncovering issues that may not be apparent during development or testing under controlled environmental conditions. Furthermore, a field testing setup can be useful to gather further requirements by potential testbed users. However, a scientific process to experimentation may not be given in a productive plant setup as the repeatability cannot be maintained with constantly changing environmental conditions that originate from a manufacturing system operated under economic considerations.

Following up on validation, accreditation methods refer to the processes and techniques used to evaluate and grant some sort of official recognition or certification to testbeds. This recognition or certification indicates that the accredited testbed meets certain established standards or criteria. Accreditation is typically performed by authorized accrediting bodies or agencies, i.e., a third external party. Accreditation can be used to claim same sort of compliance within a specific domain, e.g., by complying to the standard IEC 62443 within the manufacturing domain [228] or the standard ISO/SAE 21434 within the automotive domain [235]. This can be achieved with certification programs that involve a formal process through which the testbed

is assessed for the purpose to receive a certificate, mostly from a third-party like a certification agency. Such organizations define the certification requirements, establish assessment criteria, and conduct audits to accredit the testbed. Third-party audits are conducted by independent auditors or assessment organizations that are separate from the entity being evaluated. Another possible approach to accreditation that is used in academia is peer review. Peer review involves the assessment of a research work by qualified peers within the same field. Peer review serves as a mechanism for quality assurance and accreditation within academic research communities.

6.1.5.4 Testbed Design Strategy & Summary

In this section, we take the previous discussion on the design of information security testbeds in manufacturing into account and define a specific testbed design strategy that serves our interest in studying digital twins and information security evaluations in manufacturing. Especially, we take the considerations discussed within Section 6.1.3 into account. Also, we give a summary, where we highlight our design choices.

The intended usage and purpose of our testbed is primarily focused on providing attack simulation. In coherence with the outlined attack scenarios (see Section 4.2), we include attackers and tooling for the attacker that enables the simulation of basic attacks as well as APTs. Particularly in the latter case, i.e., the study of APTs, impact assessment is an important aspect for evaluating and estimating the consequences of a cyber attack on a large and connected system such as a modern manufacturing environment [60]. This is complemented by the integration, simulation, and evaluation of security controls within the testbed's library. Together, the intended purposes for our testbed, i.e., attack simulation, impact assessment, and information security control evaluation, provide a reasonable framework for a digital twin and information security evaluations within manufacturing [59].

For the design principles, we start with discussing fidelity as a central aspect of digital twins [28]. The definition of different levels of fidelity can serve of enabling different types of simulations while still providing a future-oriented testbed implementation [42]. Levels of fidelity refer to the degree of accuracy and completeness with which a digital twin testbed represents its physical counterpart. In the following, we define the levels of fidelity to be used for our attack scenarios.

Low fidelity: A low level of fidelity within digital twin testbeds provides a simpli-

fied or abstracted representation of the physical twin. This can be by the usage of simplified models or simplifying assumptions to simulate the behavior of the physical twin, and can very well not include all of its original components or present environmental factors. Examples for low fidelity simulations are simplified geometry or representation of physical components, approximate values for physical parameters (e.g., size, weight, or power consumption), or the inclusion of only basic environmental factors such as temperature or humidity. A low-fidelity digital twin testbed within an information security system can further provide a simplified representation of the network and communication architecture, such as a basic topology diagram. The testbed may simulate basic traffic patterns and security events, but may not capture the full range of transmitted communication. Low fidelity digital twin models are well-suited for the study of rudimentary attacks and can also serve as a means for testbed proof-of-concepts [60, 55].

Medium fidelity: A medium level of fidelity for digital twin testbeds can provide a more detailed representation of the physical twin. As this level, the use of more accurate models and the addition of further components and environmental factors can take place. However, simplifying assumptions or exclusion of certain components or functions of the physical are still present. Examples for medium fidelity simulations are more accurate geometry models for representation of the physical twin, more precise values for physical parameters, or an increased number of parameters transferred between the twins. A medium-fidelity digital twin testbed in the context of information security can provide a more detailed representation of the network, including more accurate models of the connected devices and their connections. The testbed may simulate more realistic traffic patterns and security events, but some of the overall traffic is still excluded. Medium fidelity digital twin models are suited for the study of more advanced attacks and their impacts, e.g., in the face of an APT threat [60, 55].

High fidelity: Finally, a high level of fidelity digital twin provides a highly accurate representation of its physical twin. At this level, complex models are applied and all of the physical components and potential environmental factors are included in the digital twin. Only a minimal set of simplifying assumptions is required. Examples for high fidelity digital twins are highly detailed geometry for all physical components, precise values for physical parameters with a complete transmission of all parameters, and comprehensive environmental factors (e.g., such variables like chattering in case of a tooling machine [183]). A high fidelity digital twin testbed for

information security use cases should provide a highly accurate and detailed representation of the network where all devices and interconnections are included. Factors such as network latency and bandwidth are all included in the representation of the communication network. In principle, an information security testbed with a high-fidelity digital twin can be used to implement advanced information security use cases [28]. However, high-fidelity digital twins are currently not achievable [42, 18]. These levels of fidelity given above are not necessarily mutually exclusive meaning that different aspects of the digital twin testbed can have different levels of fidelity [54]. For example, a digital twin testbed can have a high-fidelity representation of a PLC but a lower fidelity representation of most environmental factors. Ultimately, the level of fidelity required for a digital twin testbed depends on its intended use and the purpose behind its construction. A higher level of fidelity can be necessary for certain applications, such as predictive maintenance or shatter detection [46, 47], while a lower level of fidelity can be sufficient for initial testing or prototyping. By choosing the level of fidelity for variables such as the complexity of the network topology, the accuracy of the traffic patterns and security events, and the range of possible threats and attack vectors, a digital twin testbed for information security can be customized to suit the needs of the attack scenario studied. This can help in ensuring that security controls can be suitable within the given context.

Continuing with the other design principles, we also consider those that are related with scientific experimentation [34, 27]. We already discussed fidelity above, the remaining of these design principles are measurement accuracy, safe execution of tests, and repeatability. Repeatability is best evaluated within testing of the testbed. If the repeatability is not given, the underlying issues can, thus, be identified and corrected. The safe execution of tests can require some physical measures as well. In our case, physical measures are necessary as our target of evaluation for the digital twin is a robotic arm as is discussed in more detail below [55]. Security testing in general can have quite disastrous results as outlined in Section 6.1.2.1 [199]. Therefore, we must ensure that users, other people, and equipment are protected from potentially escalating behavior of the device under test. The robotic arm used by us, consequently, is included within a protective encasement that enables its free movement but prohibited interaction with anything outside that encasement (see background of Figure 6.3). The remaining design principle related to scientific experimentation is measurement accuracy [26]. Measurement accuracy is also supported by the definition of usable and reasonable metrics [164, 196, 55]. In the context of our attack

scenarios given in Section 4.2 and that we set impact assessment as one intended purpose for the testbed, we propose a metric for assessing the impact an attack can have on a production system. The metric is based on our previous work and summarized here [60]. We measure the availability of a manufacturing system in the context of dependable systems [48] (see also Section 4.1.2). The availability of manufacturing systems is measured within control and reliability engineering in order to provide a statistical estimate on how long the machine might still be operational before requiring maintenance [236, 45]. This can be extended upon to also include the degradation introduced by malware [60]. This is achieved by introducing additional degradation factors to the equation. The numerical estimates are based on published studies from literature [234].

We now discuss the remaining design principles that are relevant to our testbed, i.e, adaptability, scalability, and interoperability. We summarize the requirements based on our discussion above and do not discuss each of the three remaining design principles by itself. Also, we take some of the discussions on the choice of simulation technique ahead. First, we aim to include virtualization techniques within our testbed where possible. For instance, PLCs or HMIs are components that can be virtualized with comparable low effort as there are libraries and tooling available that support this [50, 51, 237]. Therefore, we should include virtualization or emulation where it is possible within our testbed. Second, the promotion of an open architecture is reasonable in order to provide an extensible and broadly accessible testbed architecture that can be developed further on in the future. However, open architecture requirements can be weakened when this supports the integration of common interfaces that are used within the manufacturing industry and are expected to be worth considering for information security applications within the foreseeable future. That includes the support for certain standards and bests practices from the field of manufacturing. Third, for the overall testbed architecture we choose a modular and distributed approach. That implies that the different components of the testbed's library should be able to be exchanged easily and to interact with each other in a distributed manner, e.g., when some library components are executed on different devices that are separated from each other by geographical means.

We now briefly discuss the coverage of the testbed. As indicated, our testbed should include the physical process executed by the manufacturing environment, sensors, actuators, PLCs or devices with similar functionality providing connectivity and control, HMIs as way to interact with the system, industrial communication

protocols, and a data historian for data collection (see also Pages 189ff.). We know discuss the choice of simulation approach for each of these library items. As indicated above, devices such as the PLC or the HMI should be implemented with virtualization or emulation techniques in order to promote fidelity and support for measurement accuracy within the testbed. The same is true for the sensors and actuators. Especially the sensors serve as the main data source within the manufacturing environment. That is why the data collected from them is required with a high resolution. Otherwise, some applications building up on these data may not function within expected parameters [46, 47]. For the implementation of the physical process executed by the manufacturing environment we choose an approach based on standard simulation techniques [30, 238]. Building a high- or even medium-fidelity simulation is beyond the scope of our work. Also, the data historian can be implemented rather rudimentary as its sole purpose within a testbed is the collection of data. Therefore, the implementation of a simple database is sufficient for our intended testbed purposes. The communication protocols and other interfaces, on the other hand, are best implemented with their default configuration meaning as a COTS device. The communication stack of a network protocol for the usage within manufacturing can be implemented without much effort and should therefore be included to further promote interoperability [225, 21, 22, 17]. That leaves the digital twin as the reaming component within the library. Digital twins require high resolution of their data but may be difficult to verify [42, 18]. Therefore, we include the digital twin in a double fashion: a simulation model of the real device and the real device itself. This means, our digital twin is included within the testbed via hardware-in-the-loop (HIL) as simulation technique [55]. HIL allows the digital twin to interact with real hardware components and systems, creating a more realistic simulation environment. This enables testing and validation of the digital twin's performance under real-world conditions, providing more accurate and reliable results. In addition to this, HIL also can be used within the verification of the testbed.

That leaves the testing methodology as the final part to discuss within our testbed design strategy. We aim at building a reliable and believable testbed for digital twins in manufacturing. Although the overall implementation effort is high, it is not comparable to implementation of large-scale software development. In small-scale systems with limited complexity and a narrow scope such as our testbed, automated source code analysis is not considered necessary by us. Rather, we resort to testing of individual components in isolation, integration testing, and system testing as means

to verify our testbed's design. For validation of the testbed, we rely on prototyping as a validation method rather than field testing. Prototyping allows for a rapid feedback loop, allowing us to refine our testbed design quickly and enhance the testbed further. Also, prototyping enables testing under controlled conditions, ensuring to evaluate the repeatability of our testbed, which may be difficult within a live production environment during field testing. Field testing may, however, be conducted at a lager stage at suitable partner locations where access to a live production facility is possible. Accreditation is, for the most part, not deemed necessary for our testbed. Formal certification is unnecessary for a testbed that is ongoing development and should only be conducted when go beyond the prototyping stage. However, we do rely on peer reviewing of our concepts by the research community via publication and critical discussion of our work.

The testbed design strategy outlined by us above is the basis for the subsequent implementation and evaluation of the testbed, which is the topic for the remainder of this chapter. By following our design strategy, we can establish a solid foundation that guides the development and assessment of the testbed and ensures that it is suitable for its intended purpose. We discuss the practical aspects of implementing the testbed, including the selection of hardware and software components, the integration of the digital twin, and the establishment of evaluation criteria. Furthermore, the evaluation process is examined, covering various metrics and methodologies employed to assess the performance and accuracy of the testbed. By adhering to the testbed design strategy, the subsequent discussions aim to provide insights for the realization and validation of the testbed, thus, contributing to advancements in this domain.

6.2 Baseline Architecture for Implementation

In this section, we discuss our baseline architecture for implementation of our testbed. The baseline architecture is based on the conceptual system architecture discussed in Section 4.3. The conceptual architecture is an abstract representation that conveys our basic principles, goals, and directions we take in our research. The conceptual architecture serves as the starting point for the development and implementation of our digital twinning testbed and the integration of information security evaluations. With this, we establish a common foundation for our further steps. In addition, a baseline architecture is a more specific representation based on the conceptual architecture. It includes more detailed information and specifications that are required for realization of the concept. Our baseline architecture is an elaboration of the concept and involves the concrete setup for our testbed and makes some technical definitions for implementation. For the purpose of this chapter, i.e., the design, construction, and evaluation of the information security testbed, we partially detail the reference architecture as seen in Figure 4.6. Specifically, all the components from the Digital Twin Framework seen in the upper third of Figure 4.6 are part of the baseline architectural design discussed here. Furthermore, the Manufacturing Environment seen in the middle of Figure 4.6 is also part of the baseline architecture excluding the Edge Device. That also applies to the Cloud Infrastructure and the remaining components following on the bottom of Figure 4.6. Their design and implementation as well as the integration of the cloud-enabled components with the testbed is discussed in the following Chapter 7.

For now, we present our baseline architecture that realizes the testbed's design goals (see Section 6.1). The baseline architecture is shown in Figure 6.2. We discuss the individual elements of the baseline architecture from top to bottom and from left to right. Thus, we start by the digital twin. When integrating a digital twin in a baseline architecture we need to consider several aspects that are related to the concept of a digital twin [28, 59, 18] (see also Section 6.1.4). First of all, we need to consider the development of a suitable digital twin model that represents the physical twin with a sufficient amount of fidelity, i.e., that is able to implement a level of fidelity as, for example, those defined by us in Section 6.1.5 [42]. The digital twin is supposed reflect the architecture and behavior of its physical counterpart. Thus, we include a kinematic representation or model of the digital twin within the baseline architecture. A kinematic model is a representation of the motion and

movement of physical twin. The kinematic model focuses on geometric aspects that are related to motion and movement, such as position, velocity, and acceleration. The dynamics of the dynamics or underlying physics involved are not taken into account by the kinematic model. It does not include a complete physics model that also considers forces and torques causing the motion. This is provided by the overall simulation of the manufacturing environment that also includes a physics engine (not seen in Figure 6.2). Rather, the system in a kinetic model is typically represented as a collection of rigid bodies connected by links, e.g., via joints between the rigid bodies. The interaction between these bodies and their links are the focus of the kinematic model. They allow for the determination of their positions and orientation relative to each other as well as the overall movement of the system. With this, a simulation of the kinematic model allows for an accurate representation of the motion and movement for the physical twin. Thus, our digital twin model can simulate the behavior of its physical counterpart in regards to position, velocity, and acceleration. Also, simulation engines for kinematic models provide a decent visual representation that enables visualization and monitoring of the digital twin's behavior. Providing visual feedback of the presentation can greatly increase the acceptance of the entire testbed [33].

As digital twins, furthermore, consume data from their physical counterpart (either historical or real-time data, see above), we further need to consider the data connection for the twin [28, 59, 39, 42]. Our concept for interaction between the twins includes controllers attached to each twin and a connecting element between them. For the digital twin, this is a virtual controller that is in turn connected to the controller switch. The controller switch is part of the data historian. The data historian is a database within the simulation of the manufacturing environment that collects and stores data from various sources. The data historian is tasked with collecting and storing data produced by the physical twin, i.e., the robotic arm in our laboratory setup. Furthermore, the data historian offers data processing and also provides analytical capabilities to handle and analyze the incoming data streams. This includes data streams from both, the digital and the physical twin. The data points or data sets collected from the physical twin are processed and partially filtered by the controller for the physical twin, i.e., the robotic arm controller. In our case, this controller is the controller required for operation of the robotic arm. Its controller acts in unison with the virtual controller than can feed collected data via the controller switch into the digital twin. Both controllers can be employed in-

terchangeably [47]. The controller switch enables these alternation between virtual and robotic arm controller. This provides us with the possibility of comparing the results of information security tests conducted with digital and physical twin with each other. It is worth noting that the robotic is required to be operated within a so-called safety shell that ensures safe integration of the robotic arm within our laboratory environment [27]. Security testing can have unintentional or unforeseen side effects on the behavior of the tested component, such as our robotic arm [199]. So, for example, in [199], the authors report that a penetration test of a large robotic arm caused the arm to move unexpectedly and rapidly around. No personal was located close to the arm during the test but such unforeseen behavior can possibly cause a lot of harm to a human. We need to avoid this in our testbed and laboratory setup.

The twins are embedded in the larger context of the simulation of the manufacturing environment. That means, that there are also other communication entities present within the testbed. The communication between these entities is managed via the network bus of the simulation. In the context of a simulation, this network bus summarizes the mechanisms and underlying infrastructure used for communication and data exchange between the entities of the simulation [30]. With the infrastructure provided by it, the entities of the simulation can interact with each other and exchange information. Thus, the network is a means for coordination of their activities and behavior. This network bus is not to be confused with our discussion on fidelity in Section 6.1.5.2 [224, 18]. Though the network bus can be used to also integrate different simulation techniques, e.g., in the context of a high-level simulation architecture [30], the network bus in our baseline architecture refers to processes executed in the background of the simulation, thus, enabling the simulation as a whole. This implementation of the internal communication or network bus can vary depending on the simulation framework and modeling techniques used. As we employ a behavior-based model of the manufacturing environment (see Chapter 5), we include an event-based simulation system for the simulation of the manufacturing environment. Here, the entities participating in the simulation generate events by themselves and respond to events received that are generated by other entities. Events represent specific occurrences or changes in the state of the simulation [74, 114]. Typically, these events are consumed and processed by other entities and, therefore, can trigger actions in other entities. The network bus handles the routing and delivery of events to the appropriate entities. For the most part, the

entities referred to above are located within the model of the manufacturing environment. This model is a direct implementation of the reference scenario detailed in Section 5.3. Thus, it is structured alongside production lines that contain individual workstations. The digital and physical twin are one of these workstations. Again, the baseline architecture is for outlining the implementation of the testbed including its underlying simulation infrastructure. For that reason, the components enabling digital twinning are not represented as a part of the model of the manufacturing environment in Figure 6.2 as their implementation and integration into the simulation differs greatly from the other entities. Logically, the simulation handles the digital and physical twins as a part of the respective production line. Lastly, the remaining entity within the simulation of the manufacturing environment is the attacker. They are represented by the attacker node. The attacker node does not necessarily need to participate actively within the simulation. Active participation within the simulation, at least in our testbed, does require issuing of events. Depending on the attack scenarios, the activation of encoded actions within the workstation can suffice (see Section 5.2.4). The realization of the attack scenarios is the topic of the following section.

6.2.1 Attack scenarios

In this section we discuss the realization of the attack scenarios that are first introduced in Section 4.2.2. Specifically, we focus our attention on two attack scenarios given in in Section 4.2.2. These two attack scenarios are Attack Scenario 1 (Sorting, see Section 4.2.2.1) and Attack Scenario 2 (Sawing, see Section 4.2.2.2). The remaining scenario, i.e., Attack Scenario 3 (Data Sharing, see Section 4.2.2.3) is related to data acquisition and discussed in Chapter 7.

For the attack scenarios discussed in the context of our baseline architecture for implementation of the testbed, we mainly need to consider two aspects, that is, the implementation of the attack scenarios as well as their evaluation. Implementation of the attack scenarios means the additional requirements that may be introduced by the attack scenarios and that need to be considered by the implementation of the testbed. Of course, our testbed is conceived in order to cover a broad range of attack scenarios for our study of information security evaluations on manufacturing, those additional requirements should be few and only have, if any, a small impact on the overall testbed design. In order to derive those requirements, each attack scenario needs to consider the following specifics in context of their scenario. This

can start with the actual execution of the attack needs to be specified. This is best done by defining a step-by-step process of how the attack is going to be executed within the testbed architecture. For this, a sequence of actions with the involved tools is derived. These tools include also attack tools and related resources and need to be identified. For example, distributed denial-of-service attacks can require a different type of attack traffic generator than attacks targeting a single component [38]. Examples for attack tools can include network scanners, frameworks used in penetration testing, vulnerability scanners, or custom attack scripts. Such attack tools usually also require the payload for the corresponding attack. They can already be included within the attack tool, e.g., in the case of penetration testing frameworks, need to be developed by the researchers, or obtained from malware found in the field. In the later case, caution is advised as malware obtained via field samples can have additional or hidden functionality that is not observable in any environment, e.g., virtualization detection mechanisms of the malware can be overlooked when the malware is obtained from non-virtualized environments. Overall, payloads for attacks can include malicious code, exploit scripts, crafted network packets, or any other malicious data capable of exploiting vulnerabilities. The simulation of the attack scenarios within a controlled test environment further needs to include the components for that attack scenario. This can involve setting up additional virtual machines, emulators for certain network protocols, or any other components and interactions required for a believable simulation of the attack. Lastly, the collection of the data produced by the simulation needs to be considered for each attack scenario. Mechanisms to collect relevant data during and after the attack simulation need to be conceived and realized. This can require specific mechanisms for a given attack scenario such as logging network traffic, system events, or error messages. Furthermore, any other relevant information for analyzing the impact and effectiveness of the attack must also be considered [60].

These aspects of implementation are related to the evaluation of the attack scenarios and of the results generated by the testbed. Data collection is required to determine the success of an attack. This can be achieved by defining metrics that provide a measure for the attack's success within its attack scenario. Such metrics can focus on quantifying the extent to which an attacker achieves its goals. These metrics can include measurements and derived variables such as the percentage of attacked systems, the time taken for attack execution until the goals are reached, or the amount of confidential data accessed. Closely related to these attack success

metrics is assessing the impact of an attack. Impact assessment, in contrast to attack success measurement, focuses on evaluating the consequences as well as the severity of an attack on the production environment it is targeting [60, 55]. The intention of impact assessment is to understand the effects an attack has as the result of its successful execution. This can include factors such as service disruptions, financial losses, or compromised sensitive data. These are all tangible impacts of an attack on a manufacturing environment. Also, intangible effects of cyber attacks such as reputation damage could be included but are, at least in the context of our testbed, out of scope [183]. Further out of scope are activities related to documentation and derivation of subsequent actions. These include identification of vulnerabilities exploited during the simulation of the attack scenarios. Furthermore, a structured risk analysis that aims to understand the potential consequences and likelihood of the attack scenarios is not conducted by us. Our attack scenarios are based on documented cases that are, in principle, possible to occur again (see Section 4.2 and Section 2.1.3 for reference). Mitigation strategies are discussed by us in the context of the simulated attack scenarios. However, although the strategies and security controls discussed by us can provide mitigation for the specific attack scenarios, we do not claim that this constitutes a comprehensive security strategy.

For the remainder of this section, we discuss the specifics for implementation and evaluation for both attack scenarios, Attack Scenario 1 (Sorting) and Attack Scenario 2 (Sawing, see above). Our focus is mostly on tooling and metrics.

6.2.1.1 Attack Scenario 1

Attack Scenario 1 (Sorting) is an attack scenario is an attack scenario that explores an attack vector with the intend ot cause direct damage to manufacturing equipment (see Section 4.2.2.1). The attacker targets the availability of the manufacturing equipment and aims to reduce the time the system is available for correct service (or any service at all). This is achieved by directly targeting an individual component located within the manufacturing environment. The target of the attack is a specific component but can, in principle by any component located within the simulated production facility. This reflects our attacker model of the disgruntled employee (see Section 4.2.1.1). This type of attacker aims to cause a large amount of damage but at the same time tries to limit its own effort and complexity of the attack vector [19, 193]. The payload for the attack is, furthermore, delivered by a dedicated attacker node that uses a variety of different communication channels for remote and local

delivery of the payload. This means that in our discrete assembly manufacturing process, which is divided upon multiple production lines, the attack can be executed on any manufacturing equipment present. For simplicity, we assume that only one type of manufacturing equipment is used along the production lines. That means, that the manufacturing equipment present at any given node (cf. Figure 4.3) is the same for all nodes. This assumption is reasonable, e.g., for assembly lines in automotive or furniture production that often employ the same equipment model such as robotic arms [69, 205]. Furthermore, the assumption does not limit the testbeds scope as the construction of more complex and diverse production is still possible by adding more models or digital twins to the testbed's library.

The attack scenario executed in Attack Scenario 1 targets the availability of one or several production lines located within the same manufacturing environment [60]. As the attacker's goal is to cause as much service disruption as possible, the overall percentage of downtime is their primary measure of success. Thus, we measure the percentage of total operation time where the production line is unavailable as a result of the attack. This does explicitly not include unavailability due to scheduled maintenance. Cyber attacks can, however, be regarded as unscheduled downtime in which the machine tools and manufacturing equipment require additional maintenance [111]. For example, if the production line is operated for 24 hours a day and an attack on it results in the production line being offline for 6 hours, the overall downtime percentage is 25% per day. Continuing from that example, if another identical production lines is also affected by the attack with a total downtime of 75% (i.e., 18 hours), the overall downtime for both production lines and, thus, for the manufacturing system, is 50%. This touches another possible attack success metric that is related to the overall downtime, that is, the time for recovery of the system. It measures the time taken to restore the production line to continue with its expected operation after the successful attack. The recovery time can also serve as a measure for the efficiency of any incident response or recovery processes such as reinitialization procedures [48] (see Section 5.2.5). Another possible metric that can be used within this attack scenario is the total number of affected systems. For the attacker, this can be a measure on how far his attack has spread and, thus, providing a measure of success for the attack the more systems are affected.

The total number of affected systems can also be used as a measure the impact an attack has on the manufacturing environment. Deriving the number of affected systems can provide an assessment for the scale and scope of the attack's impact.

This can help in understanding the reach of the attack within the production line infrastructure and evaluating the potential consequences. One consequence of a cyber attack can be reduction in output produced by the manufacturing environment. This metric can measure the reduction in production output as a result of the attack, e.g., by comparing it to historical production output data. This way, the number of products or units that could not be manufactured due to the attack can be estimated. This can be used to provide further metrics for attack impact assessment, e.g., evaluation of the financial impact resulting from the reduced production output [111]. Factors such as lost revenue due to non-manufactured products can be considered in the calculation of the financial loss. Also, increased operational costs as a result of unscheduled maintenance or penalties for failing to meet contractual obligations can be considered in calculating the financial impact of a cyber attack.

6.2.1.2 Attack Scenario 2

Attack Scenario 2 (Sawing) is an attack scenario that examines the damage that can be caused when targeting the integrity of the production process or parts of if (see Section 4.2.2.2). As with Attack Scenario 1 (Sorting, see above) the attacker tries to cause damage with its attack but has a different scope and approach. Instead of causing immediate downtime to the production equipment, the attacker considers a long-term approach for execution of its attack. Also, they target the manufactured products rather then the machinery itself. Thus, their goal is to reduce the product's quality by impairing the production process. This reflects the attacker model that considers a sophisticated APT as the adversary [19, 23]. The APT establishes itself within the production system, e.g., in the MES or at a PLC connected to a tooling machine. As the MES and other management systems are custom-mode components and systems that are realized differently within different manufacturing environments [60], it appears reasonable for an attacker to target standardized components such as PLC devices [13, 103, 68] (see also Section 2.1.3 and Section 2.1). Therefore, a less complex simulation of the MES suffices for this attack scenario as our focus is on the tooling machine and its directly associated devices. This in turn means that the model of the tooling machine needs to provide a high accuracy for its simulation. This is provided by a medium-fidelity digital twin in the context of this work as discussed in detail in Section 6.1.4 and Section 6.1.5.4. Additionally, a simulation of the production processes is needed that captures the illegitimate

alterations to the manufactured product. This can be realized by kinematic and physical models of the production environment [55].

These alterations in product quality are also one of the measures of success for the attacker [164, 196]. This can be expressed by the defective product rate, which is a measure of the percentage of defective products produced as a result from the attack. The defective product rate quantifies the impact of an attack on the overall product quality. It assesses the percentage of products that do not meet the quality requirements. This is related to another metric which counts the number of production parameter violations. With this metric, the number of manufacturing tolerance violations caused by the attack is observed. Manufacturing processes often have specific tolerance limits that need to be adhered to. An attack targeting the integrity of the production process and the products' quality is likely to cause exceeding of these limits during manufacturing. Trespassing these process boundaries often leads to quality issues within the produced goods. Another measure of success is the time the malware can remain its cover, i.e., stays undetected by security controls and human operators. This detection evasion time assesses the success of the APT in evading detection. The time it takes for the attack to be detected by existing security systems or by pure chance is measured for this. The longer the APT group remains undetected, the higher the potential impact it can have on the overall product quality, thus constituting to a measure of the attacks success. Then again, the detection evasion time also describes the amount of time the attacker can successfully carry out the attack. For a prolonged campaign that is planned in advance and can be intended to be in the field for several years, maximizing this time frame is an important attack success metric.

The impact of such a prolonged threat campaign can be more difficult to assess. For example, it is unclear how successful threat campaigns like Stuxnet have been in retrospect [107, 108]. Though Stuxnet undoubtedly caused a disruption in the manufacturing process, it is currently not known by what exact amount of time the project was delayed. However, in more concrete scenarios where products are manufactured and sold to customers, some estimates on the impact a cyber attack on product quality can achieve can be made. For financial impact estimates, the product recall costs can be taken into account. They quantify the financial impact resulting from product refund requests due to compromised product quality. An increased number of product refund requests can also serve as an early indicator for an ongoing APT within the production facility. Related to product refunds are

product recalls and their associated costs. Product recalls cause additional costs for the manufacturer. This additional costs can include logistics, customer notifications, replacement or repair expenses as well as potential legal fees like court settlements. From these more tangible impact assessments, intangible impact assessments can be derived. For example, customer satisfaction and loyalty could be measured and estimated when enough data on product sales and customer recurrence is available. Such intangible impact assessments are, as discussed in Section 6.2.1, out of scope for us.

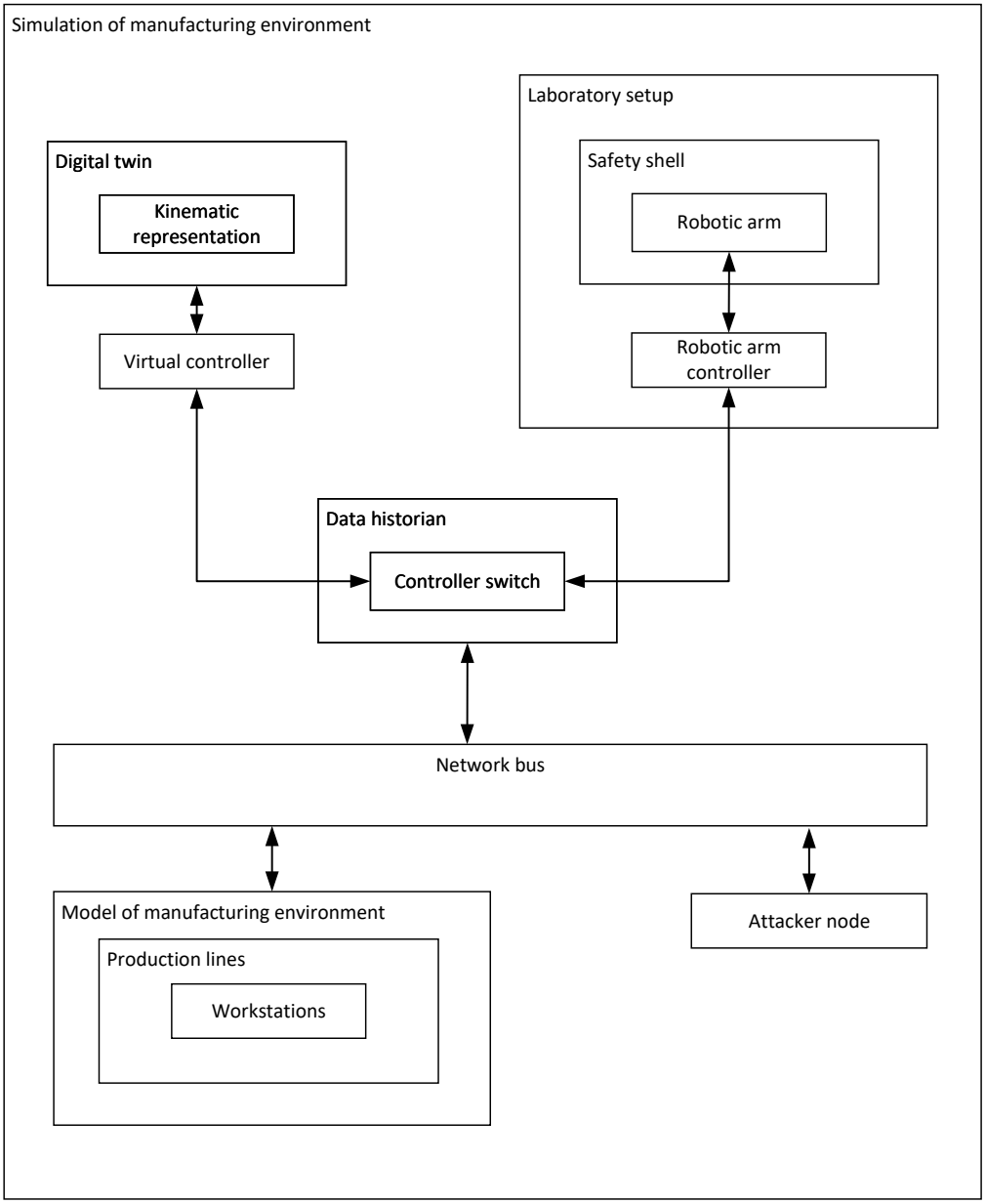


Figure 6.2: Baseline design of the testbed's simulation architecture with digital twin, physical twin, and manufacturing environment.

6.3 Implementation of the Experimental Testbed

In this section, we discuss the implementation of our testbed. Our implementation follows the design of the testbed given in Section 6.1 and realizes the baseline architecture as described by Section 6.2. We start by describing the programming languages, technologies, and tooling used in Section 6.3.1. We discuss our reasoning for our choices in implementing the testbed and discuss how those choices align with the goals and requirements of the testbed derived earlier in this chapter. Following up on the technical basis, we discuss the concrete realization of the testbed's infrastructure in Section 6.3.2. There, we describe the system architecture and specifics of the implementation ranging from modification to the technology stack to the core functionalities of the testbed. After implementation of the testbed is concluded, we discuss how we tested the functionality of our testbed in Section 6.3.3. In this section, we describe how we ensured our testbed is functioning in accordance to our design strategy of Section 6.1.5 before continuing on with the actual execution of attack scenarios in Section 6.2.1.

6.3.1 Technology Stack and Testbed Foundations

In this section we discuss the technology stack our testbed is built upon. This is the technological foundation for our architecture first given in Section 4.3 by Figure 4.6 and influences how experiments are executed with the testbed. Note that the edge device and the related cloud infrastructure are not discussed in this chapter. Rather, the realization of the privacy-preserving communication channels and its associated technology stack is the topic of the following chapter. However, as both parts of the architecture are intended to function in combination with each other, technological choices and considerations met by us in this section are also relevant for the realization of the edge and cloud components. The technology stack for the testbed involves hardware as well as software. As our testbed is strongly reliant on software and most components can be realized by software such as simulations, emulations, or virtualization, we begin by discussing software first followed by a discussion of the relevant hardware used by us.

The software implementation is based mainly on the robotics framework Robot Operating System (ROS) [239, 240]. It is not an actual operating system like Microsoft Windows or Linux as can be expected by its name. Rather, ROS is a framework for building robotic systems. The framework of ROS includes a collection of

tools and libraries that are intended to aid with development and deployment of software specifically intended for robots. ROS is a widely used middleware in the robotics research community and can be used for research, prototyping, and also for industrial applications. Lower level hardware details of the specific robots used are abstracted within it, which allows to focus on higher levels algorithms and tasks. From its functional view, the ROS software framework centrally controls all modules and functional blocks that are implemented within the ROS framework. It is open-source, accompanied by a comprehensive documentation, and actively developed as of compilation of this thesis. The hardware abstraction offered by it supports us in keeping our testbed open to future extensions, where other scenarios, hardware components, and simulation tools may be included. Also, ROS is well-suited for the implementation of digital twins with different levels of fidelity as ROS offers an extensive ecosystem.

In regards to testbed development, ROS offers some advantages that reflect with our testbed design strategy [48] (see Section 6.1.5.4). Several design choices from our testbed design strategy are also reflected within ROS's architecture. For one, ROS promotes a modular approach to development. This allows us to integrate different sub-systems and distribute complexity among those sub-systems or sub-components. This results in smaller, reusable software components. Within the terminology of ROS, these are referred to as nodes [241, 242]. Nodes within ROS can, thus, be independently developed and tested and then being integrated into larger systems for subsequent testing or roll-out. This modular design of ROS's overall architecture encompasses flexibility, scalability, and interoperability, which are considered relevant design principles for our testbed. The nodes within ROS communicate by a publish-subscribe messaging system. Here, individual nodes communicate with each other by one node publishing messages assigned to a certain topic while another node can subscribe to messages for topics relevant to it. This decoupling of the communication between the nodes is beneficial when integrating several different sensors, actors, and software processes such as simulations together into one testbed [55].

In addition to this, ROS offers various benefits in regards to the integration of digital twins. As mentioned above, the modularity in ROS's framework architecture enables the integration of different sub-systems into the testbed environment. For digital twins, this enables the integration of different simulation modules that provide data with different levels of fidelity. For example, a medium-level fidelity simulation of a PLC with its attached actuator can be operated together with low-

fidelity simulation of the manufacturing environment where the medium-level fidelity OT component is located in [54]. Furthermore, the message passing system of ROS and how it handles data exchange fosters the integration of digital twins into a dedicated testbed environment. With the publish-subscribe messaging system different components of the digital twin can exchange data with each other. The various data generated wihtin the testbed, e.g., simulation data, sensor measurements, or control commands, can be collected and stored for analysis. Furthermore, data can be seamlessly communicated between virtual and real devices such as controllers for OT equipment allowing for an easy switching between real and virtual components. This way, the data sources and data sinks can be adjusted within the testbed when experiments are executed. Lastly, ROS provides mechanisms for processing and handling real-time data as well as synchronization in real-time. This is essential for digital twins as it potentially allows for realization of high-fidelity digital twins at a later stage [42].

ROS offers a comprehensive and growing ecosystem that aids us in realizing our testbed with integration of the digital twin. The ecosystem of ROS includes open-source software libraries, tools, and packages that can be used for testbed and digital twin implementation. Among it, various simulation environments, robot models, or control algorithms can be found. By using these resources, it allows us to focus on the integration of information security evaluations into the testbed. In general, the integration of information security into ROS is not as advanced as other parts of its ecosystem [55]. We now describe the libraries, tools, and packages used by us for testbed realization. One of these software packages is the simulation environment *Gazebo*. Gazebo is a physics-based robot simulator. It is used for developing and testing robotic systems or algorithms. With Gazebo, a dynamic environment simulation can be implemented in which robots are integrated and simulated. This can allow for evaluating their performance and behavior, e.g., during the development phase or early stages of prototyping prior to hardware implementation. A key feature of Gazebo is its physics simulation. Gazebo simulates the physics of objects together with their interactions, which provides representations of some real-world environments and allows to examine robots' dynamics in the presence of environmental factors. These factors can include gravitational forces, collisions with fixed or moving objects, friction that occurs during movement, or other physical properties. Gazebo can also simulate various different sensors such as cameras, light sensors, or contact sensors. This is particularly useful in industrial environments where, for

example, conveyor belts often employ different sensors to facilitate a proper flow of materials through the production lines. Using the sensors simulated by Gazebo This allows developers to generate simulated sensor data, including images. With this, sensor data such as distance or temperature measurements can be collected from the simulation in order to provide a data basis. To summarize, Gazebo is very well suited for providing a simulation of the manufacturing environment in which the digital twin is located.

The digital twin itself can also be realized within the simulation environment of Gazebo. In order to realize a medium-level fidelity for the digital twin, we employ additional tooling from ROS's ecosystem. The first software package we use is a visualization tool called *RViz*. RViz is used within ROS applications for visualizing and analyzing data. It provides a 3D environment that allows us to visualize and analyze data in a virtual environment. It can visualize sensor data captured from robots or from Gazebo, movement and trajectories of robot models, among other things. With RViz, we can understand and analyze the digital twin's behavior, the state it is currently in, and the data produced by it. When used in combination with each other, Gazebo and RViz can offer a comprehensive development and visualization platform for digital twins. These is utilized by us in the following manner. As already mentioned above, Gazebo serves as the main simulation engine for the manufacturing environment with its digital twin. Here, Gazebo provides and manages the physics-based modeling of the virtual environment together with sensors and actuators. RViz is then used to render and visualize the digital twin's robot model. That includes its physical components, joints, and movement. Additionally, RViz visualizes the virtual environment surrounding the digital twin, including all objects such as obstacles or other production equipment. Also, RViz can be employed to visualize the sensor data generated by the digital twin within Gazebo. RViz can also be used for interactive analysis of the simulation enabling us to stop the simulation and to examine trajectories and component states within the simulation model. This way, the real-time analysis of data within the visualized scene is also possible. As Gazebo and RViz are both part of the ROS ecosystem, they integrate seamlessly with ROS. This allows us to use the communication infrastructure of ROS for data exchange and communication between the simulation environment and the additional tools.

The combination of Gazebo and RViz within ROS allows for the simulation of a large amount of different robotic applications and use cases. As ROS is intended as

a general robotics platform, these applications and use cases can be located within a number of different domains such as autonomous vehicles, research and education, agriculture, or space technology. For applications within our domain of interest, i.e., manufacturing, a dedicated extension to the ROS frameworks exists with the *ROS-Industrial* (ROS-I) extension. ROS-I is an open-source project that aims at extending ROS capabilities towards its usage in the context of manufacturing, industrial automation, and production settings. For this, ROS-I provides an additional framework to ROS that includes tools and software packages that enable the application of ROS in industrial scenarios. ROS-I supports a wide range of industrial robot types by providing software packages and drivers that enable communication and control of these robots via ROS. It is worth noting that ROS-I adheres to industrial standards for safety applications and thus is capable of providing a safe execution environment for experiments using these drivers. Furthermore, ROS-I provides interfaces and protocols to connect ROS with other industrial automation components such as PLCs. Also, the ability of ROS in regards to perception and manipulation tasks is extended upon in ROS-I. It provides libraries and packages for tasks such as object detection and grasping of objects, e.g., by for robotic arms. These are all relevant features in regard to realizing digital twins in manufacturing as they allow for a better integration of the digital twins's model into existing systems and can provide a more accurate simulation of the digital twin.

For the ROS-I framework, a number of different software packages that offer support for different industrial robots exists. One of which is the ABB IRB 120 robotic arm for industrial applications. The IRB 120 is a small-scale robotic arm that is commonly used in applications within industry. It is a compact robotic arm manufactured by ABB Robotics. It is about 0.7 meters high, weighs 25 kg, and is placed on a square space of 180x180 mm [243]. It is operated along six axes and has a total working area of 165 degree, which allows for flexible usage of the arm in various industrial production settings. The software package for the IRB 120 of ROS-I provides the necessary drivers and interfaces to communicate and control the ABB IRB 120 robot using ROS. It allows to send commands directly to the robot and to receive messages sent from the robot's controller. This enables control of the robot's motion and behavior and to validate the execution of the commands based on the received feedback. Also, the IRB 120 package supports visualization and simulation tools within ROS, that is for our case, Gazebo and RViz. This way, we can simulate the behavior of the ABB IRB 120 robot within a virtual environment and visualize

its behavior with RViz. A physical system of a IRB 120 is available to us during the cause of our research. The robotic arm used by us is seen in its current location in Figure 6.3. As can be seen in the background, the robotic arm is completely enclosed within a transparent casing. This serves as the safety shell (see Figure 6.2 in Section 6.2) as it prevents human personal from being in reach of the robot while it is in operation. Furthermore, the robot has a custom attachment mounted on the top joint where the hardware interface for actuators is located. This attachment is used for demonstration purposes and is not movable by itself. The robotic arm is attached to a controller device (not seen in Figure 6.2). This device is a small industrial PC running a Linux-based operating system that is real-time enabled. This hardware controller is responsible for handling all messages sent between the robot and other devices such as SCADA systems or PLCs.

Figure 6.3: Robotic arm used within the experiments as TOE.

6.3.2 Building a Flexible Testbed Infrastructure for Research

In this section we cover the details of the implementation of the technologies described in the previous section. We explain how we went forth to realize a flexible and scientific testbed environment for information security evaluations in manufacturing. Specifics of the implementation such as modification to libraries and integration of different components are given in this section as well. As mentioned in the previous section, a core feature of ROS's architecture is its modular approach. This is realized by the concept of *nodes* that represent individual components that are interacting with each other within the ROS framework. This is true for all components that are seen in Figure 6.2 with the exception of the safety shell. As mentioned above, the safety shell is a transparent encasing for the IRB 120 robotic arm. However, the safety routines that are implemented in ROS-I according to accepted safety standards are still part of the overall framework. Also, the hardware devices, i.e., the IRB 120 robotic arm its corresponding hardware controller, are integrated into the framework and are represented in it as a separate node. In the context of ROS, a node in general is an executable file or software construct that is executed by the ROS framework when required [241, 242]. Nodes communicate with other nodes inside the framework meaning they are able to send and receive messages. This is achieved by the aforementioned publish-subscribe mechanisms. In the terminology of ROS, a node can subscribe to *topics* provided by other nodes. Those nodes post messages relevant for the topic, which are then delivered to other nodes within the framework. Also, each node is capable of data collection allowing for a straightforward method to store and extract data from the testbed.

A core feature of our testbed's design is the ability to switch between the digital and the physical twin meaning to adjust the data streams from the observed component within the testbed. We implemented a simulation model of the digital twin as described above using Gazebo and RViz [55]. For the virtual controller, we developed a separate software component that is realized as a node. As programming language for all nodes referred to in this section, we used C++, which is one of the programming languages supported by ROS. The other major programming language supported by ROS is Python. We choose C++ over Python as it is better suited for usage with industrial equipment in general. The reasons for this are that C++ provides more extensive support for low-level programming and direct hardware interaction. This is an important point to consider for industrial applications as they often require interfacing with specialized hardware or external devices. Python is

also capable of interacting with hardware through dedicated libraries but may not always offer the same degree of control and efficiency as C++. Also, C++ is a statically typed and compiled language, whereas Python is an interpreted language. This generally results in faster execution times for C++ programs when compared to Python scripts. In industrial testbeds, performance needs to considered by us as this enables better handling of real-time or close to real-time data streams [21]. Our virtual controller is, thus, capable of interfacing also with the physical twin without using the hardware controller shipped with the IRB 120 robotic arm. The change between controllers is performed by the ROS framework by using different *launch configurations*. Launch configurations are files used by ROS to define and manage the launch of multiple ROS nodes with associated parameters. Launch files specify which ROS nodes to launch and provide the necessary information to start them properly. These parameters define configurable values that define the behavior of individual ROS nodes, for example, thresholds or file paths. These parameter values can be passed to the respective nodes during startup. Furthermore, launch files can reassign topics the nodes provide and are subsribed to. Using launch files enables us to configure our testbed more flexible as they allow for customization without changing the source code of ROS and other tools in the ROS ecosystem.

Before we discuss the actual execution of the attack scenarios in the next section (see Section 6.4), we describe the steps taken by us in order to implement each of these scenarios. We provide a generalized approach to attack scenario implementation at this point. This generic approach that can be used to implement a broad range of attack scenarios as outlined in Section 4.2.1.1. However, each attack scenario may still need to consider some specifics for its individual scenario. Examples for such specifics are discussed in Section 6.2.1.1 and Section 6.2.1.2, their concrete realization in the following section. For now, we provide the generic approach that is used by us in order to built and simulate new attack scenarios within our testbed. For better readability, we refer to the physical device represented by the digital twin as target of evaluation (TOE). In order to conduct experiments and to test the TOE with our testbed, we require a detailed model of the TOE. This model should capture the behavior, functionality, and potential vulnerabilities of the TOE and offer support for a medium-fidelity data representation. Creating this model for the TOE is key for meaningful information security evaluations within manufacturing as discussed above [28, 42, 156]. Therefore, additional effort in creating a comprehensive TOE model is a reasonable investment as it allows to perform more

detailed testing and analysis. For our TOE, that is the ABB IRB 120 robotic arm with its controller, we use the existing simulation models that can be executed with the ROS-I extension to the ROS framework. Once the TOE model is available, the next step for us is to build an environmental model that represents the manufacturing environment in which the TOE operates. This environmental model serves as the context for the TOE and provides a low-fidelity yet realistic simulation of the surrounding conditions (realistic here meaning that all laws of physics are adhered to and all environmental factors are accounted for). In our attack scenarios, the environmental model includes the manufacturing environment that consists of three production lines (see Section 5.3). The TOE is represented by a single node of one production line, e.g., the first station in Production Line 1 (cf. also Figure 5.4). By incorporating the environmental model, the evaluation process can better account for the specific conditions and challenges that the TOE may encounter in its operational setting.

These environmental conditions do not only consist of such factors as weather or physical boundaries. They also encompass human behavior. In particular, such behavior is considered by us that originates from an adversary and is intended to be harmful to the TOE or the environmental model. In order to evaluate the security of the TOE, it is essential to simulate such attacker behavior. We accomplished this in our testbed in one of two ways: either by replaying recorded attack traffic (that is, historical data) or by actively controlling and directing the actions of the attacker. The first case is realized by feeding back PCAP files into the simulation that were recorded previously during penetration testing of the TOE. The second case is realized via attacker scripts that specify the individual steps of an attack vector. Virtual tests are then performed by running the respective launch file configuration. These tests involve running simulations with attack scenarios in order to evaluate the behavior and responses of the TOE as well as its interactions with the environment and the attacker. Subsequently, physical tests with the actual ABB IRB 120 robotic arm are conducted by us by executing the respective launch file configuration. Before introducing attackers, an initial functional test is performed to ensure that the TOE operates correctly without any intentional adversarial actions. This baseline performance test helps establish the functionality and operational capabilities of the TOE in its intended environment with an active adversary present. The final step involves drawing conclusions from the experiment based on the collected data and measurements. The gathered information is analyzed to evaluate the TOE's perfor-

mance, identify vulnerabilities, and determine the attack's success with its impact. These conclusions can lead to us making adjustments in the experimental design or the refinement of the TOE model. Thereby, we can improve the overall simulation of the attack scenario and also potentially in the testbeds implementation. By following this systematic, step-by-step approach, we can effectively evaluate the behavior, security, and performance of the TOE within a manufacturing environment.

Note that for some scenarios is can be reasonable to adjust the order given above. For example, sometimes a model for a TOE is not immediately available as it can take a longer time to construct that model with the desired level of accuracy. In such cases, the environmental model can be built and tested beforehand with the TOE being integrated and tested at a later stage in scenario implementation. For the remainder of this section, we provide an overview on the specific tooling and software used by us for the implementation as described above. For ROS, we use the version titled *Indigo* that was released in 2014. Indigo is primarily targeted for the Ubuntu operating system in version 14.04. This is a version of Ubuntu that offers long-term support, meaning that it is supported by the development team with updated and patches over a prolonged period of time in contrast to other versions. We choose Indigo mostly as it was a current and widely used version of ROS when we implemented the first modules and nodes of the testbed as well as it offers the best support for the simulation model for the IRB 120 we use. Consequently, we use ROS-I with Version 0.4.3 as this version supports the IRB 120 simulation model and was implemented on Indigo. For Gazebo we use Version 2.2.3 and for RViz we use Version r1.11.19. These were the most advanced versions for Indigo at the time we started the realization of the testbeds core architecture. Running the simulation and the testbed further requires a standard office PC with a dedicated, recent graphics card.

6.3.3 Initial Evaluation and Testing Procedures

We now discuss the initial evaluation of our testbed before realization of the actual attack scenarios. This initial evaluation of the testbed encompasses the verification of the testbed as outlined by Section 6.1.3. Results from validation of the testbed as well as external assessment of the testbed are discussed in Section 6.5 after the attack scenarios are executed in the following section. Verification of our testbed involves assessing its performance and functionality at different stages. As outlined by Section 6.1.3 we include different testing strategies for this. We start by unit

testing, progressing form there to integration testing, and finally incorporating system testing. Each of these testing stages requires a specific evaluation process to ensure the testbed meets the desired requirements. During unit testing we tested individual software components of the testbed in isolation from other components. We tested their core functionality and examined how the units performed within certain test cases. Those test cases are small-scale tests that cover basic functionality required for the simulation of general attack scenarios as well as fundamental functionalities of the individual units. For example, in case of the virtual controller for the robotic arm and its twin, it was tested if the corresponding ROS node can receive and properly process and interpret commands that are later relayed to the robot or its twin. Such test cases are executed by us in order to verify the units behave as expected and, if necessary, adjust their implementation and repeat the test cases until adherence to the requirements can be verified.

From there, we continue to integration testing evaluation. Here, the focus of the tests shifts from testing individual units in isolation to testing the interactions and integration between different software components. Also, integration of the robotic arm hardware is considered by our integration testing. In integration testing, we focus on testing the communication, data exchange, and collaboration between two or several components. This means to verify if the integrated components behaves as expected and, again if necessary, detecting and resolving issues related to their interoperability. For example, sending of control commands from the virtual control towards the digital twin in the Gazebo environment is one test case that helps us to verify the correct interaction between individual components. An important aspect is integration testing of common interfaces used within the testbed. Proper functioning interfaces, e.g., for industrial protocols or data exchange, are paramount to ensure the integration points between individual components are functioning correctly and that our testbed can be extended in the future by additional modules and components. With integration, we can verify further that the testbed is able to handle and recover from errors and exceptions. Furthermore, we are able to measure performance of the integrated system parts, which helps us to identify potential bottlenecks or issues that might arise when scaling up the simulation.

Testing several integrated components already develops towards testing the entire system. Here, the entire system with all components is evaluated as a whole to ensure it meets the intended requirements and performs the desired functionality. The tests performed during this stage already come close to simulating an actual

attack scenario. Our tests at this stage, however, are designed to test the core functionalities of the testbed without simulating an entire attack scenario. For example, we observed that the switch module was able to direct data flow to the digital twin and the IRB 120 robotic arm. Such test cases are executed to verify if the system can offer the functionality that is expected off it and, thus, meeting its design requirements. It allowed us to test and observed the system's performance, reliability, and usability for a prolonged period of time and to improve upon some aspects where necessary. We evaluated the testbed's ability to handle real-world scenarios and user interactions while watching the overall performance and stability of our testbed.

The next step in testbed evaluation is the implementation and execution of the attack scenarios as well as discussing the results produced by the testbed.

6.4 Attack Scenario Simulation and Results

In this section we cover the simulation of two attack scenarios. The first is Attack Scenario 1 (Sorting, see Section 4.2.2.1), the second is Attack Scenario 2 (Sawing, see Section 4.2.2.2). For each of those two scenarios, the testbed implementation described in the previous section is used. We followed the step-by-step approach we described in Section 6.3.2 for realization of the two scenarios. That means that the testbed architecture remains the same as well as the industrial component realized as the digital twin, that is, the ABB IRB 120 robotic arm. Both attack scenarios are executed on the layout of the production facility as described in Section 5.3. Thus, the simulated manufacturing environment consists of three production lines with four, three, and two workstations respectively (cf. also Figure 5.4). The simulation of the scenarios and the reference scenario is executed within Gazebo. Both scenarios are first executed and evaluated entirely within the simulation environment and the digital twin. Once this evaluation was concluded successfully, the scenario is repeated by using the physical twin. For this, different launch configurations are used in order to trigger the switch for directing the data flow between the virtual and real controller. This enables us to conduct a real-life verification of the simulation results, thus, verifying the suitability of our digital twin model. The remainder of this section is dedicated to discussion of the attack scenarios and their results. Attack Scenario 1 (Sorting) is the topic of Section 6.4.1 and Attack Scenario 2 (Sawing) is discussed in Section 6.4.2.

6.4.1 Attack Scenario 1: Sorting

Attack Scenario 1 (Sorting, see Section 4.2.2.1 for details) is the first attack scenario, which demonstrates the impact an attack targeting the availability of production equipment can have. Thematically, Attack Scenario 1 is a case study on the security of manufacturing environment in kitchen furniture manufacturing. The attacker in this scenario is an insider, specifically a disgruntled employee, whose objective is to maximize the damage caused within its target facility [193]. The attacker has already completed the initial steps of the generic attack vector model (see Section 4.2.1) including reconnaissance and acquisition of the necessary credentials for intrusion of its target. The attacker's aim is not for a prolonged threat campaign or for remaining undetected for as long as possible. Instead, they rely on their access privileges and insider knowledge of the target facility to cause damage. This is achieved mostly

by manipulating control commands without using sophisticated malware. Attack Scenario 1 is inspired by the Maroochy Shire as described in Section 2.1.3 [102, 13]. The target facility is, however, not a geographically distributed SCADA architecture; rather, the target facility is spatially confined within the perimeter of an industrial production plant. The attack is targeted at an individual tooling machine and when it is successfully executed, the attack stops the tooling machine and, as a consequence, also the entire manufacturing operation of the production line where the tooling machine is located in.

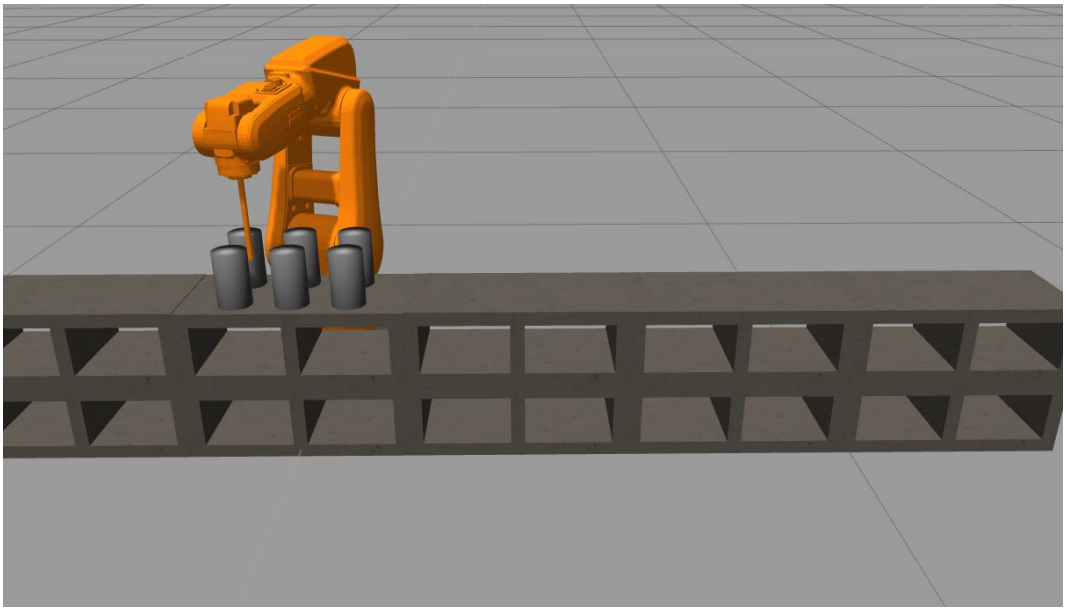
Thus, Attack Scenario 1 directly targets the availability of the assembly line. The threat to the availability of the assembly line is realized by the manipulation of a pick-and-place task by the attacker (cf. Section 6.2.1.1). This pick-and-place task here is the sorting of raw material. Sorting occurs, for example, in furniture manufacturing. As a piece of furniture typically consists of individual wooden parts, the individual parts need be sorted and prepared accordingly for processing [205, 69, 11, 9]. The attackers aim at minimizing their effort and maximizing the impact at the same time. Therefore, they attack the integrity of the communication between the control devices [21]. The attackers finally achieve their goals by direct manipulation of the sorting parameters within the MES and SCADA systems and, thus, influencing the devices interacting with the assembly process [60, 55]. Specifically, the sorting portion is targeted by the attackers. This can be represented in the reference scenario by any of the workstations in Figure 5.4 (see also Section 5.3) as pick-and-place operations are useful for almost all production flows [205].

The results of simulation within Gazebo is seen in the two images in Figure 6.4. They show one component, i.e., a robotic arm, of the assembly line positioned next to the conveyor belt. On the left-hand side, normal operation of the robot is shown, where the robot is moving between a couple of cylindrical objects. The movement is controlled by a ROS node executing a predefined program of movement instructions. On the right-hand side, the results of the successfully executed attack described by Attack Scenario 1 are shown. A manipulated parameter file is sent to the simulation via the attacking ROS node, which causes the robotic arm to alter its movement and crush into the conveyor belt. The effect for the attack is visualized by the destruction of the affected conveyor belt segments. Those segments are assumed to be non-operational after the impact caused by the arm making the belt (and also the arm) unavailable until repairs are finished.

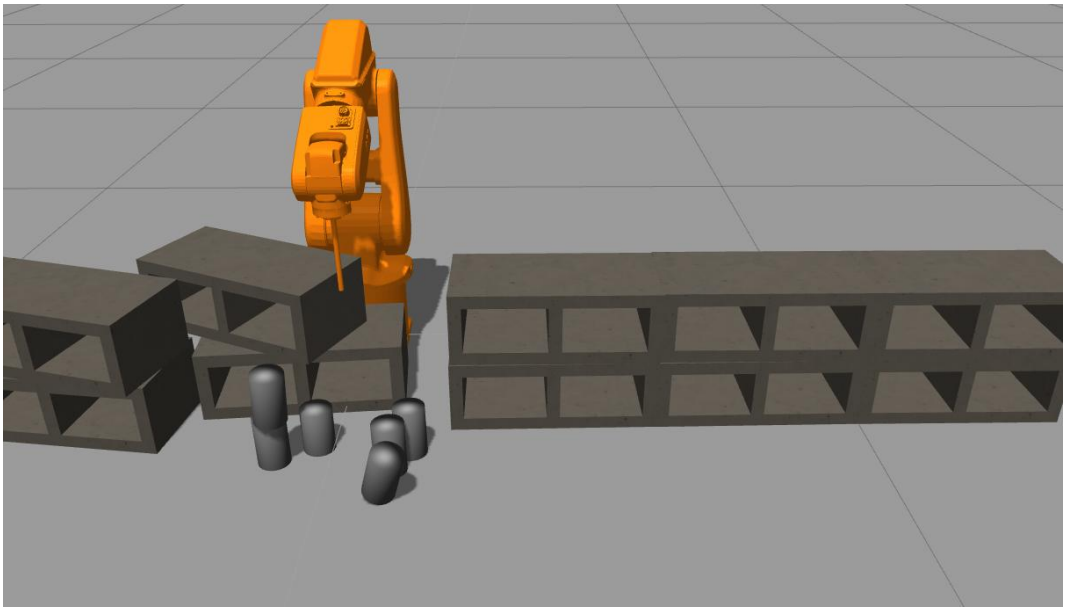
The computation of attack impact is performed within the simulation and is based on a statistical model and our prior work [60]. The impact of the attack, which is represented within the model as a form of unexpected, rapid degradation, is computed by a survivor function as described by [244]. We use this function to simulate the degradation of all components, that is, all the robotic arms within a production line. For more advanced simulations with a variety of industrial equipment, different functions, as well as parameters, can be used for each component. Also, data from the actual production line can be used to develop a more suitable statistical model. To improve visualization of the attack effects, an additional simulation for normal operation without the presence of an attacker is carried out and provided as reference for the experiments.

The simulation results are given by Figure 6.5. It shows the development of the overall system availability A_S over time t. The manufacturing environment is targeted by a cascading attack affecting each production line after a certain time with the result of shutting down the entire production until repairs are finished. For this, an arbitrary component in Production Line 1 is affected at $t = 0.25$, in Production Line 2 at $t = 0.5$, and in Production Line 3 at $t = 0.75$. The normal operation reference of A_S shows the expected degradation occurring over time. The effects of unscheduled maintenance, i.e., a successful attack, of a production line on A_S are given for each of the three lines. The added effects of the attack are displayed as well, where $A_S = 0$ between $t = [0.8, 1.0]$. At $t \approx 1.6$, repairs are finished and normal operation continues.

The simulated scenario in Attack Scenario 1 shows the results of an attack on industrial equipment with the goal of reducing the availability of the production line. In the case with no security controls present, the availability of the production line dropped from 100% to 0%, whereas the availability stayed constant with sufficient countermeasures in place [21].



(a) Simulation of Experiment 1 during normal operation.



(b) Simulation of Experiment 1 during attack scenario.

Figure 6.4: Simulation of Experiment 1 during normal operation and attack scenario.

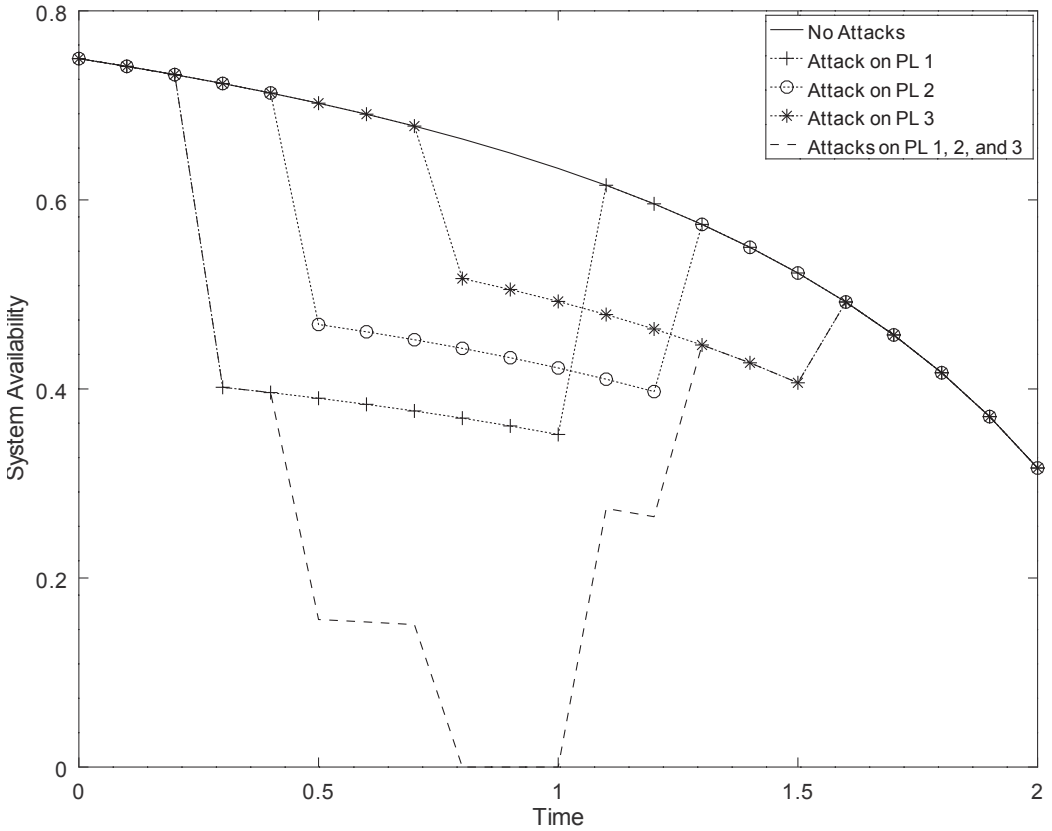


Figure 6.5: Simulation results of Attack Scenario 1. Results of non-stealthy attack simulation.

6.4.2 Attack Scenario 2: Sawing

Attack Scenario 2 (Sawing, see Section 4.2.2.2 for details) is the first attack scenario, which is inspired by the Stuxnet malware and focuses on the actions of a nation-state attacker [23, 107, 108, 19]. In such a scenario, the attacker's objective is to cause significant damage to the production process. Unlike with first attack scenario (see Section 6.4.1), the attacker aims to conduct a prolonged threat campaign while remaining undetected for as long as possible. The attacker possesses a high level of expertise and tremendous resources, enabling them to execute more sophisticated attacks on the target facility. The planning and intrusion stages, as outlined in the generic attack vector model for manufacturing (see Section 4.2.1), have already been completed before the start of this scenario. The attacker has gathered extensive knowledge about the target facility and possess malware specifically designed to achieve their goals. Attack Scenario 2 begins with the actual intrusion into the target facility, which is accomplished through standard means such as phishing, a common method of breaching manufacturing systems [25, 218]. Once the initial breach was conducted successfully, the APT, which is at the center of this scenario, infiltrates the facility and strategically positions itself in order manipulate the industrial processes of the facility.

Attack Scenario 2 is concerned with the simulation of sawing operations. This scenario is based on our previous work but same adjustments were made for better addressing our laboratory setup of the scenario discussed here [55]. Our test arm is equipped with a metal rod attached to the custom-printed end effector without any further functionality (cf. Section 6.3.2). This hardware configuration of our robotic arm is still representative of basic sawing operations while offering a more safe environment for experimentation. The basic principles between sawing with a jigsaw and the movement of our robot are transferable as they both require precise movement within a certain distance from the educt. The difference in our laboratory setup is that no material is processed in a subtractive manner. That enables us to repeat our simulation and real life operation with the robotic arm as often as desired without having the need to replace and dispose of raw materials. The robotic arm with its attached rod applies a representative saw cut by conducting a parallel movement to the object receiving the cut. The movement is executed in a fixed distance from the object in order to apply a cut of sufficient depth. If the distance

is outside certain boundaries, the cut is to deep and, thus, likely to tear the piece of wood if sufficient pressure is applied. To verify the quality of operation for the cutting process, measurement values are retrieved from the simulation environment. The values are extracted from the ROS nodes responsible for controlling the process.

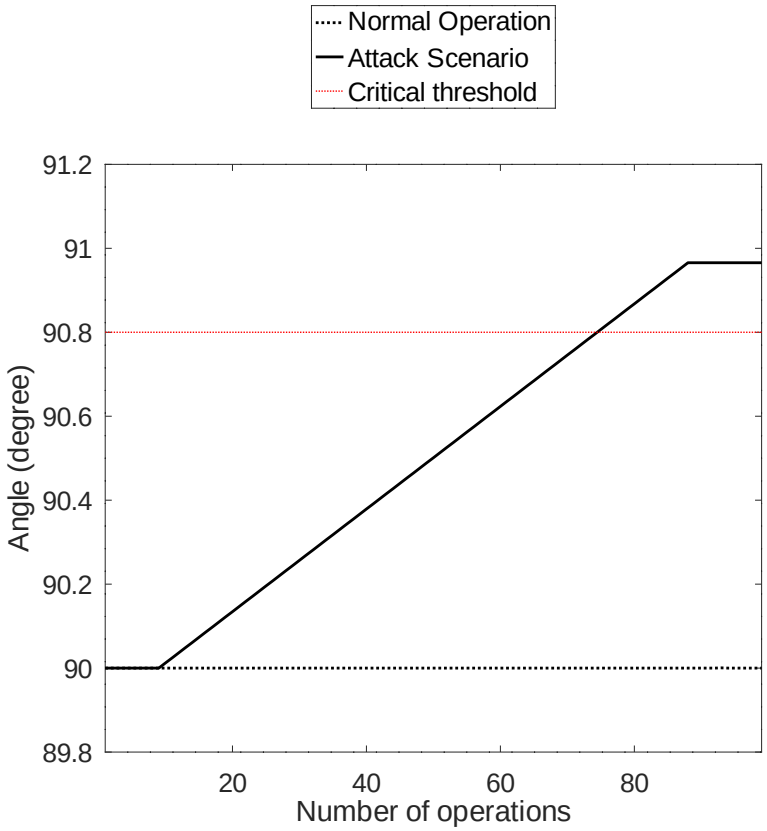


Figure 6.6: Simulation results of Attack Scenario 2 (parameter $\alpha_C = 90.8$).

The retrieved measurements are given by Figure 6.6. It shows the movement of the robotic arm over time, i.e., the number of consecutively executed sawing operations. The robot's movement is represented by the angle between the object and the rod. Ideally, this angle is close to zero during normal operation as parallel movement of the arm is the expected behavior. Normal operation is shown by the dashed line in Figure 6.6. The arm is moving with high precision keeping the angle close to 90 degree. This is realistic, as the arm used in the experiments is designed for high precision operations. The solid line in Figure 6.6 shows the development of the angle during the presence of an APT, which is implemented as a separate ROS node. The APT initially continues normal operation but starts at operation $n = 10$ to slowly

alter the movement of the arm leading to a steadily increasing angle. At $n = 75$, the APT reaches the threshold for a critical angle $\alpha_C = 90.8$ (red line), where the seam is assumed to be of insufficient quality. From $n = 90$ onward, the angle is not increased further to avoid detection. The impact of the attack can be measured by the reduction in product quality. Affected products are those products manufactured after $n = 75$ until the detection of the APT. It is assumed that those products are more likely to tear resulting in a reduced product quality and, ultimately, in a higher defect rate of manufactured goods.

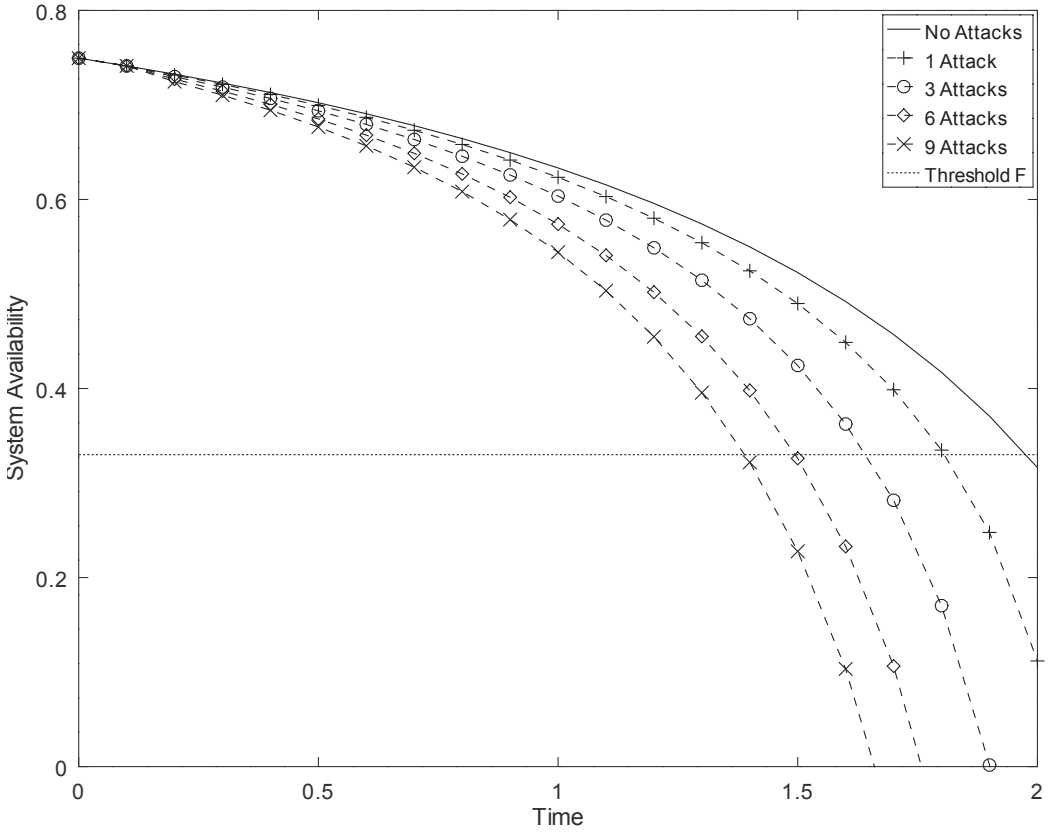


Figure 6.7: Simulation results for Attack Scenario 2 (parameter $F = 0.33$).

In contrast to the immediate effects of attacks simulated by Attack Scenario 1, the results of Attack Scenario 2 are more subtle as seen by Figure 6.7. As before, the availability of the manufacturing system, A_S, is depicted over t. The threshold for normal operation F is plotted for reference as well as the simulation of normal operation. As expected, A_S under normal operation takes the longest time until F

is reached at $t \approx 1.95$. The statistical model used in the simulation is the same as in the previous section, the APT's effect on the manufacturing process is realized by an accelerated degradation factor as compared to normal operation. This is seen in Figure 6.7 for one, three, six, and nine components affected by the attack respectively. F is reached earlier the more components are affected by the APT's sophisticated attack vector.

6.5 Discussion

In this section we discuss the results our testbed delivered. We start by evaluating our testbed based on the results produced by simulation of the attack scenarios. This validation of the testbed is in addition to the verification of testbed discussed in Section 6.3.3 and continues our evaluation methodology as discussed in Section 6.1.5.3. Validation of our testbed is conducted primarily via prototyping. Prototyping in general involves building a functional but limited version of a system or device. The prototype for our testbed is limited in the sense that not the entire coverage it provides is being used in simulation of the attack scenarios [33] (cf. also Section 6.1.2.2). Also, the selected attack scenarios reflect only a small portion of the possible attacks manufacturing equipment can face [13, 16, 18] (see Section 2.1.3). That is to be expected and providing a thorough validation of a testbed in the sense that the range of all possible scenarios is evaluated is impractical and may also not be achievable as attack vectors may remain undiscovered for some time. However, the attack scenarios selected by us do reflect on the range of possible cyber attacks in manufacturing. We selected an easily achievable attack vector that targets primarily the availability of the production facility and industrial equipment with Attack Scenario 1 (Sorting, see Section 6.4.1). On the other hand, we provided a sophisticated attack scenario that simulates an APT and subtly alters the products' quality, thus, threatening the protection goal of manufactured goods' integrity with Attack Scenario 2 (Sawing, see Section 6.4.2). These tremendously different attack scenarios allow us to examine the prototype of our testbed in different working conditions. During the prototyping evaluation stage now, our testbed is again assessed for its performance, functionality, and adherence to its requirements. This is part of an iterative refinement process based on the evaluation results, thus, ensuring that our testbed can improve from the prototyping stage.

Simulation of the attack scenarios discussed above within the testbed is the essential means of validation for our area of interest. Our testbed is used to simulate various types of attacks that the digital twin and its physical counterpart may face in real-world scenarios. By simulating such attack scenarios, the testbed evaluates the target system's vulnerabilities, potential defenses, and how the system responds to the simulated attacks. Our goal is to assess the testbed's ability to accurately replicate the behavior of the real system while providing a safe environment for security testing. For this, we executed each scenario with the digital twin first and

then executed the scenarios again with the real IRB 120 robotic arm. Our results show that the behavior of the digital twin can accurately mimic that of its physical counterpart. Our medium-fidelity digital twin aims to replicate the behavior of its physical counterpart to a reasonable extent. Several reasons can contribute to this similarity in behavior within our testbed. The model for the ABB IRB 120 digital twin we used from the ROS-I library appears to be built from using accurate and detailed models of the physical system. The model captures the key characteristics, parameters, and behaviors of the physical system. Providing an accurate model for a physical system can, thus, be influenced favorably by access to the real system as it is the case within our laboratory. Another aspect that can contribute to the similarity in results is the data integration provided in our testbed. As the digital twin consumes real-time or historical data from the physical system, our virtual controller seems to be beneficial for medium-fidelity digital twins [28, 55]. By integrating operational data into our testbed's framework, the digital twin can reflect the dynamic of its physical counterpart. This data-driven approach enhances the fidelity of the digital twin's behavior, aligning it with the actual system. Another reason can be the physics-based simulation techniques provided by the Gazebo simulator of ROS. The simulation framework of Gazebo leverages mathematical models and algorithms in order to simulate the physical behavior and interactions of the components within the manufacturing environment. By considering physical factors like forces or environmental constraints, the digital twin is inclined to replicate the behaviors observed in the physical counterpart more thoroughly. A medium-fidelity digital twin does not capture all aspects of the physical system, however, the reasons mentioned above can contribute to achieving a reasonable level of accuracy. For our attack scenarios it certainly demonstrates the feasibility of our approach. As long as attack parameters and attacker behavior remain within the same levels of granularity, our medium-fidelity digital twin of the IRB 120 can serve reasonably well for information security evaluations within manufacturing.

Furthermore, our testbed does realize the principles of scientific experimentation, which enables our testbed to be used as a powerful tool for information security research in the field of manufacturing security. To recapitulate, the principles of scientific experimentation according to [34, 27] are fidelity, safe execution of tests, measurement accuracy, and repeatability (cf. also Section 6.1.5.2). Fidelity is discussed by us above in detail in the previous paragraph, where we provided our insights in the construction of medium-fidelity digital twins for manufacturing. Safe

execution of tests is realized by us in several ways. First, we provide a safety shell for the physical device that protects personal from accidentally coming in contact with the robot that might result in injury [199] (see also Section 6.2). Second, we test all security-related interactions with the physical IRB 120 robot first within our virtual testing environment and its digital twin. Within the simulation of the attack scenarios, we are able to collect data of high resolution, i.e., our testbed does provide a high measurement accuracy. The majority of measurements can be directly extracted from the simulation environment via ROS nodes that follow the corresponding topics. Those measurements can be stored within a data historian for central collection and access to the acquired data. The measurements from the IRB 120 robotic arm are of a sufficiently high accuracy and can be taken either with the hardware controller included within the IRB 120 or with our virtual controller [243]. During verification of the testbed (see Section 6.3.3) we also verified the collected data to ensure the accuracy of the data is sufficient and that the data measurements are reasonable. This involves checks for outliers, consistency, and logical relationships within the collected data and the expected results. During verification it also showed that our test cases can be repeated as often as required by us. This is enabled by using the standardized ROS API without modifications within a constant software ecosystem (see Section 6.3.2). This reduces variability when executing the same test case repeatedly and, thus, enhancing repeatability as the results tend to stay consistent over several simulation runs. Also, the IRB 120 is placed within a controlled laboratory environment so no alterations of the arm or any environmental factors can introduce variation or cause interference. To summarize, our testbed does provide the design principles required for scientific experimentation [34, 27, 33]. It is worth noting that only about 10% of published information security testbeds for manufacturing *at best* consider all four required principles for scientific experimentation [33]. In fact, it appears that only [27] comprehensively follows the rigorously scientific approach outlined by themselves according to the survey of [33]. This makes our testbed an outstanding contribution to the field of scientifically rigorous experimentation for information security within connected manufacturing.

We further developed a baseline architecture in Section 6.2 that detailed our conceptual system architecture from Section 4.3. We started by defining the baseline architecture for the testbed including its digital twin. Our baseline architecture included the various components, interfaces, and data flows that needed to be handled by our testbed in order to provide a comprehensive testing environment for infor-

mation security evaluations within manufacturing, that is, according to our testbed design strategy (see Section 6.1). We determined the key points for interaction of the digital twin with its physical counterpart and specified the data flows that occurred between them, which is critical for the proper functioning of digital twins in our testbed [42]. These data flows can provide real-time or near-real-time data for the digital twinning framework. Also, we established methods and components for data collection from the testbed and the twins. This also involves data acquisition from the simulated devices within the testbed. The analysis of the collected is performed by us manually after execution of the respective attack scenario has commenced. For each attack scenario, there are different metrics and data points that are of interest to the analysis [60, 55]. For the purposes of producing and acquiring that data, our virtual controller with its switching capabilities proved to be an essential feature of our testbed. It wed us to enable control of the digital and physical twin, e.g., by integrating control algorithms or decision-making mechanisms to simulate the effect of different control actions on the system's behavior. It further allowed us for testing and optimizing the simulation of the attack scenarios in a virtual environment before conducting the experiments with the physical system. Ultimately, this enabled us to validate the digital twin's accuracy and performance against the real-world system as discussed above.

We already discussed verification and validation of our testbed. According to our evaluation methodology of Section 6.1.5.3 that leaves testbed accreditation open for discussion. We decided to forego formal accreditation to a standard. Rather, we choice to use a peer review process in order to achieve a degree of scientific soundness. We engaged with the scientific community early on in our research proposals and maintained ongoing collaboration within the context of publicly funded research projects. These research projects are the IUNO research project and its successor, the IUNO Insec research project [3, 245]. Both research projects are thematically located in the area of information security for manufacturing environments and industrial automation. This way, we came into contact with industry professionals and domain experts from manufacturing, which allowed us to gain valuable insights into that domain. We could conduct some field studies and data analysis at production facilities that provided experience for us that we used in testbed conception [9, 11]. We disseminated our findings further by publication and discussion of our findings on testbed construction in scientific conferences [9, 60, 21, 55]. This way, the wider scientific community evaluated your work independently. Sharing of our results

added further to the credibility of our testbed. The reviewers feedback was carefully considered by us and integrating in the final version of our work.

6.5.1 Future Work

For evaluation of our testbed we used attack scenarios that leverage a discrete manufacturing scheme (see Section 3.1.1). As outlined above, discrete manufacturing and the processes in it are a reasonable choice for simulation and evaluation. That is because discrete manufacturing processes tend to be more flexible and controllable than continuous processes. Therefore, discrete manufacturing processes are more easy to adjust and change, which is a valuable property for verification and validation (see above and also see Section 6.1.3). However, in order to cover a broader representation of the manufacturing landscape, our testbed should also include continuous processes or at least consider them at some point. As sketched in Section 5.4.1, adjustment of the simulation models appears possible with reasonable effort. Most effort probably would be needed in deriving a proper simulation model of the continuous processes. However, if tooling machines or similar industrial equipment is not used by that process, further adjustments to the overall architecture of the testbed are required (cf. also Section 7.6).

The digital twin we constructed within this chapter is a medium-fidelity digital twin of a specific industrial component, i.e., an ABB IRB 120 robotic arm [42]. The construction of a digital twin is directed at the physical component the digital twin is going to represent. Thus, the construction of a digital twin is time-consuming. For our testbed, this means that the replacement of IRB 120 by another digital twin is a resource intensive task. While we laid out and detailed our systematic approach to construction of the digital twin we used in our testbed in Section 6.3, our testbed and the associated tooling does not provide special support for the construction of digital twin. Different twins can be used within the testbed, e.g., via the usage of common interfaces such as PCAP, but their creation needs to be done within the tools provided by the ROS framework. Tooling designed for the creation and simplified adjustment of the digital twin can improve the usability of the testbed and can be considered for future work.

In Section 7.4.1 we discuss how Docker is used by us in order to provide common interfaces for integration of our work. While Docker has proven to be a useful tool for our purposes, we should make further use of it. For our testbed, which is the topic of discussion in this chapter, this can also aid in the construction of

digital twins. A Docker container does allow to encapsulate the entire environment required to run a digital twin. For digital twins in manufacturing, this includes also proprietary dependencies and legacy libraries with their specific configurations. This greatly improves the storage capabilities of the digital twin in our testbed, which is important for some use cases at the end of the physical twins product lifecycle [28]. This would further increase the broader application of our testbed as it could support more use cases.

7 Data Acquisition within Competitive Environments

In this section, we discuss the challenges associated with data acquisition in manufacturing environments, as already indicated by us in Section 1.2. The ownership and governance of data in manufacturing predominantly resides within the private sector. This makes it particularly difficult for researchers to obtain data for experimental purposes [49, 46]. However, in order to conduct meaningful experiments it is imperative to have access to realistic and representative datasets [34, 27, 32]. While there are a few datasets available that contain data from industrial sources, such as equipment and processes, these datasets are both limited in their scope and application area [49, 216, 109]. Therefore, it is beneficial for us and other researchers to have access to data from real-world manufacturing equipment and operations. This can be achieved primarily by two approaches, either by procuring manufacturing equipment for usage in a laboratory setup or by accessing real-world manufacturing environments operated by companies for data extraction. For the construction of our information security testbed, we relied on the first approach as can be seen by the IRB 120 robotic arm used in our laboratory setup [55] (see Section 6.3). For our work presented in this chapter, we rely on a similar approach by accessing data producing on a large-scale tooling machine operated within a small manufacturing environment that is used for research purposes [47]. This allowed us to work with data produced and procured from a realistic manufacturing setup. However, for most other research conducted in this area, this is not always the case (see Section 3.2). This is mostly due to the proprietary nature of manufacturing data, coupled with concerns about data privacy and the protection of trade secrets and intellectual property (IP). This intensifies the reluctance of private sector entities to share their data with researchers and especially among each others. Nonetheless, addressing this challenge is crucial for future research in the manufacturing domain. While this certainly true for information security research, this is also relevant for

other fields of research. Only by obtaining authentic and representative data sets can we conduct experiments that yield meaningful results and insights.

In the following sections, we explore our approach to address this challenge of data acquisition in manufacturing. In the context of the research project Anonymization4Optimization (A4O), we developed a method for data extraction from industrial equipment. Our approach is privacy-preserving for the data produced, which is important for acceptance among plant operators. In fact, this privacy-preserving aspect is crucial for gaining acceptance among plant operators, who are concerned about the security and privacy of their data and IP [183, 46]. We consider further requirements from tooling machine producers and from the users in order to develop a suitable architecture and privacy model. These requirements encompass considerations such as data sensitivity, data governance, and specifics of the manufacturing domain. By incorporating these inputs, we devised a suitable architecture and privacy model that addresses the unique demands of the manufacturing environment. An essential aspect of our approach is the selection of a suitable privacy model, which ensures that the extracted data is adequately protected. We elaborate on our considerations for choosing an appropriate privacy model. For this, we discuss selection of a suitable privacy model in general and in particular for industrial environments. To evaluate our approach, we have collected different data sets from the tooling machine we could access. Each data set represents a real-world manufacturing scenario where goods are processed by the tooling machine. Through analysis of the data and experimentation with the data, we illustrate how our approach achieves the desired level of privacy without compromising the utility and, thus, the value provided by the extracted data for their specific purpose. This evaluation encompasses the configuration of the privacy model and its underlying algorithms in order to strike a balance that maintains high privacy levels while enabling meaningful analysis at the same time.

By addressing the challenge of data acquisition in manufacturing through our approach, we aim to enable manufacturing operators to leverage their valuable data for a variety of use cases ranging from information security to general optimization. Our work not only focuses on developing a privacy-preserving architecture but also emphasizes the practical implementation and evaluation of this approach in real-world manufacturing scenarios. Through this comprehensive analysis, we contribute to the existing state of the art on privacy-preserving data acquisition in manufacturing and enable a variety of possible future research within this field.

7.1 Privacy-Preserving Application Design in Manufacturing

In this section, we discuss our approach for the design of a privacy-preserving application framework that can be used in the manufacturing domain. Although some work exists on the design of privacy controls, this work is limited to specific legislation such as the GDPR in Europe [246]. Also, dedicated design guidelines for the contexts where IoT devices are generally used are not available (see Section 3.2.1 for an overview on those contexts). Therefore, we adopt a design approach that is reminiscent to our approach for testbed design as discussed in Section 6.1. We also incorporate processes that are related to privacy-preserving data publishing as described by the privacy pipeline in Section 2.3.3 [178, 130, 155]. Thus, the presented design approach here builds on established processes and design guidelines and is applicable to the domain of privacy-preserving data publishing. It is worth noting that other, more general design approaches are discussed in literature, for example the methodology proposed by [179]. However, these approaches lack either the integration of anonymization and privacy models or are high-level descriptions not directly applicable to the domain of connected manufacturing. To conclude, a holistic design approach towards establishing privacy-preserving measures for applications within connected manufacturing appears, at least from our perspective, not to exist currently.

We now discuss our approach to the design of a privacy control to be used in connected manufacturing. The approach is aimed to implement the procedure seen in Figure 2.7, where the approach can be interpreted to reflect at least partially upon the task that may be expected of a privacy engineer [130]. First, we need to define the purpose of the privacy control [33]. The purpose of a privacy control refers to the intended goal for that privacy control and what the control is supposed to achieve for privacy protection. This is a clear definition of the reason why a privacy control is implemented and what it is supposed to accomplish. The purpose can vary depending on the specific manufacturing environment and use case the privacy control is being designed for [119, 46]. This can evolve raising requirements from stakeholders or following best practice approaches within the target industry. For example, in manufacturing, this can involve means of data collection and storage [144, 145] or considering the specifics of IIoT applications [246]. This step resembles creating or updating the utility and privacy policies [130] (cf. Section 2.3.3) and discussed by us in more detail in subsequently following Section 7.1.1.

Next, within the design of any privacy control, the data that requires protection by that privacy control needs to be identified and assessed [178]. For this, the types of data that are expected to be collected and processed by the application that requires privacy-preserving measures. For data that is related to individuals, this usually involves to identify the specific elements within the data set that contain personally identifiable information (PII, see Section 2.3.1) such as names, addresses, or phone numbers. As discussed in Section 3.2.1, such data is out of scope for our use cases as our focus is on the data produced by machines. Those data, even though it does not contain any PII, still can contain sensitive information that is related to the intellectual property of a company [46, 47]. Therefore, it is paramount to assess the dataset in order to determine the sensitivity of the information it contains in regards to company IP or any other IP. The potential privacy risks that can originate from such data need to be assessed and understood properly for the development of a suitable privacy control. This involves understanding the type of data, the potential information that can leak from that data, and the potential impact if the data is compromised. In the case of data produced from machines, this requires the expertise of domain experts. These domain experts work with the data curator and the privacy engineer in order to address those privacy concerns. We discuss the data landscape in manufacturing in general as well as the specifics for our use case further in Section 7.2.

The next step within the realization of our privacy control is the definition of suitable privacy goals. These privacy goals are not to be confused with the design goals or the intended purpose of the control. The privacy goals are related to the other goals and are influenced by them but constitute their own set of goals with a specific intention behind them. The intention of the privacy or anonymization goals is rather to determine the level of privacy protection needed for the privacy-preserving application. Now this is influenced by the privacy policy, which contains all relevant requirements, and aims at defining the specific privacy techniques and models to be applied to the data set. The decision by the privacy engineer could be to only employ basic privacy-preserving measures such de-identification of the PII from a data set when re-identification seems highly unlikely. However, for most use cases related to processing huge amounts of data from manufacturing environments, more sophisticated privacy measures are required [144, 145, 146]. The decision for employing more advanced privacy measures has a tremendous impact on the specific anonymization techniques and privacy models to employ, which is the next step in the design of

the privacy control. Each privacy control requires some sort of privacy-preserving technique. In most cases, this involves the application of a privacy model [121, 155]. That means the privacy engineer selects an appropriate privacy model that aligns with the privacy goals, the data types with their associated privacy risks, and the requirements summarized by the utility and privacy policy [178]. One commonly used privacy model, which has gained traction in the field of embedded devices and manufacturing, is differential privacy [144, 145, 146] (cf. also Section 3.2.1), but other privacy models like k-anonymity can also be applied to data collected in manufacturing environments [134]. The selection of a suitable privacy model for our use case is discussed by Section 7.3.1.

Once the privacy model is selected, appropriate steps towards realization of the privacy control are conducted. For most data sets, this requires some initial effort in data curation and preprocessing. For data sets containing data from individuals, this obviously involves the removal of PII. For data generated from machines, similar deletion of entries form the data set can be performed, e.g., for machine labels, work plan assignments, or brand names. But not just sensitive data points can be deleted during this stage. The data set can be exempted from any unnecessary or irrelevant information contained in it. This can include, for example, duplicate records or data records where no data is contained because of idle time performed by the tooling machine. If proper data acquisition techniques are implemented, a potentially very large data set can be collected from tooling machines [46, 47]. Therefore, also such variable and parameters can be removed from the data set that serve no purpose within the use cases being studied. This way, the data sets' size can be reduced which is beneficial for storage and transmission of the data [185]. Reduction of the data set further adheres to the principle of data minimization, which is a relevant aspect in the design of any privacy control [119]. Also, data transformations can be performed an the data set, such as changing variable formats for better processing by certain libraries or tools. Thus, the preprocessing of the data set can increase the quality of the data set, which can be crucial in providing an anonymized dataset that still offers a reasonable degree of utility.

Preprocessing is performed in an automated fashion and, thus, a part of the privacy-preserving application that is being implemented. The implementation of the privacy control can involve several steps as it is required within any software development process. For our discussion here, the implementation of the chosen privacy-preserving method, that is, the privacy model, is of special interest. The

implementation of the privacy control typically involves incorporating mathematical algorithms or mechanisms [130, 47]. For example, in the case of differential privacy, a mathematical algorithm for generation of the noise that is added to the data is required. The choice for the specific algorithm to be used in correspondence with the selected privacy model can be done in a previous step, most likely straight after selection of the privacy model, but can also be postponed to a later step. Postponing the section of the algorithm can be reasonable if it is unclear during selection of the privacy model how certain algorithms implementing that privacy model perform in an industrial setting [144]. Testing these algorithms, e.g., as part of a unit test, can then be part of the implementation process (cf. also Section 6.1.3 and Section 6.3.3). Testing at this stage in development can also involve parameter tuning, which depends on the privacy model. Privacy-preserving techniques often involve one or several parameters that control the trade-off between privacy and data utility. For example, for k-anonymity, the parameter k is the only parameter regulating these trade-off, whereas the privacy model of (h, k, p)-coherence has up to three parameters [121]. These parameters may require careful tuning and testing in order to achieve the desired balance between privacy and utility. This step involves selecting appropriate parameter values and assessing the impact on privacy as well as data utility.

Finally, after implementation and initial testing of the privacy control and privacy-preserving methods, the evaluation of the privacy control takes place. For this, we adopt a similar methodology as discussed in Section 6.1.5.3. That is, applying isolation, integration testing, and system testing during the development of the privacy in order to establish a prototype version of the privacy control and its associated architecture [33]. Also, a scientific peer review process is performed [46, 47]. Preprocessing, parameter tuning, and implementation of our privacy control are discussed in Section 7.3 and in Section 7.5.1 respectively. Evaluation of the implemented privacy control is also discussed in parts in Section 7.5.1 and concluded in Section 7.6.

7.1.1 Requirements for Industrial Environments

In this section, we discuss the development of the privacy and utility policies for our privacy control. As outlined in Section 2.3.3 and by Figure 2.7, the privacy and utility policies are the basis within the privacy pipeline, an informal process that can support within the development of suitable privacy controls for specific applications [126, 132, 130]. The context for our privacy control is provided by Attack

Scenario 3 (see Section 4.2.2.3), where a collaborative data sharing environment with the presence of a malicious competitor is described [46].

The development of both, privacy and utility policy, is driven by the raising of requirements in order to provide meaningful policies for the development of the privacy control. Privacy policy and utility policy are two distinct types of policies that serve different purposes and address different aspects. The privacy policy outlines practices and procedures regarding the acquisition, usage, storage, and sharing of the collected data. This document can help in establishing trust within an organization or application as the privacy policy communicates how the organization or application handles the data it requires for operation, e.g., in order to create some sort of value-added service such as data analysis for optimization purposes. A privacy policy is not defined by a normative body and can, thus, contain different elements. We mention some of the possible elements for a privacy policy that appear relevant to our context. A privacy policy for connected manufacturing should provide details about all types of data collected from the manufacturing environments.The types of machine data collected from manufacturing processes, such as sensor readings, performance metrics, error logs, or maintenance records should be described within the privacy policy. The purposes for which the collected data is used should be specified, for example, process optimization, quality control, predictive maintenance, or anomaly detection [216, 47]. If it is required, it should be indicated whether the collected data is shared with third parties and in what manner the data is shared. For example, data could be shared with third parties such as trusted business partners, company subsidiaries, or authorities, which may be legally required by some legislation. With the exception of legal needs, data should not be shared with external parties and a control mechanisms for sharing should be specified within the privacy policy. This goes together with data retention and deletion within the application that processes the collected data [119]. In particular, the duration for which the machine data is stored and retained needs to be given by the privacy policy. This should take into account both operational needs and legal requirements. The methods and techniques for data anonymization and aggregation should also prominently explained by the privacy policy. The methods and practices employed to anonymize or aggregate machine data should be given with sufficient technical detail including any preprocessing steps such as de-identification or deletion of data points. Also, the general data security measures need to be included within the privacy policy, that is, a description of the implemented security controls for protection of the collected

machine data from unauthorized access, alteration, or disclosure. This should also include technical details such as the encryption algorithms used or the version and configuration of TLS for data transmission [140].

On the other hand, the utility policy is concerned with the usage and expectations related to the service of application. The utility policy outlines the conditions under which the application provides its intended service to the data producers. We briefly go into some of the key elements that can be included within the utility policy and can be relevant to the scenario studied by us, also, when taking future extensions to other use cases into account [46]. Some of the elements within the utility policy are reminiscent to elements from the privacy policy, that is, because some aspects can be regarded from the perspective of privacy and usability. First of the all, the service offered by the application should be described including its features, functionalities, and limitations. For example, the purpose of an optimization scheme or anomaly detection framework should include the proposed benefit as well as the limitations of the used algorithms [47, 109]. Also, the obligations the data producers have should be mentioned by the utility policy. At least, this should include the amount of data that is required to be collected from the manufacturing environment for the algorithm to perform properly. As mentioned by Section 1.2, a sound data basis is required in order to perform meaningful information security evaluations or any other type of research within connected manufacturing [49]. Thus, the consent for data collection needs to given by the plant operator as data acquisition from a manufacturing environment can involve installation of dedicated hardware and software on the company's premises. This includes information about the data collection activities including its schedule to not interrupt any time critical production operations due to overhead introduced by the data collection [21, 22, 17]. This includes control over the application and to some degree also its customization. This can mean that plant operators are able to customize the time and type of data that is collected from the manufacturing environment, e.g., by opting in or out of specific data collection activities or adjusting the frequency of data transmission. Such measures can increase the trust that is based on in the privacy-preserving application framework and the user retains some agency within the data collection. Lastly, it should be specified within the utility policy that the intellectual property of the organization stays within its ownership including any copyright claims, registered trademarks, or patents. It is important to note that the specific content of the privacy and utility policies can vary depending on a variety of factors. These

factors can include the type of manufacturing processes being performed in the target facility, its geographical location, or any applicable laws or guidelines. However, the development of both policies, privacy and utility policy, is strongly reliant on raising proper requirements.

The requirements for the developing of the privacy control form the basis for the realization of that privacy control. Thus, it is important to gather relevant and substantial requirements that can constitute a meaningful privacy and utility policy. Therefore, we rely on requirements collected directly from tooling machine users and manufacturers in regard to collaborative data sharing platforms. The requirements are raised by questionnaires sent to such companies that operate a production environment in Germany [46]. The resulting requirements are compiled by us, the data from which those requirements are derived of is from the study conducted by [183]. Parts of this study related to the various data types produced by tooling machines within connected manufacturing are discussed in more detail in Section 7.2.1. Here in this section, we focus on those parts from the data set that are relevant in developing privacy and utility policies. The questionnaire was sent primarily to SME as they are a relevant factor in industrial information security [3, 247, 245]. SME, as opposed to large companies, often lack the resources in constructing and maintaining large-scale IT and OT infrastructures by themselves. This also includes information security awareness and management of information security [248, 25, 203, 15]. Thus, SME can benefit most from the collaborative data sharing platform and its privacy-preserving framework proposed by us. Of course, larger companies are also considered by our architecture and can participate in the data collection schemes. Let us consider a more detailed example that illustrates the benefits of collaborative data sharing schemes within the context of SME. Consider the manufacturing of kitchens as we already discussed to some degree in Chapter 5 and in our previous work [9]. The European furniture industry is mainly composed of SME that operate within their local markets [249]. Some of these SME may suffer from an increased pressure to participate within the global market and selling their products further abroad. In order to stay competitive within the present globalized economy, the adoption or expansion of new business models may be required by some of the SME (cf. also Section 1.1 and Section 2.1.1). For example, the increasing customer desire for further customization of furniture design can be such a new business model [1, 11]. As mentioned previously, SME tend to lack resources available to larger enterprises, thus, SME may consider pooling of resources among each

other more easily [250, 251]. The work of [250, 251] showed how this can be achieved within a local market with several participating companies, i.e., Italian SME in the furniture production industry.

We now discuss the requirements that are raised by us and that can contribute to the scheme as described above. Our focus is on providing a privacy-preserving application framework for data acquisition and subsequent data analysis that is cloud-enabled and secure. The requirements are formulated based on the needs reported to us by the study conducted by [183] that is included within our work [46, 47]. The particular needs of SME are well reflected within the survey data and are generalized by us for broader usability. In total, seven main requirements are identified by us that are now discussed in more detail. The first requirement derived by that method is the requirement that the data analysis conducted by the application needs to provide value-added services. For our proposed framework, we consider privacy and security the core services provided by it. However, for participating companies, an additional outcome of participation within data sharing is expected. That means that any analysis of data needs to result in some beneficial services that improve upon the current state of company. Thus, a value-added service, in our context, can mean anything that helps a participating company, for example, in improving the performance or energy consumption of tooling machines or provide a better security maturity level by the introduction of industrial anomaly detection [146, 247]. If the results of the data analysis do not provide any tangible benefits, the efforts for integration of the application do not yield a return. Therefore, potential participation partners might consider investing their resources elsewhere. This requirement is part of the utility policy as it is related to the expected service provided by the application and does not relate to privacy concerns.

The next requirement states that we should follow a vendor-agnostic approach when using large amounts of industrial data. Vendor-agnostic here refers to a solution that is designed to work with various tooling machines rather than being developed with a specific type of machine or vendor in mind. By being vendor-agnostic, we are able to collect and analyze data from diverse sources. This can be beneficial for some types of algorithms that require diverse data sets. Also, this makes our approach independent from specific vendors and allows for broad appeal and a potentially wide range of usage. In addition, we can still focus on a specific vendor should the need arise for such a manner of collaboration. For example, similar tooling machines from one vendor that are operated under comparable conditions

but in different manufacturing environments can produce data that can be combined for analysis. This way, particular use cases can be addressed by the vendors by gaining a more comprehensive view on the nature of a particular problem. For example, the detection and reduction of chattering noises, which is a frequently encountered problem by tooling machine vendors, can benefit from this [47, 252, 253, 254]. This requirement is also related to the utility policy as it describes the expectations towards the application more concrete as the previous requirement.

The next requirement reflects upon the fact that even in a vendor-agnostic framework cooperation among the data subjects may not be desired on all levels. To be more precise, the requirements states that loss of intellectual property (IP) should be prevented. In our understanding, this can mean that the privacy-preserving techniques should be designed for that particular use case in mind (see also Section 7.1). We need to consider that companies, even though they may be willing to cooperate on some levels, are still located within a competitive market environment and, thus, require protection for their IP [250, 251]. As we discuss in more detail in Section 7.2.2, IP can be derived from shop floor data as it can contain detailed information on the machine tools used as well as their configuration [46]. Furthermore, time-series data in general can constitute towards a loss of privacy. This has been demonstrated in the past by various examples and further potential for the loss of IP exists in several cases [255, 181]. As connected manufacturing is a strongly data-driven environment, attacks specific to those environments need to be considered and addressed by us. This requirement now is clearly related to the privacy policy as it directly is concerned with the establishment of privacy-preserving measures within our proposed framework.

This is supported by the next requirement we derived from the available data set. This requirement now states that the control over the data generated on the shop floor should remain with the plant operator, i.e., by the data provider or data subjects [119, 130, 155]. They are in control whether or not data leaves the company boundaries, e.g., is sent for processing to an externally hosted service. This active decision by the operator introduces an additional check and ensures that no sensitive data is inadvertently shared with external services or parties without explicit approval from the operator. This requirement also implies that it should be possible for the operator to interact with the data collection process. Ideally, the workload for this is as low as possible, which can be realized by the introduction of a central data collection point that stores the acquired data and only transmits

it when an explicit note of approval or release is issued by the plant operator or its delegate. Typically, such a requirement can be addressed within manufacturing environments by an edge device as discussed in the following [256]. As indicated, this requirement also is related to the privacy policy but does touch on the utility policy as it details the expected interactions with the application.

The next requirement, consequently, states that privacy-preserving measures should be applied at the edge of the network. That refers to the point where data is generated or used. The edge within an network is oftentimes conceptualized as a boundary between two different networks. For example, as the boundary between IT and OT networks or as the boundary between a company's internal network and the Internet. Applying the selected privacy model at the network edge, thus, means to aggregate and anonymize the collected data while still within the company's network. Thus, in accordance with the previous requirement, this requirement also supports control over the data. Additionally, IP protection can be realized by this as a privacy model is applied to the data. Overall, this requirement once more emphasizes the relevance of anonymizing data before it is shared in order to ensure that any sensitive information is adequately protected or removed. This requirement is more related to the privacy policy than the utility policy as it describes the mechanisms on how the privacy-preserving measures are applied the the collected data.

The next requirement highlights the particular properties of the manufacturing domain as it states that the technologies used within our proposed application need to support real-time applications. This is also true for anonymization techniques employed by us, which directly impacts the privacy policy and, thus, the choice of the privacy model as is discussed by us in Section 7.3.1. In order to integrate our proposed framework into plant operations, we need to be able to address the real-time constraints introduced by the industrial equipment and communication protocols [21]. This calls for a close to real-time capable solution, especially within the context of anonymization operations conducted on the company premises. The collection of sensor values, e.g., for chatter detection, from the tooling machine is possible in real-time by its PLCs [257, 211], therefore, making the anonymization operations performed on the edge a potential bottleneck within such a setup. This is a requirement that is directly related to the utility policy but also has strong implications towards the privacy policy and needs to be considered by it.

The last requirement raised by the SME participants in the survey of [183] is not directly related to security or privacy, rather, it is of a general nature. It states

that the interfaces presented to on the edge device needs to be easy-to-use by users potentially inexperienced with concepts and algorithms from anonymization. This still needs to be considered by us for our proposed solution as this directly relates to the acceptance of the solution and the potential for data extraction from the manufacturing environment. As some operations of our proposed framework need to be conducted at least partially on the premises of the company, e.g., in order to retain control over the data by the operator (see above), the GUI for the edge interface needs to be usable with minimal training. Operation of the manufacturing process should not be slowed down by allocating too much time for administration and use of the edge device. This is a requirement that only applies to the utility policy as it also relates upon the usability of our proposed framework.

The requirements discussed above give an insight into what needs to be considered when conducting research within a live, productive environment, where valuable data is extracted from. In contrast to, for example, the framework proposed by [186] (cf. also Section 3.2.1), we consider a solution that can extract data for a prolonged period of time. Thus, we elaborate in the following sections of this chapter how we address the requirements raised by us in this section. We do not provide a fully formalized and formulated privacy or utility policy within the context of this thesis, however, the requirements discussed by us are all potential input for the compilation of such documents. As indicated above, utility and privacy policies alike are also a means of fostering support while still providing guidelines for the implementation of the privacy-preserving measures. Before their realization is discussed, we first consider the data it is intended to be applied on.

7.2 Data Landscape in Connected Manufacturing

In this section we discuss the data landscape in connected manufacturing. For this, we explore the various types of data that are present within the complex domain of connected manufacturing in Section 7.2.1. Additionally, we investigate how this data is generated in Section 7.2.2, where we examine its sources. Also, we provide first insights into the relevance of privacy-preserving data acquisition for data produced within manufacturing environments. The discussion in this section serves mainly as a frame for our research and does not capture the diverse and expansive data landscape within connected manufacturing in its entirety. Some more general remarks on this topic including references with additional information can be found in Section 2.3 and Section 3.2.1.

7.2.1 Data Types in Connected Manufacturing

In this section, we give an overview on some of the data types present in connected manufacturing. The data types are those commonly produced by tooling machines. We choose to focus on data from tooling machines for several reasons. One of those reasons is their relevance as tooling machines are frequently encountered in manufacturing environments. Tooling machines directly contribute to the production process by processing raw materials and partially finished products. The data generated by tooling machines is, thus, highly relevant in understanding and examining manufacturing processes. By analyzing this data, attackers can gain insight into company secrets and the current production schedule [46]. Also, sophisticated attackers like APTs can learn about the environment and launch high impact attacks on the production environment [107, 108, 23] (see also Section 2.1.3 or Section 6.4.2). Also, for information security researchers, the data produced by tooling machines is valuable as it can help in identifying attack patterns or adversaries present within the production facility. Tooling machines are, furthermore, a reasonable choice for data acquisition as they can provide data with a high granularity. Tooling machines capture data at a basic level of the manufacturing process, thus, providing high-resolution data for specific operations as well as several parameters involved in the manufacturing process. This can be relevant for the construction of digital twins as well [38, 42]. Such levels of granularity can enable security researchers to identify minute variations, detect anomalies, and apply precise mitigation strategies [60]. Other data sources may not offer the same level of fidelity but rather may provide

a more generalized view of the manufacturing environment. Such alternative data sources are, for example, data related to the supply chain or data retrieved from the MES or ERP systems [9].

For us, the only source for data is that retrieved from tooling machines. Tooling machines, also referred to as machine tools, are mechanical and electrical devices used in manufacturing processes. They are often stationary within the plant and are tasked to perform various operations related to the manufacturing process implement within the specific plant. Examples for such tasks are shaping, cutting, drilling, or otherwise manipulating materials into specific forms or sizes. Tooling machines nowadays are typically controlled by digital computer systems and can achieve high degrees of precision and efficiency when used in production [8]. They are an important part in the production of hugely varying products and are employed in various industries such as automotive, aerospace, or electronics. One prominent example of a tooling machine used in modern manufacturing are milling machines. They are a common tooling machine used in manufacturing and perform various operations such as cutting, shaping, or drilling [47].

Tooling machines produce all types of data that can occur in manufacturing environments. We discuss some of the broader categories for that data first before discussing specifics about that data later in this section [183]. First, tooling machines generate all sorts of geometric data. These geometric data describes, for instance, the shape or physical dimensions of the manufactured product [46]. This can include mathematical objects such as coordinates in reference to a coordinate system, vectors, or curves. Also, complete surface profiles that describe the path the tool has taken and the resulting shape of the workpiece are contained within this category. As discussed in Section 7.2.2, obtaining geometrical data related to the manufactured product's geometry can be critical if obtained by attackers or competitors. The next category of data produced by tooling machines is operational data. Tooling machines produce operational data that relates to their general function and performance during operation within the manufacturing process. These operational data can include parameters such as machine tools speed, spindle rotation, tool position within a two- or three-dimensional space, or power consumption. Also, the data collected from the sensors connected to the particular machine tool can be of relevance. This is because many modern tooling machines capture real-time data during from their sensors during the manufacturing process. Sensor data can be diverse and include information about physical variables such as tempera-

ture, pressure, or vibration. While sensor data is quite specific for a certain physical phenomenon and the machine tool it is connected to, process data on the other hand is more generic. Process data takes more of a bird's eye view on the manufacturing process and provides insights into the sequence and duration of specific manufacturing operations. Process data is made available by machine tools and can include start and stop times for the machine tool, change in tools or operations performed at a specific time. With such information, production sequences can be reconstructed from the collected data, which constitutes another threat towards the IP of a company. It is worth noting, however, that there may be sources better suited for acquiring information on the processes executed within a manufacturing environments, e.g., more comprehensive scheduling data from a MES or volume-related information extracted from an ERP. Nonetheless, the possibility of at least partial reconstruction of process data from tooling machines highlights that their data is relevant but also requires protective measures.

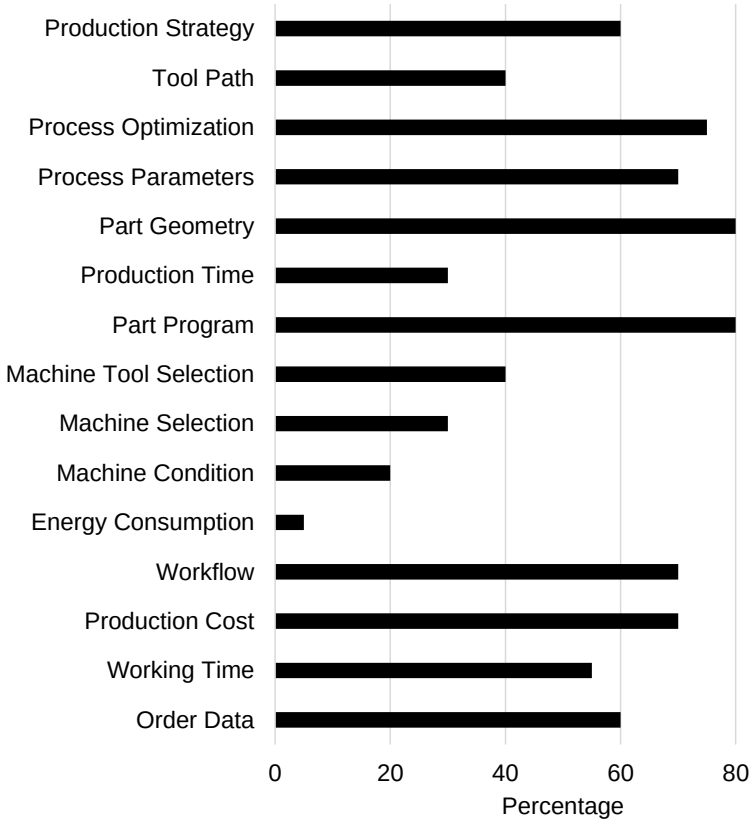


Figure 7.1: Production data and associated criticality [183].

We approach the topic of assessing the relevance of different types of data in manufacturing by examining their criticality. The term data criticality describes the assessment and determination of the relevance and of the impact data can have for an organization or within a system such as manufacturing systems. Assessing the data criticality involves categorizing and prioritizing data based on its significance and the potential consequences of its loss or in the event of unauthorized access [183]. This captures also leakage of data, e.g., from a tooling machine and their connected devices. The further discussion in this section is based on the evaluation conducted by [183]. Also, our focus is on the possible loss of IP and its implications. The survey of [183] is conducted among twenty users of machine tools as well as manufacturers of machine tools. The participants of the survey were asked to rate different types of production data in regards to their potential impact for loss of IP if the data is accessed in an unauthorized manner. This impact rating is performed on a five point numerical scale with values ranging from '1'=*'no impact'* to '5'=*'unacceptable impact'*. Figure 7.1 shows the results of the survey and gives the percentage of participants answering either with '4' (*high risk*) or '5' for a certain data type. The most impact was contributed to the data types *Part Geometry* and *Part Program* meaning the shape of the part and machine program creating this shape respectively. This illustrates the strong value put onto the actual manufactured good by machine tool users and manufactures. For those reasons, we use part geometry and part programs whenever possible in this work for illustrative purposes. Other noteworthy data types with a potentially high loss of IP are those related to the process executed on the machine tool (*Process Optimization* and *Process Parameters*), the steps taken to produce a product (*Workflow*), and its associated cost (*Production Cost*). These types of data do not contain the IP of the product but contain IP that is required to manufacture a product. For example, consider a use case from the IUNO project that is related to process parameters [70, 8]. Here, a producer of tooling machines equipped with lasers develops a business model based on the configuration parameters of their lasers. Proper configuration of the tooling machine parameters can help in reducing cutting waste for the machine tool operators when working with new and potentially expensive metals, thus, resulting in cost savings. If those data is retrieved by a competitor, this, also, constitutes a loss of IP for the machine tool operator. Also note that the data associated with energy consumption (i.e., *Energy Consumption*) is rated as the least critical by participants of the survey.

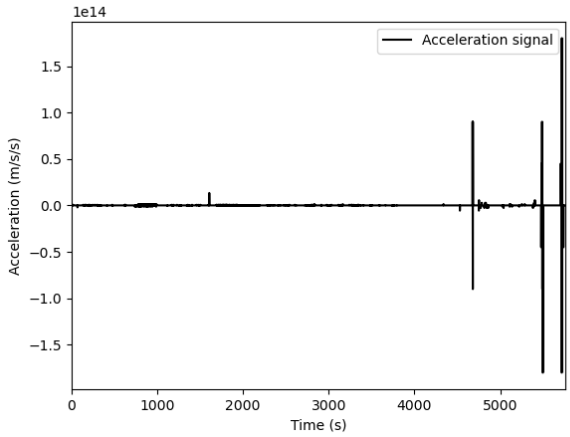
However, energy consumption can also lead to IP leakage as studies from energy and power networks have shown [255, 30].

7.2.2 Data Generated in Connected Manufacturing

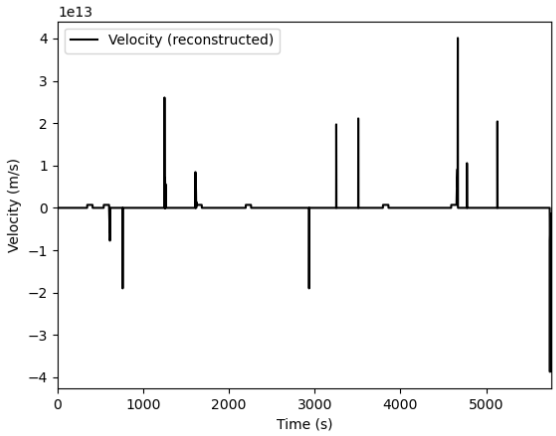
In this section, we discuss how data is generated in modern manufacturing environments and what potential applications these data possesses. We further highlight the potential risks that can arise from illegitimate access to this data and hint at possibilities to address these challenges.

In Section 2.1, we discussed the current and prospected future developments within the domain of connected manufacturing. There, we highlighted the data-driven nature of many novel applications and business models within the context of connected manufacturing [8, 3]. One technology that is enabling these new applications and business models are cyber-physical systems (CPS). CPS are frequently integrating in manufacturing processes be it in tooling machines (see previous section) or other areas of industrial automation [81, 84]. Their constantly increasing data processing capabilities offer the possibility of accessing data in quantities not available before. Harnessing this data can improve the chances of successful market participation of a company within a globalized and increasingly digital business environment [1]. However, certain challenges need to be addressed in order to best use this data. Selecting and analyzing the right data in an appropriate way is a task that requires expert knowledge [9]. Most plant operators, especially those of small- and medium-sized companies, typically lack the budget for human resources to perform such big data operations for themselves in a sufficient quality [46]. While outsourcing these data-driven tasks to an external service provider may seem like a viable solution from a monetary perspective, it is frequently met with reluctance by plant operators. This hesitance arises because many third-party services require the extraction and prolonged storage of company data outside the company's border. This aspect is critical for companies since the data derived from the shop floor contains intellectual property (IP) of the company such as proprietary product geometry [46]. The concern over maintaining control of that data and the requirement of protecting sensitive information discourages most companies in engaging with external service providers.

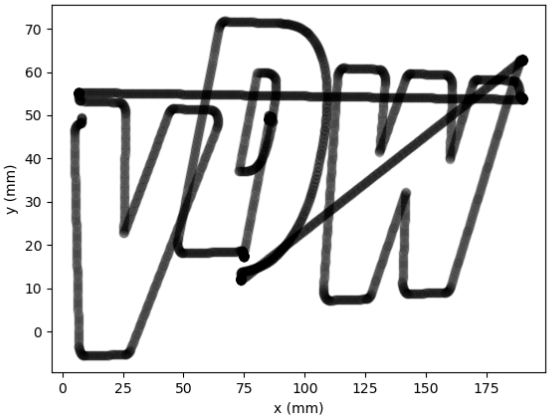
Figure 7.2 shows an example for a loss of IP that can occur from unprotected tooling machine data. The data series shows a simulated acceleration signal (see Figure 7.2a). Acceleration signals can be collected from machine tools via PLCs [46,



(a) Accelaration signal recorded from a tooling machine.



(b) Reconstructed path velocity from the accelaration signal (see Figure 7.2a).



(c) Reconstructed tool path from the velocity signal (see Figure 7.2b).

Figure 7.2: Depiction of IP loss in manufacturing.

47]. In this case, an acceleration signal represents the movement of the machine tool within a three dimensional coordinate system, i.e., the working area of the machine tool. By integrating this signal, an attacker is able to reconstruct the tool velocity (see Figure 7.2b). By integrating the tool velocity again, the path taken by the machine tool can then be reconstructed (see Figure 7.2c). As can be seen in Figure 7.2c, the letters *VDW* are visible. These three letters show the path the machine tool has taken during operation. Note that the reconstructed letters are offset by some margin from its original position as the original design file specifies horizontally aligned characters (cf. Figure 7.11 for reference to the original design file specification). This is due to numerical properties of the implementation of the integration function and the offset of an unknown constant C introduced by integration. This effect is increased as integration is performed twice on the captured data set from the acceleration sensor. The reconstructed path, however, still corresponds to the geometry of the workpiece in a way that makes it easily comprehensible to humans. This, in turn, makes the reconstruction of the design file for the workpiece possible, which in turn results in a loss of IP if that data is accessed by unauthorized entities.

It is, however, also possible to address these challenges within connected manufacturing. Edge-computing paradigms are enabling technologies to perform computations at the edge of a local network [256]. In our case, this edge represents the company boundary, from where access to an externally hosted service is possible. In contrast, CPS are located in a certain distance from the network edge. For a clear distinction, we define an edge resource in a connected manufacturing environment as a dedicated resource located between industrial CPS and cloud data centers. The edge resources are still located within the local network and, consequently, within the company boundary and in control of the plant operator. For the remainder of this chapter, we go beyond the illustrative example provided in this section. We explore how a privacy preserving for connected manufacturing applications can be developed and used for enhancing the privacy of tooling machine data for data sharing and collaborative research.

7.3 Architecture for Privacy-Preserving Data Analytics in Manufacturing

In this section, we provide an overview on the proposed architecture for our privacy-preserving data analytics framework. We provide details on our reasoning behind its development and give insights into the implementation of the individual components. We continue the privacy-preserving design approach outlined in Section 7.1 with the discussion on the selection of our privacy model and corresponding privacy-preserving measures in Section 7.3.1.

7.3.1 Selection of a Suitable Privacy Model

In this section, we discuss our process for selecting a privacy model and give our reasoning for it. The choice of the privacy model is based upon the requirements raised by us and discussed in detail in Section 7.1.1. From there, our sketches for the privacy policy and the utility policy form the basis for the further realization of our privacy control. Both policies directly influence the decision of choosing a suitable privacy model.

Choosing an effective privacy model is one of the most critical aspects in the design of our privacy-preserving framework as it provides protection to sensitive information in manufacturing, that is, IP for our case [46] (see also Section 7.2.2). To ensure our privacy model meets the desired level of protection, we consider a structured approach to it [178]. Specifically, we consider the selection of a suitable privacy model from three different dimensions: attacker model, data types, and degree of perturbation allowed. A structured approach further helps us in narrowing down from the large available body of literature regarding privacy models and their application [121, 123] (see also Section 2.3.1 and Section 3.2.1). This approach is summarized in Figure 7.3. It shows the three dimensions of attacker model, data type, and perturbation within a three-dimensional coordinate system. The dimension of data type is laid out on the x-axis, attacker model on the y-axis, and perturbation on the z-axis. As can be seen in Figure 7.3, the privacy model of differential privacy is located at an intersecting point within this coordinate system. The coordinates for the intersection are the data points for time series on the y-axis, for probabilistic attack on the x-axis, and for perturbation on the z-axis. That means that differential privacy is applicable to industrial time-series data sets, offers

a guarantee against statistical attacks, and replaces (i.e., perturbates) the original data. We now discuss the reasoning that lead to the selection of differential privacy for each of the three dimensions illustrated by Figure 7.3.

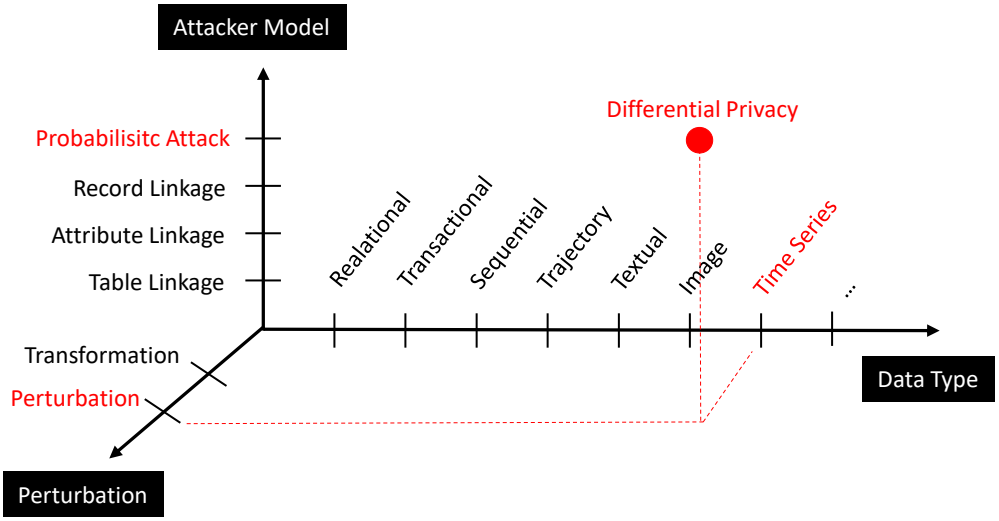


Figure 7.3: Generic approach to selection of privacy models.

We discussed generic attacker models for manufacturing in Section 4.2.1.1 and more specifically for the loss of IP in Section 4.2.2.3. In general, our privacy model needs to be capable of being applicable to a big data environment, which is what modern manufacturing environments are today [1, 21] (see Section 2.1). That means that the privacy model should be well-suited for providing aggregated, i.e., statistical, data analysis. This also reflects in parts the requirement for our privacy-preserving data analysis framework to provide value-added services as outlined in Section 7.1.1. Differential privacy has a proven track record of applications within a big data environment and is a suitable choice in that regard [130, 182, 258]. Furthermore, the big data environment we are operating in means that we need to defend such privacy attacks that try to exploit the statistical nature of the data. Such attacks are referred to as probabilistic attacks and require, in contrast to some of the other frequently discussed attacker models in literature such as k-anonymity a different approach to privacy protection [121]. Models such as k-anonymity are concerned with retrieving specific pieces of information from the data rather than an aggregated statistical view. Differential privacy is a privacy model specifically designed with this type of attacker in mind [126, 132, 131].

We discussed the data types in manufacturing in Section 3.2.1 and in Section 7.2.1. From there, we derive that our privacy model needs to be applicable to time-series data as this the type of data type most commonly produced by modern tooling machine and in manufacturing [46]. Research on privacy-preserving data publishing originally is concerned with relational or transnational databases as they are frequently encountered within the healthcare domain, where most research in anonymization and PPDP was originally conducted [121] (see also Section 2.3.1). In principle, privacy models other than differential privacy can be applied on time-series data [177]. However, differential privacy has also proven to be effective and offers an outstanding performance when applied to time series data [259, 258, 144]. This leads us to consider one of the requirements raised by us in Section 7.1.1 more closely, that is, the requirement for our privacy-preserving framework being capable of processing data in real-time or close to real-time. In [144], real-time capable privacy models for the application on data produced in manufacturing are discussed. Their work considers trajectory as well as time-series data and shows that differential privacy is well-suited to apply anonymization operations within a real-time environment, much more so than other privacy models. It is worth noting that the domain of manufacturing, the differentiation between whether or not real-time enabled data processing is required for a privacy model can be considered as an additional dimension for selecting suitable privacy models within the method we follow here [178].

The last dimension, perturbation, is concerned with whether the original data is required by the subsequent data analysis or not. If the original data is required, then our privacy model needs retain some of the original data. Different mechanisms exist for this, e.g., via suppression of information. For example, removing the final digit of the age 38 results in 3x, which covers the age range of 30-39 and, thus, resulting in a higher privacy protection of a specific person's age (see also Section 2.3.1). Privacy models using this sort of mechanisms are referred to as transformative privacy models [121]. If the original data, however, is not required, the privacy model can change or completely replace the original data. This can be reasonable, e.g., when a data analyst is only interested in the results of aggregated queries from the data set. For example, an analyst can be interested in the average age of a population in a data set and not in the individual's age. The original data set can then be replaced by a synthetic data set that gives the same (or a largely similar) response to the analyst's query. Such privacy models are then called perturbative as the original

data is perturbed or altered. As perturbative privacy models further increase the protection of IP within the anonymized data, we decide to use a privacy model that offers this while still providing strong privacy guarantees. Differential privacy provides a strong mathematical definition for privacy guarantees and appears, taking our discussion above into account, to be the most suitable choice for our interests. For the remainder of this section we now give an overview on differential privacy and its formal definitions with their implications.

We discussed differential privacy in detail within Section 2.3.1.1. There, we also provided several examples that are aimed at giving an intuition on differential privacy for interested readers. Now, we discuss the formal definition of differential privacy and its mathematical background.

The definition of differential privacy is given by Equation 7.1 [126]:

$$|ln\frac{P[R(T_1 = S)]}{P[R(T_2 = S)]}| \leq \epsilon \tag{7.1}$$

T_1 and T_2 are data sets differing in exactly one record. R is a random function (depending on the algorithm used) with S being a subspace of possible results of R. The parameter ϵ is the adjustable value in the privacy model of differential privacy allowing for configuration. The left side of Equation 7.1 quantifies the difference between two (almost identical) data sets T_1 and T_2. This difference is expressed by ϵ that bounds the level of noise added to T_1 and T_2 by R. A small ϵ means a high value of privacy as the original data is strongly perturbated by R. In contrast, a large ϵ implies high utilization of the data by the data analyst. The value of ϵ is dependent on the factors of privacy needed by the data curator and the intended purpose of usage [119]. It is worth noting that differential privacy is a perturbative privacy model, meaning that the original data is modified [178].

The definition given by Equation 7.1 focuses on the parameter ϵ. In its original definition, differential privacy also introduces an additional parameter δ [131]. This parameter δ weakens the privacy guarantee of differential privacy by introducing a probability where a privacy loss may occur. The definition of Differential Privacy (DP) in Equation 7.1 guarantees that the absolute value of the privacy loss is bounded by ϵ. With the enhanced privacy model given by Equation 7.2, this absolute value is still bounded by the parameter ϵ, however, with a probability of at least $1 - \delta$.

$$P[R(T_1 = S)] \leq P[R(T_2 = S)]^{\epsilon} + \delta \tag{7.2}$$

Using the parameter δ can be useful for experimentation or in scenarios where a certain amount of privacy loss is acceptable. Initially, we started by working with the definition given in Equation 7.1. In our case, we then observed reasonable results by enforcing the absolute privacy that can be guaranteed by DP with a probability of 100%, i.e., setting $\delta = 0$ (see Section 7.5.1.3). That is why we are not required to relax those guarantees at any point in our experimentation. We, thus, continue with differential privacy meaning the definition given in Equation 7.1, i.e., setting $\delta = 0$ for the remainder of this work.

7.3.1.1 Choice of Privacy Enhancing Technology

In this section we continue our discussion from selection of a suitable privacy model to the selection of a corresponding PET. As outlined in Section 2.3.2, a PET for us is a framework for implementing a privacy model. Also, we like to stress again the point that authors of scientific literature seem to arrive at varying understandings on what constitutes a PET [135]. We focus in this section on the discussion of such PETs that appear to be promising for privacy-preserving application in manufacturing and that offer a future-oriented approach [143]. In particular, we discuss the group of PETs that are related to privacy-preserving data analytics. These PETs enable the analysis of sensitive data while preserving its privacy. These PETs include techniques like secure multi-party computation (SMPC), homomorphic encryption, and federated learning. Each of these PETs allows for computations to be performed on encrypted or distributed data without revealing the underlying information. We discuss these PETs for the remainder of this section. We introduce each PET briefly and discuss their potential usage within manufacturing and our specific use case.

The first PET suited for privacy-preserving data analytics is SMPC. SMPC is a cryptographic technique that allows multiple parties to compute a function on their inputs. Within a SMPC scheme, the function is computed jointly on the inputs of the participating entities while those inputs are kept confidential [260]. Thus, it can be used by a set of participants that require their sensitive data to be kept secret to the other participants. However, each of the participant requires the input of the other in order to collaboratively perform computations on sensitive data for some beneficial outcome. The data in SMPC is shared among the participants without disclosing it to them and still the correct output of the computation is obtained.

SMPC may not be well-suited for certain scenarios in manufacturing, which is due to the following reasons. The cryptographic computations performed for SMPC can be computationally intensive and resource-consuming. Though CPS and industrial equipment can handle more computational tasks nowadays (see Section 1.1), performing frequent cryptographic computations may be not suitable when applied to real-time operations [21, 22, 17]. Thus, in manufacturing settings where real-time or high-speed processing is a hard requirement, the overhead introduced by SMPC can be impractical, which can result in the entire privacy-preserving data analytics framework to become inefficient. This needs to be strongly considered by us as some of the algorithms that may be used in our proposed scheme may involve a large number of participating machines and organizations. The computational and communication requirements of SMPC increase with the number of participants, which can potentially lead to scalability issues. Another reason why SMPC may not be a reasonable choice for usage in our research is the assumptions imposed by SMPC on the trust among the participating entities. SMPC assumes that all participating entities, in our case the machine tool operators and the producers of those machine tools, follow the SMPC protocol without trying to circumvent it. As outlined by Section 4.2.2.3, that is not the case as the machine tool owners may not fully trust the producers of those machine tools due to a possible loss of IP [46, 47]. For the reasons outlined above, SMPC appears not to be a reasonable choice for our proposed privacy-preserving data analytics framework for collaborative data sharing in manufacturing.

The next PET suited for privacy-preserving data analytics discussed by us is homomorphic encryption. Homomorphic encryption is, like SMPC, also a cryptographic technique that enables computations to be performed on encrypted data sets without the need for decrypting it prior to processing [261]. It allows mathematical operations to be computed directly on the encrypted data, which in turn generates an encrypted result. This encrypted result can be decrypted and corresponds to the same result as if the mathematical operations are performed initially on the unencrypted data. Homomorphic encryption shares some of its properties with SMPC that also make in impractical to use within the domain of manufacturing, at least within the context of this thesis. First of all, as homomorphic encryption is also based on cryptographic primitives, it may introduce the same delay in communication as SMPC potentially and is, therefore, not suited for real-time or high-speed settings as connected manufacturing. Second, homomorphic encryption schemes

have limitations in terms of the types of computations that can be performed on the encrypted data. Complex statistical calculations deriving aggregated results from the encrypted data sets appear to be out of reach for now. This is an issue as manufacturing processes and value-added services based upon them can involve a wide range of complex computations beyond the currently supported operations by homomorphic encryption. Third, homomorphic encryption typically involves the generation and management of encryption keys. For our privacy-preserving data analytics scheme, this calls for some sort of key distribution scheme, preferably managed by a public key infrastructure (PKI) [122]. Introducing a PKI into the scheme does, however, introduce a new set of challenges that we aim to avoid by using a more suitable PET [61, 62]. Overall, homomorphic encryption appears due to its shared properties with SMPC and the limited functionality in addition with the need to introduce additional infrastructures not to be suited as a choice of PET for our purposes.

The last PET we consider for privacy-preserving data analytics is federated learning. We introduced the concept of federated learning in Section 2.3.2.1 and hinted also at its applicability within connected manufacturing settings in Section 3.2.2. We continue the discussion from there and elaborate on whether federated learning appears to be well-suited for our intended purposes. As opposed to SMPC and homomorphic encryption, federated learning does not rely on the usage of cryptographic primitives [148, 147]. That means, that the sensitive data encountered within a manufacturing environment, such as production parameters, equipment performance, or quality control metrics, are protected by privacy controls [153, 150]. Note that federated learning explicitly makes use of privacy and is not initially concerned with confidentiality, which is an attribute from information security first of all [48, 122] (see also Section 4.1.2). Furthermore, federated learning does not require for any data, encrypted or unencrypted, to be shared with a third party. The decentralized approach of federated learning preserves data privacy by keeping sensitive information within the company premises at any time, which further minimizes the risk for data breaches or unauthorized access. As outlined in Section 4.2.2.3, the different stakeholders in our perceived attack scenario are interested in keeping their IP private while profiting from the value provided by the data [46]. Federated learning enables such a collaborative framework without the need for directly sharing any raw data. Aggregation and model training on local devices directly contributes to this. This also minimizes the need for transmit-

ting large volumes of raw data to a central server or a cloud infrastructure making federated learning more feasible and efficient in manufacturing environments with limited network resources. Finally, federated learning offers support for real-time applications and edge computing, which are important paradigms that need to be considered by us (see Section 7.1.1). As stressed above, manufacturing often requires real-time connectivity, which is much more easily achieved by federated learning and its reduced communication overhead [21, 22, 17]. In addition, the model training takes place on the edge device, which possess more computational capabilities than CPS devices [46]. For those reasons, federated learning seems to provide the best overall support for intended privacy-preserving collaborative data sharing platform. Additionally, federated learning integrates well with our privacy model of differential privacy [153, 128]. Federated learning now provides some requirements on the architecture of our privacy-preserving framework [148], which are summarized by Figure 2.6 (see Section 2.3.2.1). How these requirements are being realized by our architecture is the topic of the following section.

7.3.2 Baseline Architecture

In this section we present our baseline architecture for privacy-preserving data extraction from manufacturing environments. The purpose for the baseline architecture follows the reasoning for introducing a baseline architecture given by Section 6.2. A baseline architecture is a more specific representation of our conceptual architecture given by Section 4.3. The baseline architecture includes a more detailed specification that is required for subsequent implementation. We continue, thus, in this chapter with the realization of the overall testbed concept as depicted by Figure 4.6, that is, with the components including the Edge Device downwards as seen in Figure 4.6. The other remaining components are realized and evaluated within Section 6.3 and Section 6.4 respectively. For this section, that means we discuss the baseline architecture for the Edge Device and the Cloud Infrastructure. This includes the implementation of our privacy control specified by the selected privacy model with an accompanying PET (see previous section). Therefore, the baseline architecture integrates aspects from the conceptual architecture (cf. Figure 4.6 in Section 4.3), the privacy pipeline (cf. Figure 2.7 in Section 2.3.3), and the architectural requirements of federated learning (cf. Figure 2.6 in Section 2.3.2.1).

This is summarized by our baseline architecture given by Figure 7.4 on Page 272. Consequently, we use some of the terminology from [130, 148] in order to describe

the individual components of our architecture. We start discussing the individual components from bottom to top, that is, in direction of the data flow, which is illustrated by the arrows in Figure 7.4. We begin with the data subjects, which are located on the bottom of Figure 7.4. In our scenario, the data used for data analytics is produced by tooling machines located on the shopfloor. They are the only producers of the original data. These data is captured by a PLC device, which is connected to the machine tool, receives the data from it, and processes it. Note that a PLC can maintain multiple connections to several machine tools at the same time. Also, more than one tooling machine can participate by contributing data. For better visibility, such multiple instances are omitted from Figure 7.4. The data processed by the PLC is directly captured from different sensors that are built in the tooling machine [257].

The collected sensor data is then sent via the company's OT and IT network to the data curator for further processing. The data curator is a device with enhanced computational resources that surpass those of the CPS devices on the shopfloor, like an industrial PC that is located somewhere on the company's premises. In fact, the data curator can also be a distributed application with the individual components that constitute the data curator being executed in different instances and devices. The common denominator for those components is, however, that they are still under the governance of the plant operators. Once the data sent from the data subjects is received by the data curator, the data is stored in a local database prior to further processing. This further processing is taking place within the anonymization module, where the data is sent to next. Here, pre-processing of the data as described by Section 2.3.3 is performed prior to the actual anonymization. As the anonymization should be available in real-time, the data curator should also be capable of providing high-speed communication interfaces towards the OT network [21, 17]. The control over the data and history keeping for the data is managed by the data curator. A human operator, who decides what level of anonymization is applied to different data sets, e.g., by adjusting the anonymization parameter ϵ for differential privacy can be in control at this step [130] (see Section 7.3.1). Once the data is processed by the anonymization module, it can be sent to one of the two remaining components of the data curator. The first component it can be sent to is the local model, that is, the machine learning model training within the federated learning paradigm. The training takes place on the anonymized data to add an extra layer of privacy. The

training finishes and the computed parameters for the global model are then sent to the data control module (DCM).

The DCM is a central component within our privacy-preserving data analytics framework and is responsible for executing, among other, two major tasks: secure communication and remote attestation. The first major task, secure communication, refers to generally secure communication over the public networks such as the internet. Secure communication is realized by applying best practice security controls that are state of the art. Currently, this is provided using a properly configured TLS channel. Proper configuration of TLS is important as the incorrect usage of cryptography can lead to information security incidents themselves [140]. The second major task, remote attestation, is a way to provide control over the data raised by the data subjects to the data curator [46, 47]. It is realized by a public/private pair of keys denoted as $K_u + K_r$. For each tooling machine, a pair $K_u + K_r$ is assigned by the DCM. This serves multiple purposes. First, it provides a pseudonym for each machine tool; so the data analyst can aggregate data from one machine tool without knowing its identity. Second, by generating a new pair $K_u + K_r$, a new pseudonym theoretically not linkable for the data analyst can be easily created. Finally, via a defined message, the data curator can instruct the DCM of the data analyst to delete all data assigned to the corresponding pair $K_u + K_r$. This ensures that the data producer (the owner of the data subjects) is in control of the produced data. After the anonymized data is processing by the DCM, that data leaves the company boundaries and is sent to an externally hosted service posing as the data analyst, e.g., as a cloud-based service.

The data control module consists of two components that communicate with each other: one is operated by the data curator, whereas the other part is operated by the data analyst. At the data analyst the data is, consequently, processed again but this time by the data analyst's DCM. Here, information security controls are verified in order to ensure the authenticity of the sent data. After a successful verification of the data's authenticity, the data analyst can further process the received data. That data is stored again in database located within the data analyst's infrastructure before it is further processed. From that database, the data can be sent to one of the two remaining components within the data analyst. The first of these components is the model aggregator that, in case parameters are received by the data analyst's DCM, uses them and all other parameters received from other data curators in order to derive the aggregated machine learning model. That model is then an iteratively

trained model within the federated learning paradigm [148]. That trained model can then be used for the actual data analytics that are performed within the data analytics module. Alternatively, if anonymized data streams are received, that data can be directly processed by the data analytics module. Note that it is highly dependent on the use case and the used analytics which way data is best transmitted, that is, either by federated learning in conjunction with differential privacy or just the latter. Therefore, we include the possibility for our framework to also process an anonymized data stream directly [46, 47]. In that case, analysis of the data takes place via queries issued to the data. Note that the queried data here can be collected from different companies. This way, a vendor can, for instance, analyze aggregated data from machine tools in the field. The results of the analysis are then sent to the interested parties as feedback, i.e., the machine tool operator or manufacturer (not shown in Figure 7.4).

7.3.2.1 Attack Scenario 3

In this section we describe the third attack scenario, which is used to evaluate our privacy-preserving data sharing application. This is analogous to Attack Scenario 1 (Sorting, see Section 6.2.1.1) and Attack Scenario 2 (Sawing, see Section 6.2.1.2), which were both used for the evaluation of our testbed with its digital twin (see Section 6.4). We describe our reasoning for Attack Scenario 3 and provide a description for it in this section. The evaluation of Attack Scenario 3 is conducted in Section 7.5.2.

In Attack Scenario 3, we highlight a possible use case for the collection and anonymization of industrial data, that is, tooling machine data. Attack Scenario 3 is an illustrative case study for future developments of the testbed architecture (cf. Chapter 8 for a follow-up discussion on future work). This scenario is, therefore, less concerned with information security evaluations as Attack Scenario 1 and Attack Scenario 2 but rather with providing a concrete example from the manufacturing domain that can be used by us to evaluate our collaborative data sharing architecture. The use case we describe is specific to a certain problem in manufacturing, however, other use cases that are relevant within this domain, such as predictive maintenance or condition monitoring, can also serve as a showcase for our application [46]. This is, because the basic principle of capturing tooling machine data for optimization via a dedicated algorithm can be realized within our application (see Section 7.5).

The use case we study here is chatter detection and the subsequent reduction of the chattering. Chatter detection is concerned with the detection of chatter vibrations that occurs within machine tools [206, 47]. Chatter vibrations can be caused by interaction of the machine tool with the workpiece or by the construction properties of the tooling machine. Also, further environmental causes can also influence or create chatter vibrations. Vibrations caused by these factors are a common problem within the construction of tooling machines and their examination is a relevant field of study within the field of tooling machine construction and engineering [206, 207]. That is, because chatter vibrations can influence product quality in an undesirable fashion, e.g., the product's geometry or its surface finishing can be distorted or altered unintentionally by unforeseen vibrations of the machine tool used for processing the workpiece. This can lead to an increase in cost due to frequent re-calibration of the machine tool or can also increase the rejection rate of products with insufficient quality. Chatter vibrations can be reduced by proper configuration of the tooling machine [262, 263]. For environments that produce the same or similar products, a one-time configuration of the tooling machine can mitigate chatter vibrations to a manageable degree. However, when we consider, for instance, customer-individual production [1, 11] (see Section 1.1), this is not achievable as easily. That is due to the fact that the range of possible products can change more often. Each of these changes in product then requires a period of trial-and-error to deduce a proper configuration of the machine tool. This can result in an optimization process that is labor intensive and binds resources that could be used otherwise. For the reasons stated above, different automated chatter detection methods are proposed in literature [206, 207]. These methods have in common that they require data from the tooling machine for automated derivation of potentially relieving machine configurations. The more data is available for usage with these algorithms, the better some of these methods can perform. Aggregated data from different sites can further increase the outcome of some methods. The data can be processed by an external service provider with the proper methodological knowledge. As we show in Section 7.2.2, a loss of IP is possible for such data, which can result in the need for a privacy-preserving framework is being articulated by the data producers [46] (see also Section 7.1.1).

Thus, the primary measure of success for this scenario are the privacy guarantees provided by our privacy-preserving application. More specifically, we focus on the malicious reconstruction of the manufacturing product's geometry as product or

part geometry is considered by plant operators a potential high risk for the loss of IP [183]. That means, that the evaluation of the privacy model, its algorithms, and the supporting PETs are what is to be considered by the processes laid out in our testing methodology given by Section 7.1. This covers the evaluation of the privacy control, however, in regards to its privacy policy. The utility policy also needs to be considered by us. That means, that we require some metric that measures the performance of the chatter detection algorithm, which is accomplished by the chatter detection rate.

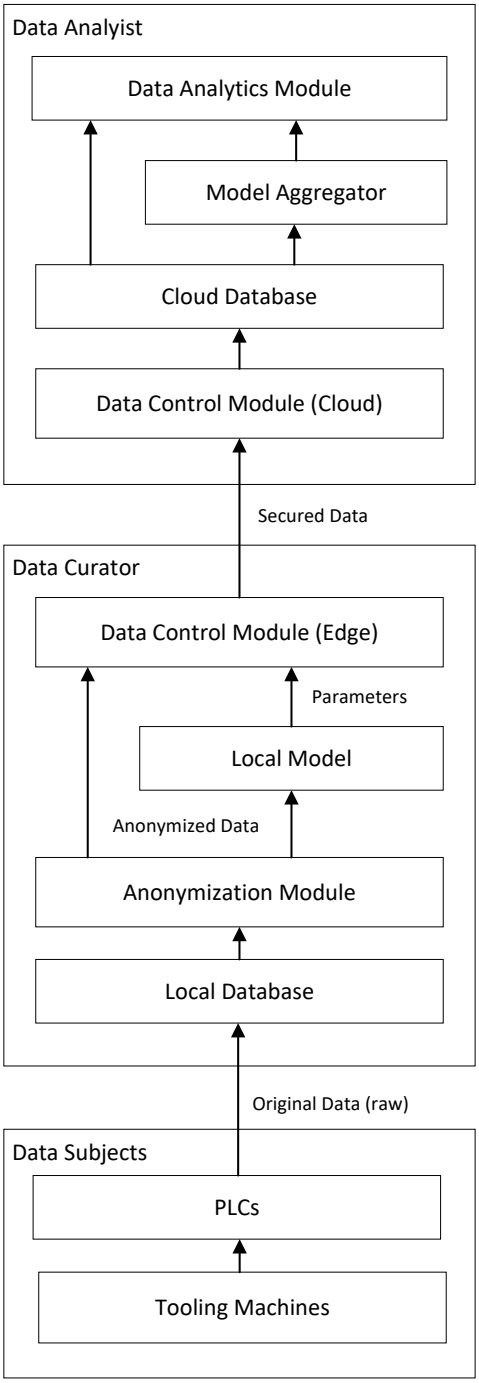


Figure 7.4: Baseline architecture for our privacy-preserving data extraction scheme.

7.4 Implementation of the Collaborative Data Sharing Platform

In this section, we discuss the implementation of our baseline architecture given by Section 7.3.2. We start by introducing the technology stack we used for implementation of the privacy-preserving framework in Section 7.4.1. Then, we give details on the implementation and realization of the individual components of our application in Section 7.4.2.

7.4.1 Technology Stack for Collaborative Data Analysis

In this section, we introduce the technologies used within the implementation of our application. Our application consists of two parts: the part of the application that is executed locally at the manufacturing site and the part that is executed remotely within a cloud infrastructure. Thus, we continue the implementation of our overall conceptual architecture (cf. Figure 4.6 and Section 4.3) that we initially discussed in Section 6.3. Both parts of the conceptual architecture, the digital twinning enabled testbed that is the topic of Chapter 6 and the privacy-preserving data sharing application of this chapter, are implemented as separate software projects. However, the integration of both, testbed and application, into a unified framework is one of our aims within this section. The integration is further discussed in Section 7.4.2 and is also the topic of Section 7.6 and Chapter 8. The technology stack for the privacy-preserving application contains hardware as well as software. Our application requires includes more dedicated hardware then our testbed but most components of the baseline architecture are still realized as pure software components that are executed on the hardware. Therefore, we start by introducing the libraries, tools, and technologies used first before elaborating on the hardware components within our implementation.

For implementation of the software components we use the same programming languages used in the implementation of the testbed components, i.e., C++ and Python (see Section 6.3.1). The choice for using these languages is in parts motivated by providing a compatible source code base, however, there are other reasons we considered during selection of the languages. C++ is well-suited for applications in manufacturing with or without a cloud infrastructure. One reason for this is that manufacturing applications often require real-time processing capabilities, as

mentioned above [21, 22, 46] (see also, for instance, Section 1.1 or Section 7.1.1). Applications in manufacturing need to be capable of handling high volumes of data and respond to a large number of events. The programming language C++ now provides low-level control of hardware resources and can be executed more efficiently than other languages. This makes C++ a suitable choice for application development within manufacturing or other real-time environments. The low-level control of hardware in C++ also facilitates better interaction of the hardware with manufacturing devices such as sensors, actuators, and controllers like PLCs. This can be useful when collecting data from those devices for subsequent analysis. Also, it may be possible to write custom pieces of code that is executed on those devices. With C++, it is possible to write low-level system code, e.g., for custom device drivers, modifications to communication protocols, or low-level optimization. Additionally, C++ provides more control for manual memory management, e.g., through pointers. That allows to realize a more precise control over memory allocation, which can be particularly useful in resource-constrained environments like manufacturing with its embedded systems. Customization of low-level manufacturing hardware such as sensors or even tooling machines is not required for our evaluation (see following sections) and is, therefore, not necessary for us during implementation. However, considerations like these are important in order to provide manufacturing systems and applications that can be extended more easily for future use [17].

While the greatest part of our application is written in C++, we also use Python for the realization of some components and modules [47]. Python offers a large collection of libraries and frameworks. The ecosystem of Python includes extensions for several topics including data analysis, data visualization, machine learning, and cloud integration. As our proposed scheme involves collecting and analyzing large volumes of data, we can use these libraries for the implementation of our framework. One of the libraries available for Python we used is Diffprivlib [264]. The Diffprivlib Python library provides tools and algorithms specifically for differential privacy applications. That includes a collection of algorithms for differential privacy [130] (see also Section 7.3.1) that can be applied to various data analysis tasks. This includes mechanisms such as Laplace noise addition and data aggregation. Diffprivlib is, as all other software components discussed in this section, available as open-source and is actively developed at the time this thesis is compiled. In general, Python integration integrates well with other programming languages as it can easily interface with software systems. This enables interoperability between different software

components of our applications and supports with the integration of cloud services, databases, and GUIs.

We further use Elasticsearch within our framework [47]. Elasticsearch is a distributed analytics engine designed to handle large volumes of data and to provide fast responses. Elasticsearch can be used for various use cases, e.g., full-text search. We employ Elasticsearch as a database management system where its tasks include data storage and indexing [180]. Our reasoning for this is that Elasticsearch is scalable and includes the ability to handle large data sets efficiently. This is relevant for us when we process and store (raw) sensor data. Furthermore, Elasticsearch provides an indexing capability that can be considered close to real-time. While it does not provide hard real-time guarantees like some manufacturing systems, the processing speed for indexing with Elasticsearch is still comparable high and suitable for our purposes. This way, we can make the stored data available quickly for subsequent operations. It is worth pointing out that Elasticsearch can be operated in a distributed fashion where it can support data aggregation and other features. This is currently not required for use case but may be an asset in an scenario with multiple tooling machines or sensors serving as the data subjects within a manufacturing environment. In conjunction with Elasticsearch we use Kibana to visualize the stored data in order to provide a GUI towards the plant operators. Kibana is part of the software library of Elasticsearch and offers all standard options for data visualization, including bar charts, line graphs, pie charts, maps, and so on. Furthermore, Kibana supports the creation of dashboards, where different data visualizations can be arranged on a single screen. Dashboards allow for a comprehensible arrangement of data towards and end user [71, 72]. This is useful within manufacturing, especially when our data curator (see Section 7.3.2) is located at or in close physical proximity to the shopfloor as it is more accessible. One additional property of Kibana that is beneficial for our purposes of applying anonymization to machine data is its increased support for the visualization of time-series data. Time-series data is the data type we are concerned with most in manufacturing (see Section 3.2.1 and Section 7.2.2).

In addition to the programming languages and the software libraries, we also use certain technologies for the realization of our application [46, 47]. The first of these technologies we discuss here is Packet Capture (PCAP). PCAP is a file format with a corresponding software library that is used for capturing, storing, and analyzing network traffic. It is commonly employed for capturing, that is, record-

ing and storing network packets. For example, the network analysis application Wireshark also supports PCAP. The PCAP library does support packet capturing in real-time as well as using historical data (cf. also our definition of digital twins given in Section 6.1.4). The recorded data is stored in PCAP files, which contain a sequential record of the captured network packets, OT protocols are also supported. PCAP files are platform-independent, which makes PCAP a method of integrating different applications with each other. Another technology used by us that is also well-suited for integration of different applications is Docker. Docker is an open-source technology for the distribution of applications and software. For this, Docker uses so-called containers that can be described as a form of operating system virtualization that encapsulates an application and its dependencies. This allows for the enclosed application to be executed consistently across different platforms and operating systems. These Docker containers are also platform-independent and can be distributed without increased effort, thus, promoting portability of applications. In our framework, Elasticsearch and Kibana are shipped as Docker containers, which allows distribution among the heterogeneous computing platforms that can be found in manufacturing systems.

Having covered aspects of data storage and application distribution so far, we now give information about the way communication in our application is realized between the different entities of the framework. To recapitulate, these entities are the data subject, the data curator, and the data analyst [130] (see Figure 7.4 in Section 7.3.2). For the communication between data subject and data curator, we employ a modern OT network protocol. This protocol is the Open Platform Communications Unified Architecture (OPC UA). OPC UA is a machine-to-machine communication protocol and also a standardized technology for industrial automation [211]. It was conceived to enable secure and reliable exchange of data and information between industrial devices and applications. The reasoning behind OPC UA is to enable data-driven applications and business models as described by us in Section 1.1 [1, 8, 3]. Thus, it promotes stronger interoperability between the different manufacturing devices and also integrates information security controls within it design, which has not been considered by prior industrial protocols [225, 21]. On the security side, OPC UA includes support for security techniques such as authentication, encryption, and access control mechanisms. Also, it is specifically designed to handle the acquisition of time-series data. Data subjects require an OPC UA server and the data curator requires a corresponding OPC UA client. Both sides, client and server, are realized by

software components that are written by us as no suitable reference implementation was available during the time we initially conducted this research.

Having covered the communication between data subject and data curator, we now turn towards the communication between the data curator and the data analyst. As data exchange format we use the JavaScript Object Notation (JSON), which is a text-based data interchange format that is also readable by humans. JSON is, as its name suggests, based on a subset of the programming language JavaScript, however, it is language-independent specification. Thus, it can also be used with our programming languages and also serves as another anchor in integration of our application with other frameworks. As JSON can be used quite straightforward, it is suited for transmitting different data types, including the data raised and shared by our application. Elasticsearch also stores data in the JSON format, which is part of the reason why Elasticsearch is agreeable with our purposes. Lastly, all of the communication between data analyst and data curator, either JSON strings or other communication, is secured by current best practices in web-based communication. These measures include identity management in the form of role-based access control (RBAC), Transport Layer Security (TLS) for the connection between Elasticsearch and Kibana instances on the analyist and curator, as well as front- and backend transmission via encrypted and server-side authenticated Hypertext Transfer Protocol Secure (HTTPS) [47]. This concludes our discussion of the different types of software used for implementing our application. We now consider the hardware that is required for the application.

The hardware that we require within our privacy-preserving data sharing framework is located at the data subject and the data curator. The data analyst is considered to be executed on a virtualized cloud infrastructure, which can be independent of the underlying hardware configuration. For the data curator, the hardware in question is one were our application framework can be executed on. That means, the device needs at least reasonable computational capabilities for applying real-time data processing and anonymization as well as sufficient storage for the recorded historical data. In our reference implementation, we use a standard industrial personal computer (IPC). An IPC refers to a computer system that is specifically designed and constructed for usage within industrial environments. That means the IPC needs to withstand potentially harsh operating conditions and environments as they may be encountered in some industrial settings. One of their most important feature distinguishing them from office PCs is their construction. An IPC typically

is constructed using components that are more durable in regards to withstanding temperature ranges, vibrations, shocks, or dust. For example, IPCs typically are constructed without a fan and use passive methods of cooling in order to avoid the increased amount of dust that can be present within a manufacturing environment. IPCs support the same range of operating systems as commercial PCs. The performance and memory of our IPC is comparable to that of a contemporary office PC.

For the data subject, the hardware can take on many different forms, potentially a large subset of the available equipment within the manufacturing landscape. It appears a daunting task to select hardware representing a data subject as proposed by our framework. As we aim to provide a data sharing application for use cases within connected manufacturing, our choice of hardware that produces that data should reflect this aspect. Therefore, we use a modern tooling machine as reference case for our evaluation that is discussed in the subsequent sections. The tooling machine we employ is a Grob G350, which is a multi-purpose tooling machine for milling operations of workpieces made from various materials. Milling here refers to a machining process that involves removing material from a workpiece using rotary cutters. The G350 is a stationary machine with a total wight of over 15 metric tons. We were able to access this tooling machine, which is installed at a model production facility at the Technical University of Munich, during the course of our research in the research project A4O (see Section 1.4) [46, 47]. The data extracted from that machine is the basis for our evaluation, which is the topic discussed within Section 7.5.

7.4.2 Realization of a Privacy-Preserving Data Sharing Application

In this section, we cover the actual implementation of our privacy-preserving data sharing application after having discussed the underlying technology stack in the previous section. We provide details on how these different technologies are integrated in order to realize the framework envisioned by us in Section 7.1. At the core of our application we leverage the privacy model of differential privacy in combination with federated learning and combine this with an edge computing framework with a connected cloud infrastructure [46, 47]. The overall architecture for our privacy-preserving data sharing and analysis framework is given by Figure 7.5 on Page 284. Figure 7.5 is reminiscent of Figure 7.4 but adds more details and specifics of the implementation based on our discussion in the previous section. Thus, it contains

more mentions of specific hardware and software. For the remainder of this section, we discuss Figure 7.5, again from bottom to top. We delve into some of the details of our implementation, conceptual decision are only touched broadly in this section. For a detailed conceptual discussion see Section 7.3.2.

On the bottom of Figure 7.5, the data subject is located, that is, the tooling machine with its connected industrial automation systems. The G350 CNC tooling machine is located at the very bottom of Figure 7.5 and is the singular data producer within our implementation. The sensor data from the G350 is collected by a standard industrial PLC that is directly connected to it. As the G350 is a stationary machine, the PLC is located in close proximity to it and possesses a wired connection to the machine. With this, we collected two sensor signals relevant to chatter analysis, that is, acceleration and noise signals [262, 263]. On the PLC, the OPC UA server is installed and executed [212]. Modern PLCs like the one we used for implementation are capable of running and executing custom software modules [211, 257]. The collected sensor data is collected from the PLC and transmitted via an OPC UA client-server architecture from the OPC UA server at the data subject to the OPC UA client at the data curators side.

The data curator is located within the middle part of Figure 7.5. In our case, the software required for data curation is installed and executed on a dedicated IPC, which is located also in close proximity to the G350 and its PLC. That is, the IPC is setup by us on the shopfloor within the manufacturing lab we were able to access. Thus, this IPC with its connection to the internet constitutes the network edge in our setup. Here, the data transmitted via the OPC UA server is received by the OPC UA client. That data is stored within the local databases, an Elasticsearch application. From here, the stored data can be directly visualized via the Kibana dashboard. This way, an initial data assessment can be conducted by the data curator. Note that the we use the term data curator to mean both, the IPC and a human operator. The human operator takes an active role in data curation when interacting with dashboards an the DCM (see below). Through interaction with the dashboard, the human operator is enabled to adjust the anonymization parameter ϵ and decides, which data is sent outside the company premises. From the local database, the data is processed by the anonymization module before further operations are applied to it. The anonymization module is implemented with Python and Diffprivlib [264]. From Diffprivlib, we use their implementation of differential privacy with the LapLace

algorithm and integrate it within our anonymization module (see Section 7.5.1.4 for more details when we discuss the evaluation of our privacy-preserving framework).

Form this point on, we now operate within our architecture exclusively with the anonymized data. The anonymized data can already be used for local chatter analysis at this point. A chatter analysis module is included on the data curator that can process and visualize initial chatter analysis. This way, the plant operator and other interested individuals can perform chatter analysis locally without the need of transmuting the data further. Note that it is also possible to directly use the raw sensor data prior to anonymization for local chatter analysis. In fact, this raw data is likely to be better suited as the results from the chatter analysis probably provide better quality. However, for our purpose, we are interested in the privacy-preserving aspects of our scheme. Therefore, local chatter analysis on sensor data that has not been anonymized prior is not implemented at this point. Later implementation of this is, however, possible and supported by our application. The same is true for the local model, which was not included during the time we had access to the G350. The local model could, if realized during our evaluation on the shopfloor, also receive the anonymized data and start the training process. The training takes place on the anonymized data, which further reduces the risks of leaking IP from the model based on the trained data [153, 128]. After training on the local model finishes, the updated parameters are sent to the local DCM. Also, the anonymized data can be sent directly from the anonymization module to there.

Within the edge computing portion of our architecture, the DCM is responsible for communication between the edge and the cloud infrastructure. With the local DCM, where the anonymized data is received and further processes, secure communication over the internet is possible. It also enables the control over the data that is sent to the cloud outside of the company's premises. Each machine tool like the G350 is in possession of its own unique pair of public and private keys. These are used to sign the data before the transmission, the public key and the signature are sent with each packet sent over the internet. This serves several purposes, one of which being to ensure that the sent data has not been tampered with during the transmission. Also, this provides a pseudonym for the machine tool so the data sent from that individual machine can be associated to it by the cloud. By generating and using a new set of public and private keys, a new pseudonym can be generated, which is theoretically not linkable to the previous set of keys. Another purpose served by this is that a deletion command signed by the corresponding private key can be sent

to the cloud's DCM. This deletion command then leads to the deletion of all data related to this pseudonym by the cloud. From a communication point of view, the DCM is the systems central point as it coordinates data flows among many of the different components within the application as well as the communication between the data curator and the data analyst.

The data analyst, which is located on the top of Figure 7.5, is implemented within a virtual cloud server infrastructure that is hosted at a site different than the site where data subject and data curator are located. That cloud computing infrastructure consists of a Debian GNU/Linux virtual server and are virtualized by employing corresponding Docker containers using volumes for data persistence. The data sent from the local DCM is received here at the data analyst by the remote DCM. The remote DCM verifies the incoming requests sent by the local DCM and processes these request. As an input, both DCM modules take a JSON based configuration file. Among others, this configuration file contains information such as internet addresses, ports, and credentials necessary to connect to the ElasticSearch instances. Furthermore, it contains information about which variables are collected by the OPC UA server/client and about their anonymization status. Figure 7.6 on Page 7.6 illustrates such a JSON configuration file and the data formatting used by us. The figure shows an excerpt from a JSON message file transmitted between local and remote DCM. The first two parameters shown are the digital signature of the message in Line 1 and the public key K_u used in Line 2. Then, the actual data transmitted is seen in Lines 6-8. In this example, the x-axis position of the machine tool (*measPos1[u1, 1]*, cf. Table 7.1), the CNC program name (*progName*, ibid.), and the recorded timestamp are contained. Figure 7.6 only gives an example for data transmitted within our architecture. In our test environment, more variables, e.g., the variables of Table 7.1 are processed and used for evaluation (see Section 7.5.1).

The data processed by the remote DCM is, similar to the data received by the OPC UA client as described above, stored within an Elasticserch instance, the remote database (see Figure 7.5). That is, if regular data is received by the remote DCM and if the verification of its signature successful. In case the remote DCM receives a deletion command, an additional timestamp verification is performed by it. This is a protective measure to prevent replay attacks. Once the signature as well as the timestamp of the deletion command are verified, all data records related to the corresponding public key is removed from the remote database. The data still within the remote database can be visualized by another dashboard. Interaction with the

data by an human operator at this point is mostly limited to viewing and analyzing the results of the algorithm. These results are produced by the remote chatter analysis module that performs its analysis on the aggregated data sets. Chatter analysis or other optimization methods can benefit from a large amount of available data making an aggregated cloud-based analysis reasonable. Cloud-based chatter analysis with aggregated data sets can hugely increase the quality of the chatter analysis and, thus, improve upon the optimization provided by the local chatter analysis module. These kind of analysis can also be performed using the aggregated model based on the parameters received from the data curator. In this case, the parameters need to be aggregated by a model aggregator before the updated remote model (not depicted by Figure 7.5) can be used [148]. Again, as the local model is not implemented in our setup used for evaluation, a model aggregator and a remote model are not implemented at this point.

Some of the software components located at the data curator and the data analyst share a similar implementation. For the most part, these are the local and remote dashboards, the local and remote databases, and also to some degree the local and remote DCM. Our application consists of two pieces of software that communicate with each other and are designed to be used together, the local and the remote parts. The local part is the data subject with the corresponding data curator, while the remote part is the data analyst. Other applications can potentially interact with those components by accessing common interfaces provided by our applications such as PCAP, OPC UA, or by using Docker containers. The implementation we presented within this section is evaluated within the following Section 7.3.2.

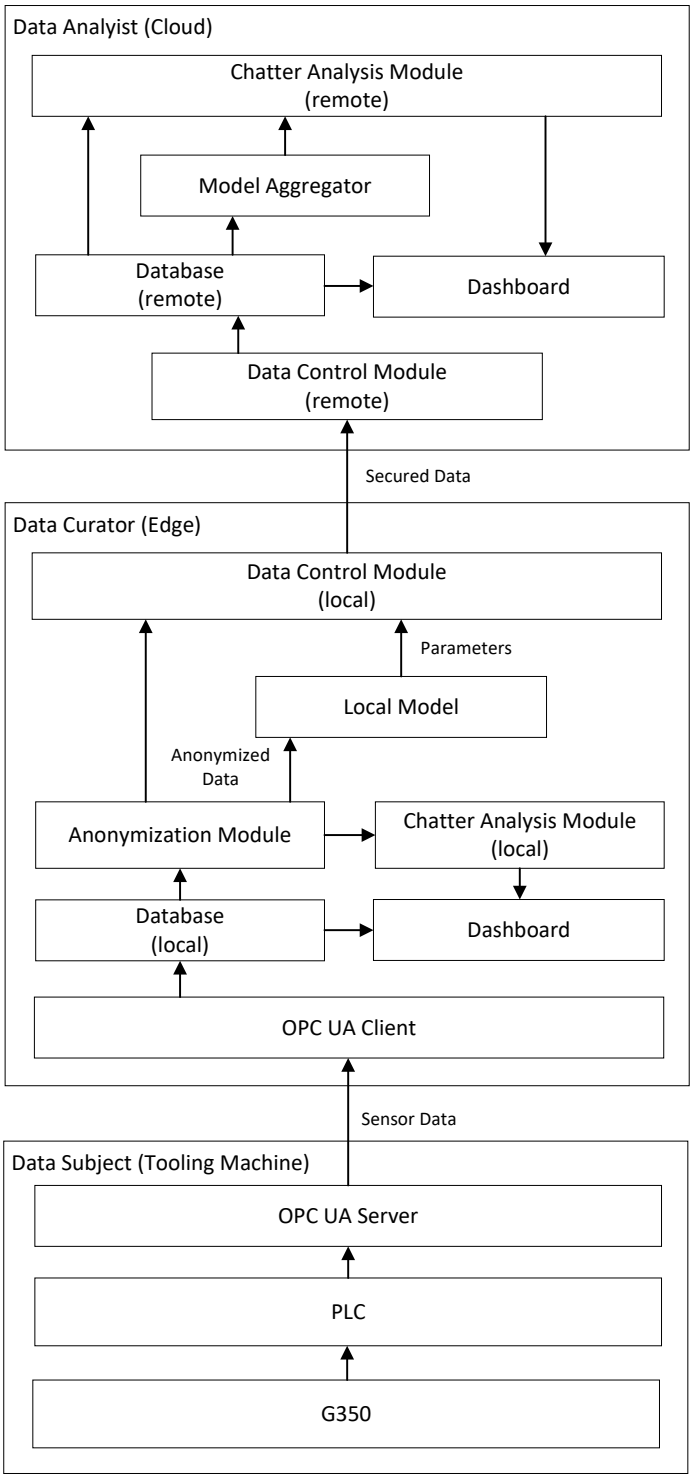


Figure 7.5: Components of the privacy-preserving data acquisition framework.

Figure 7.6: JavaScript Object Notation (JSON) data format used for communication with the Data Control Module (DCM) (both, local and remote).

```
{
"sig": "e5gtubsDNbi8Y66MIE7zYCJz4CgY3/Htv2irC1+ABgtojDyApP9fZ984
  VgpzCFw1kFC9OtDik17NMRRIPa62EA==",
"pub": "ecdsa-sha2-nistp256 AAAAE2
  VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBIgqRDtjAThk53
  PAjQx9k8DXCeUhIxestOkn3An9+tOy/gewRU9eT1grsLVxrEuMgkhHDDnp5
  QTtKWpOECRXOkQ=",
"data":
{
"measPos1[u1, 1]": 0.01383,
"progName": "_N_SN_KRAFT_NUTENSCHNITT_MPF",
"timestamp": "2020-03-12 15:07:04.248000"
}
}

```

7.5 Evaluation of the Data Sharing Platform

In this section, we discuss the evaluation of our privacy-preserving data sharing platform. As discussed in Section 7.1, we need to find suitable values for the variables used in our privacy model, that is, differential privacy (DP). The application of differential privacy to our scheme is discussed in detail by us in Section 7.5.1. Once appropriate values for differential privacy are identified, we can introduce these values into our implementation as discussed by Section 7.4. From there, we continue with the evaluation of that implementation in Section 7.5.2.

7.5.1 Application of Differential Privacy

In this section we give details on the application of our architecture as well as on the implementation and evaluation of DP in manufacturing. This is done in the context of the research project A4O (see Section 1.4). We elaborate on the implementation of differential privacy and its algorithms rather then the setup of the cloud infrastructure used as data analyst, which is the topic of the previous section. We discuss the specific data captured by us that we are applying these algorithms to (see Section 7.5.1.1) and the challenges implied by the usage of our privacy model (see Section 7.5.1.2). Then, we show how these challenges can be addressed by running several experiments on the available data sets (see Section 7.5.1.3).

7.5.1.1 Application on Tooling Machine Data

In this section we give details on how to apply differential privacy to specific machine data. As we discussed in Section 7.2.2, most data processed in manufacturing can be categorized in two types: sequential respectively time-series and trajectory data. The latter is mostly associated with the movement of workers within a plant. As our focus is analytics based on data produced by equipment (i.e., tooling machines), we investigate sequential or time-series data.

Sequential data is the data most commonly produced by tooling machines and other industrial equipment [177, 144]. Table 7.1 shows exemplary data that is typically produced by tooling machines. The data is processed by a PLC device and can be extracted from there (see Figure 7.4). The data shown in Table 7.1 represents variables observed by us within the context of the research project A4O. In the left column of Table 7.1, the name of the variable is given. The middle column

provides the corresponding data type, which matches standard data types. The meaning of the variables is given in the right column. The data shown in Table 7.1 conforms to the OPC UA standard, which is a modern standardization effort in manufacturing [211, 212].

Table 7.1: Data types collected from tooling machines (example selection) [211, 265].

Variable	Data type	Description
actSpeed	Double	Rotational speed in percentage of maximum machine speed.
actTNumber	Unsigned Word	Number of the currently active tool.
driveLoad	Double	Utilization in percentage of maximum machine capacity.
measPos1[u1, 1]	Double	Real position for the first measuring system (x-axis).
measPos1[u1, 2]	Double	Real position for the first measuring system (y-axis).
measPos1[u1, 3]	Double	Real position for the first measuring system (z-axis).
progName	String	Name of the G-Code program being executed.

Anonymization of data typically consists of a two-step process [121]. First, the dataset is de-identified meaning all variables that allow for direct identification of the target are removed. Also, variables not required for the specific analysis are de-identified. This corresponds to the principle of data minimization [119]. For the data shown in Table 7.1, de-identified variables are *progName*, *actTNumber*, and *driveLoad*. De-identification is a straightforward operation as it requires only deletion of the corresponding data columns from the dataset.

The second step is the actual anonymization of the data. For this, a privacy model is applied to the de-identified data. In our case, differential privacy is used (see Section 7.2.2). As discussed, differential privacy is a definition rather than an algorithm [131]. That means, an additional algorithm for DP is required to compute differential privacy. The algorithm ensures that the privacy guarantee for differential privacy is enforced (see Section 7.3.1). Different algorithms exist [130] as well as different implementations for industrial environments [144].

All differential privacy algorithms apply a certain level of noise to the original data provided by the data subjects (see Figure 7.4). This noise is added by statistical distributions [130]. Most commonly, Gaussian or LaPlacian distributions are

used. For our implementation, we choose a LaPlacian distribution as it is more easily parameterized (LaPlace introduces only one additional parameter to differential privacy, see Section 7.5.1.3) [126].

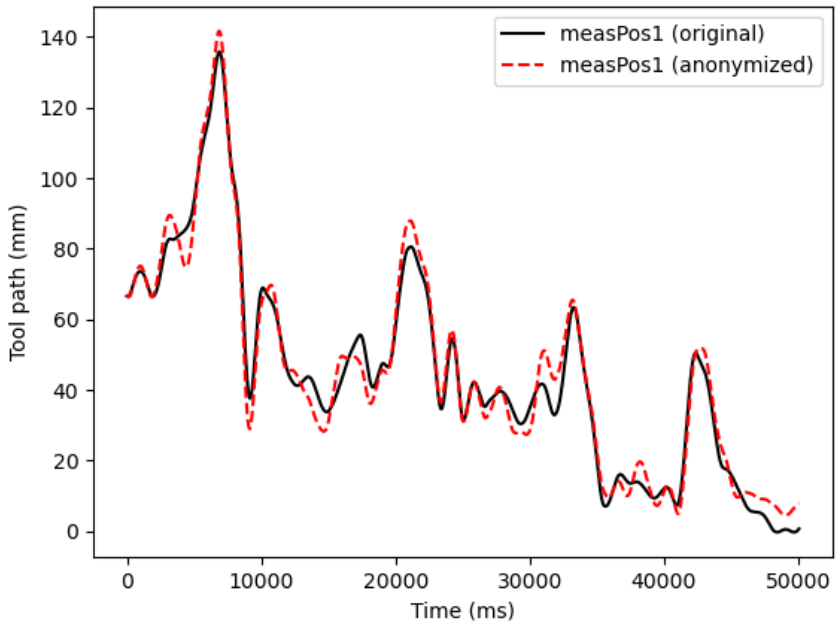


Figure 7.7: Original and anonymized datasets for the machine variable *measPos1* (Parameter values for LaPlacian DP: $\epsilon = 0.1$, $S = 10$).

Several open-source DP frameworks exist due to the increasing use of DP in recent years [266]. For our implementation, we adopted a LaPlacian DP routine from an open-source library [264]. When applying these routine to tooling machine data, the values of the data are changed in the specified way as illustrated by Figure 7.7. Here, the recorded values for the variable *measPos1[u1, 1]* (see Table 7.1) are plotted over time. The sampling interval is 200ms, the starting time is set to $t = 0$ for easier depiction [1] . The recorded data shown on the y-axis is the x-position of the monitored machine tool in reference to a three dimensional grid.

As seen in Figure 7.7, the original data (plotted as black solid line) is changed by the anonymization algorithm, a new dataset for the variable *measPos1* is generated (red dashed line). Data mining applications are then performed by the data analyst on the the anonymized dataset without knowledge of the original dataset. Depending

[1]The data set shown in Figure 7.7 originally was recorded between 11:01:10.931 and 11:29:59.966.

on the use case, a specific privacy model is used. The used amount of noise has to be sufficient for the protection of intellectual property on the one and utilization of the data on the other hand. In Section 7.5.1.3, we further discuss anonymization and selection of reasonable parameter sets within the context of tooling machine data.

7.5.1.2 Sensitivity of Privacy Parameters

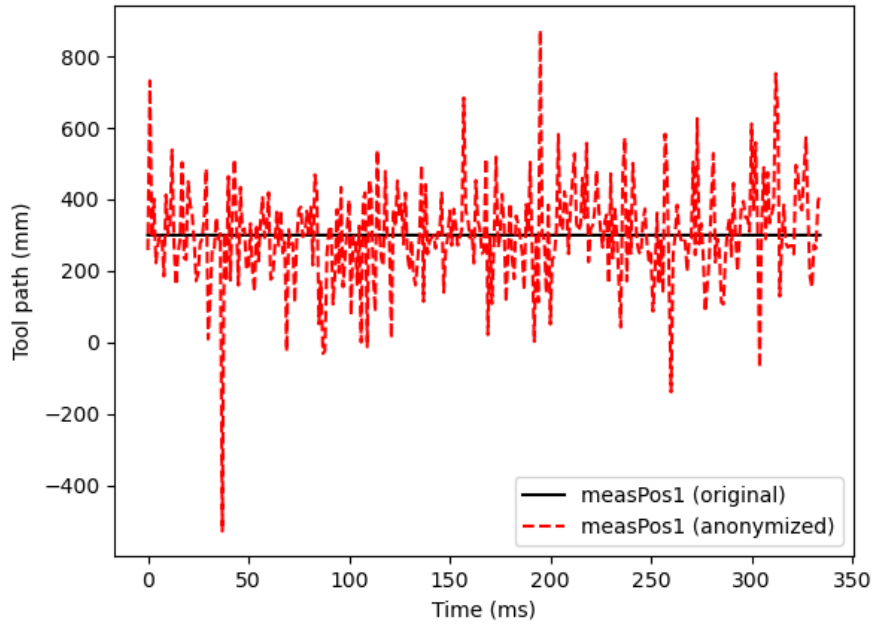
The level of perturbation from the original data to the anonymized data depends on different parameters. For DP, the parameter ϵ is always present by definition[2]. In case a LaPlacian distribution is used for the addition of noise, an additional parameter S denoting the *sensitivity* of the LaPlace distribution is required [130]. The parameter S captures the magnitude by which a single data entry in a dataset can be changed by the randomized statistical distribution.

The effect of the parameter S is illustrated by Figure 7.8. The LaPlacian DP algorithm is applied to the variable *measPos1[u1, 1]* (see Table 7.1). For better readability, we denote *measPos1[u1, 1]* as *measPos1*. The value for *measPos1* is taken from a dataset where an almost monotonous operation is performed by the machine tool. Therefore, the baseline of *measPos1* is almost constant with a value of around 300 ± 0.005. Different parameter configurations are applied to the data: for Figure 7.8a the parameters are $(\epsilon, S) = (0.1, 10)$, for Figure 7.8b they are reversed: $(\epsilon, S) = (10, 0.1)$.

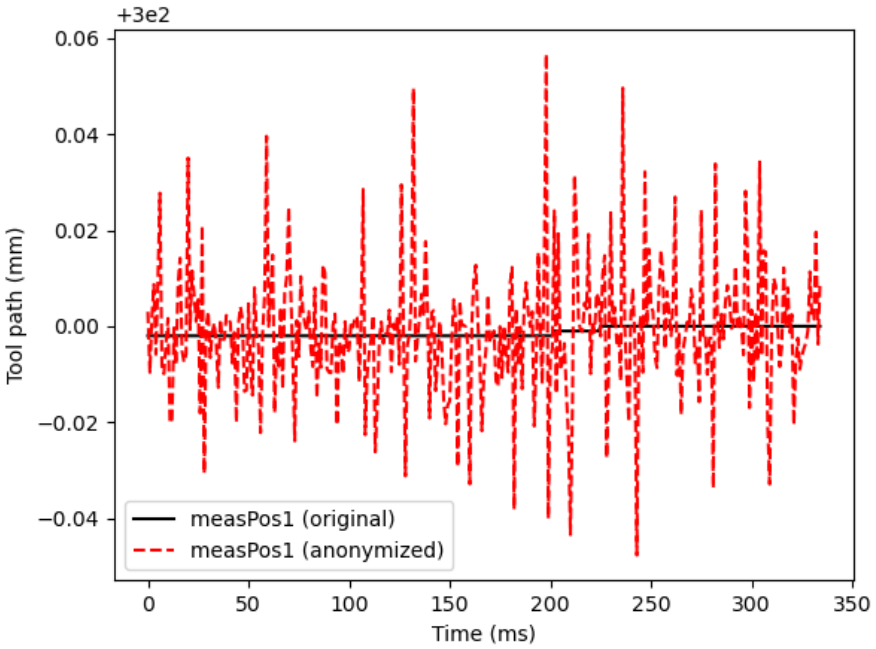
The differing effects of the parameter configurations on the original data are visible by examining the scale of the y-axis. For Figure 7.8a, the range of the statistical noise is $[830; -500]$; for Figure 7.8b, the range is $[301; 299]$ [3]. This shows the extreme effects parameter configuration can have on the results of anonymization via DP[4].

The parameter ϵ describes the requirements for the strength of the privacy guarantee offered by DP. It can be interpreted as the *privacy budget* available for anonymization. With a smaller value of ϵ the privacy budget is reduced, meaning the data needs to be available in a state close to the original data. The parameter S on the other hand describes the amount of change allowed by the LaPlacian distribution. Figure 7.8a shows the case where a small privacy budget is available but a

[2] We set the second parameter of differential privacy to $\delta = 0$ (see Section 7.3.1).
[3] Note that due to visual representation of a small numerical space in Figure 7.8b, the small variance in the baseline for the machine tool of 300 ± 0.005 becomes visible, e.g., between $x = [200; 250]$.
[4] The same is true for other statistical noise distributions, e.g., Gaussian distributions [130]



(a) Variable *measPos1* with $\epsilon = 0.1$ and $S = 10$.



(b) Variable *measPos1* with $\epsilon = 10$ and $S = 0.1$.

Figure 7.8: Difference in LaPlacian noise distribution for DP.

high level of variance or change is allowed. Figure 7.8b shows the reverse case with a high budget available with only little variance.

The values chosen for ϵ and S show extreme differences between available privacy budget and sensitivity. The values are chosen for illustratory purposes. In practice, however, a more reasonable balance between the values is desirable. Finding an optimal configuration for the parameters ϵ and S in regard to the use case at hand (see Section 4.2.2.3) is the topic of the following section.

7.5.1.3 Privacy Parameter Estimation

The variance of anonymized data (see Section 7.5.1.2) is a challenge for privacy-preserving data acquisition. Privacy-preserving data mining requires high customization of the privacy models used according to the use case at hand [121]. In order to establish suitable values for ϵ and S, we take several measurements and data samples into account. Our goal is to find a range for the parameters ϵ and S that allow parameter deviation to stay inside a certain bound.

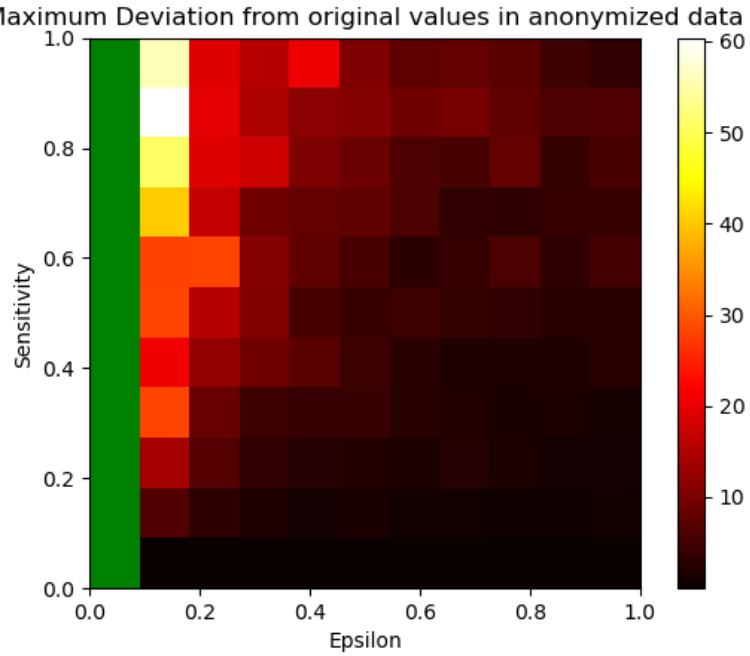


Figure 7.9: Anonymized data's maximum deviation from original data (parameter range $\epsilon = [0; 1]$ and $S = [0; 1]$).

Figure 7.9 shows the maximum deviation from the original value for different parameter combination of ϵ and S denoted as (ϵ, S). The deviation is measured in absolute values from the original data. The brighter the color in the heatmap, the higher the deviation of the anonymized data from its original value. For areas colored in green, the anonymization algorithm becomes numerically unstable[5]. The depicted deviations are the maximum deviations recorded in a sample of 100 subsequent applications of our LaPlacian differential privacy algorithm. The choice for maximum instead of average values reflects our goal to find reasonable parameters as maximum values can lead to a failure of the algorithm.

An observation on parameter behavior can be made by examining Figure 7.9. For this, we divide Figure 7.9 in two sets by imagining a line with a slope of 1 starting at $(0, 0)$. The set above this line shows stronger deviations then the set below the line. This observation can be explained by the mathematical properties of a LaPlacian differential privacy algorithm. The equations of differential privacy (see Equation 7.1) and of the LaPlacian distribution resolve to the fraction $\frac{S}{\epsilon}$ [130]. This is also reflected in the implementation of our algorithm [264].

This can also be observed in Figure 7.10. Here, the results for the same test run are plotted but with a range for ϵ from $[1; 2]$. We extend the view for the parameter ϵ rather then for S as deviations become more extreme with rising numbers of S (see Section 7.5.1.2, especially Figure 7.8).

The illustrated data of Figure 7.10 is not surprising as S tends to increase the variance or change within the LaPlacian noise distribution. The slope of the line is flatter as the view on the area of data examined has shifted but the trend is still the same. In fact, we found the best results for parameter combinations that are on or close to that imaginary line, e.g., $\epsilon = 0.5, S = 0.5$ (see Section 7.5.1.4).

Our set of variables (see Table 7.1) consists of time-sequential data variables that are a common use case in manufacturing [144]. That is, why we expect the parameter ranges discussed above to be useful for further applications. Nevertheless, new data variables and especially data types, e.g., trajectory data, need to be considered differently during initial usage.

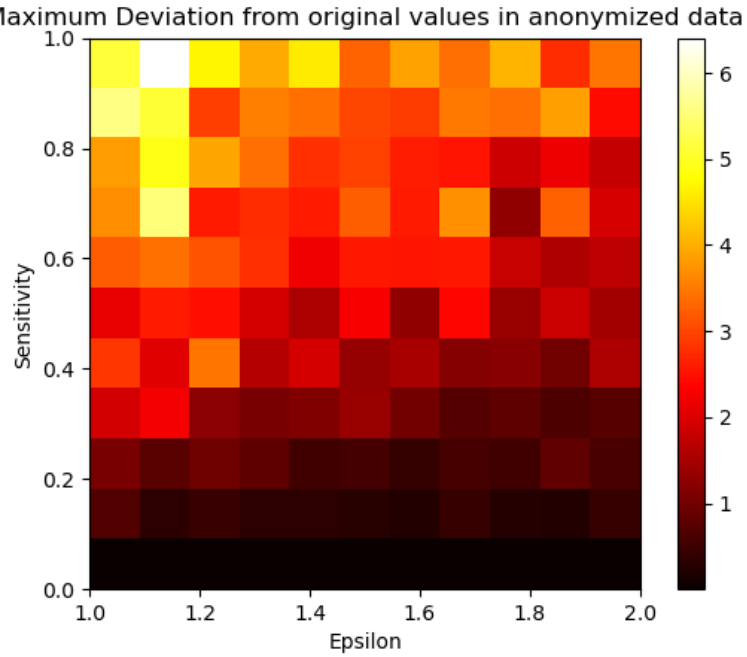


Figure 7.10: Anonymized data's maximum deviation from original data (parameter range $\epsilon = [1; 2]$ and $S = [0; 1]$).

7.5.1.4 Evaluation of Differential Privacy

For evaluation of our privacy model, we used the dataset shown in Figure 7.11. Here, the character string *VDW* is seen. The characters visualize the path of a machine tool within a 2D coordinate system. The tooling machine used is a state of the art mill-turning machine located at laboratories of the Technical University of Munich (TUM) that is used during the research project A4O (see Section 1.4). The path of the machine tool was programmed with G-code, a widely used CNC programming language [267].

For evaluation of our privacy model and the underlying LaPlacian distribution, we choose three pairs p of parameters $p = (\epsilon, S)$ with different expected results (see Figure 7.9): $p_1 = (\epsilon = 0.1, S = 0.9)$, $p_2 = (0.5, 0.5)$, and $p_3 = (0.9, 0.1)$. We chose the pairs as they offer a diversity in results from highly anonymized data (p_1) over a balanced approach (p_2) to a only minimally anonymized data (p_3). We discuss

[5] This is expressed by the software framework by returning *not-a-number* (NaN). More sophisticated data structures for numerical computation may resolve this issue to some degree. For our test run, however, we achieved sufficient results by using standard *float* and *double* data types.

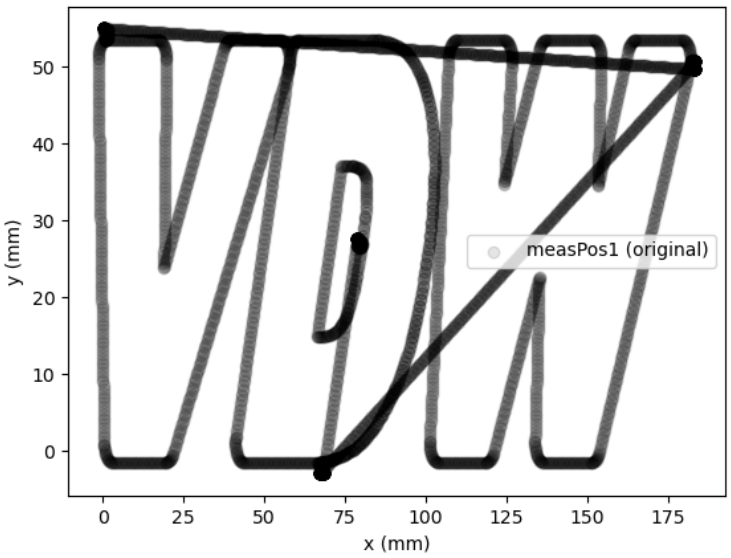
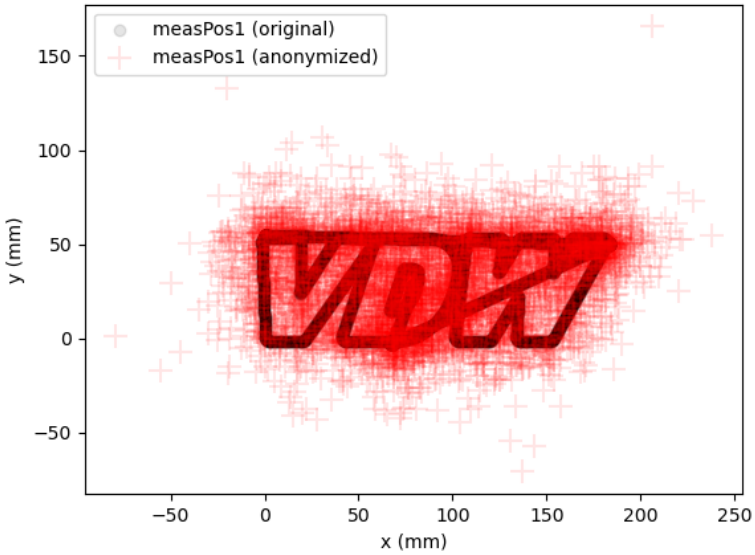


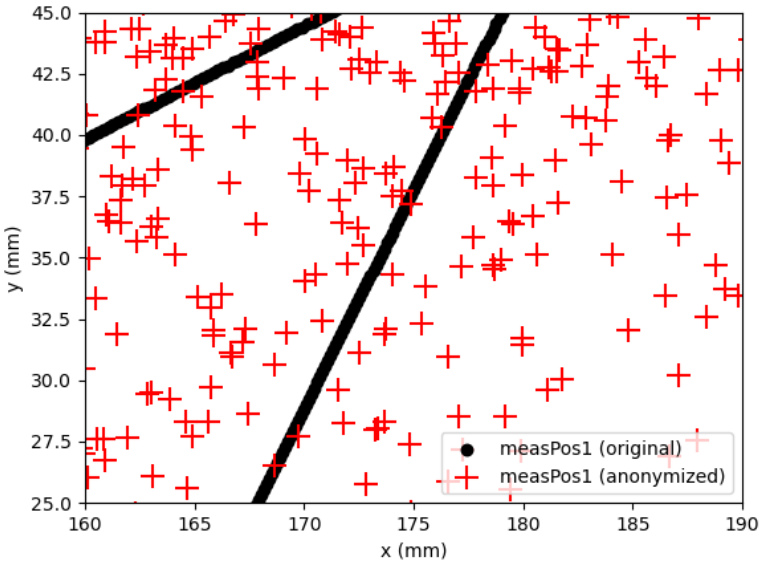
Figure 7.11: Original tool path.

the results by our observations with the naked eye as it serves mostly for illustrative purposes and as we try to take on the perspective of potential attackers. We like to point out that an attacker, however, may be in possession of advanced statistical tooling at some point in the future allowing him to make automated inferences about the specific data set(s) and use case(s). However, at the time of compilation of this thesis, none such attack tooling is known.

The results for p_1 are shown in Figure 7.12. In these figures, the x- and y-coordinate of the 2D coordinate system are located on the x- and y-axis of the graph respectively. For better demonstration, we show the entire set of not anonymized and anonymized data (see Figure 7.12a) and an excerpt from that data that is magnified for a detailed view within the coordinate range $x = [160; 190]$ and $[25; 45]$ for x- and y-coordinates respectively (see Figure 7.12b). As seen from Figure 7.12a, the anonymized data points are present within the entire area of the figure making the original data (shown in blue color) unrecognizable from the anonymized data (red crosses). This is further confirmed by magnification (see Figure 7.12b) where an individual anonymized data point cannot be mapped directly to its original data point. Estimation of the original distribution of the data points is not possible



(a) Data anonymization with $\epsilon = 0.1$ and $S = 0.9$.



(b) Enlarged excerpt of Figure 7.12a.

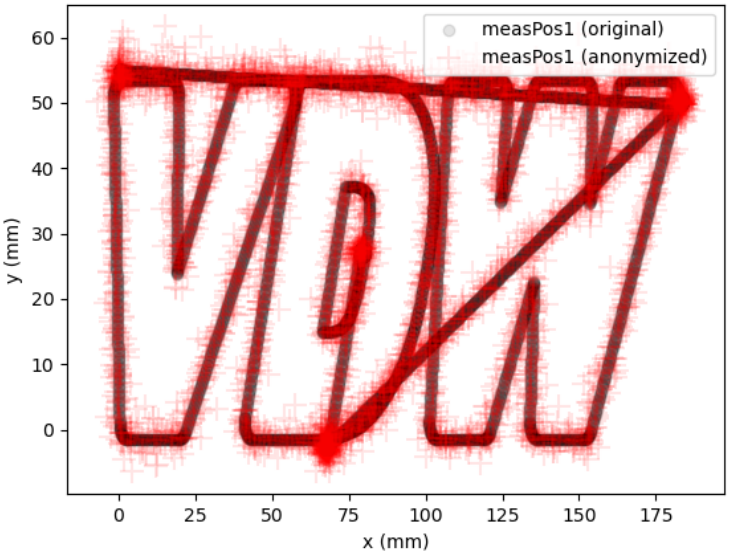
Figure 7.12: Original (blue) and anonymized (red) machine data with $p_1 = (\epsilon = 0.1, S = 0.9)$.

whether by attackers or data analysts. The usability of the data is, therefore, not given as the anonymized data basically represents random data to an analyst.

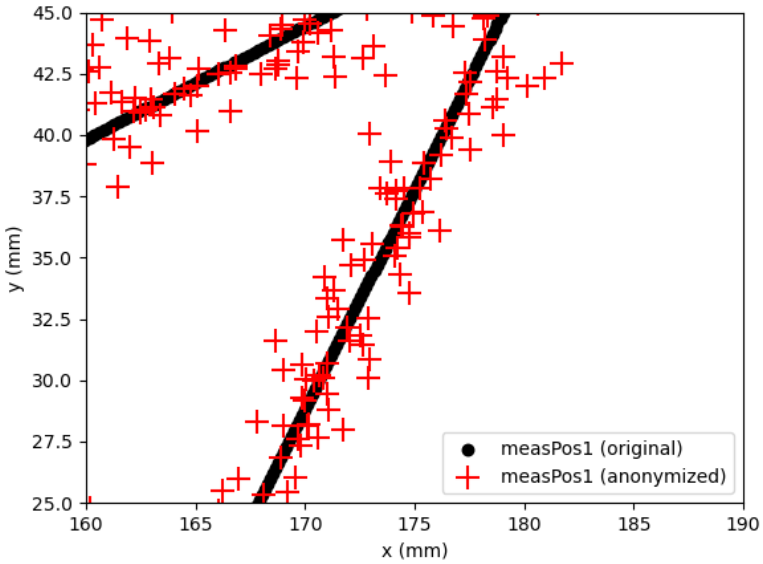
Figure 7.13 shows the results for the balanced pair p_2. Here, the outlines of the string *VDW* can be seen more clearly (see Figure 7.13a). However, a detailed view (same coordinate range as in Figure 7.12b) shows that the distribution of the anonymized data points still contains variance within their distribution (see Figure 7.13b). For attackers only in possession of the anonymized data points, it cannot be inferred reasonably whether the points follow a linear distribution or a distribution with a higher polynomial order. Here, the reconstruction of part geometry is possible, however, only with strong assumptions that cannot be verified. On the other side, the analysis of this data can still lead to reasonable results depending on the concrete target of the analysis.

Finally, we show the results for minimal anonymization in Figure 7.14. The data points match the original data closely (see Figure 7.14a). This is further confirmed by magnification (coordinate range as Figure 7.12b and Figure 7.13b) where a linear distribution of the original data can be easily estimated by human attackers by meeting weak assumptions (see Figure 7.14b). This parameter pairs offers only minimal protection in regards to data privacy for data producers; the usability of the data, however, is high.

The privacy model with the implemented anonymization algorithm is tested during the course of the research project A4O (see Section 1.4) [46, 47]. We found a reasonable compromise between data privacy and usability for data analytics by using the balanced parameter pair p_2.

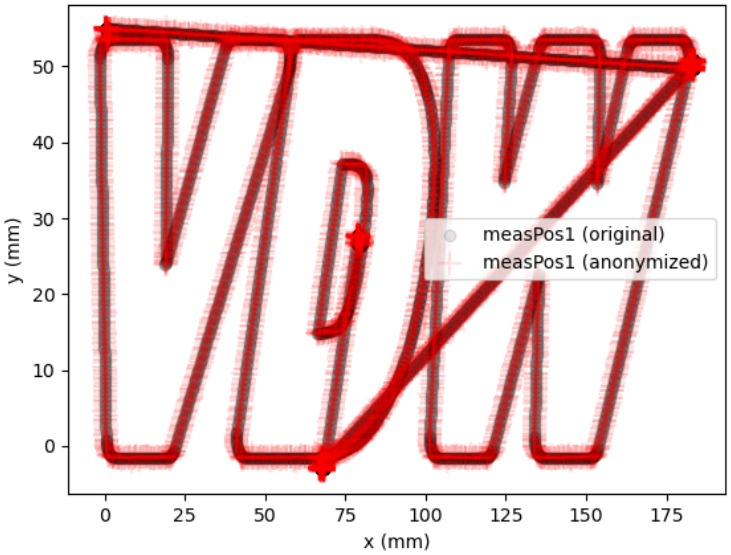


(a) Data anonymization with $\epsilon = 0.5$ and $S = 0.5$.

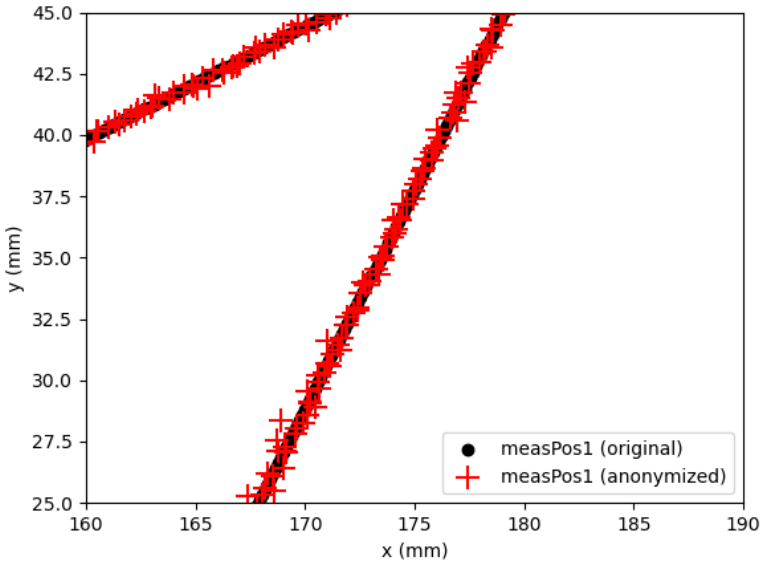


(b) Enlarged excerpt of Figure 7.13a.

Figure 7.13: Original (blue) and anonymized (red) machine data with $p_2 = (\epsilon = 0.5, S = 0.5)$.



(a) Data anonymization with $\epsilon = 0.9$ and $S = 0.1$.



(b) Enlarged excerpt of Figure 7.14a.

Figure 7.14: Original (blue) and anonymized (red) machine data with $p_2 = (\epsilon = 0.9, S = 0.1)$.

7.5.2 Attack Scenario 3: Privacy-Preserving Data Analysis

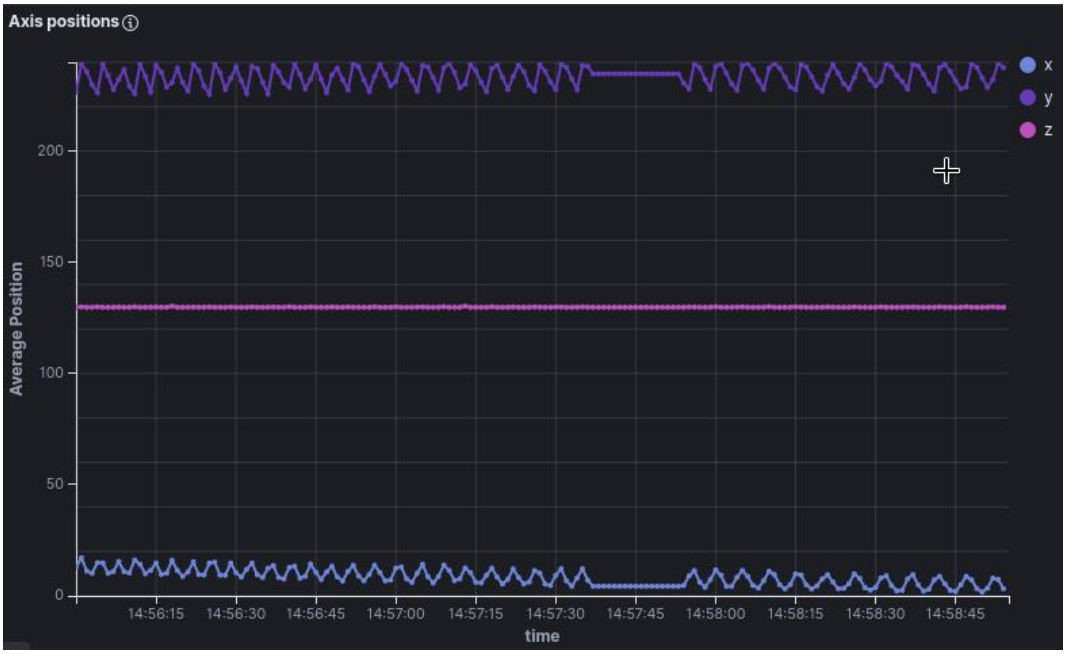
In this section, we evaluate our implementation described in Section 7.4 based on the description of Attack Scenario 3 given by Section 7.3.2.1. We focus on the aspect of data acquisition and show, how it is used for general evaluation purposes other than information security. There are also multiple cases, where data can be used reasonable for information security evaluations, e.g., for anomaly detection and intrusion detection systems [49].

Our experimental setup consists of the G350 tooling machine, a standard IPC, and an externally accessible cloud infrastructure. The evaluation serving as showcase as this experiment is based on a method for chatter detection using an auto-correlation coefficient [252]. This coefficient is computed by using a signal from an external acceleration sensor. The auto-correlation coefficient is computed within the local chatter analysis module (cf. Figure 7.5 and Section 7.4.2) and the data sent via the remote DCM together with other required data towards the data analyst within the cloud infrastructure. In addition, data collected from a PLC via OPC UA is also transmitted into the cloud. The variables sent are related to the tool path (see Section 7.2.1 and Section 7.5.1.1), namely, *measPos1[u1, 1]*, *measPos1[u1, 2]*, and *measPos1[u1, 3]* (cf. also Table 7.1). These variables record the position of the machine tool in a three dimensional coordinate system over time. This data is anonymized using the parameter configuration ($\epsilon = 0.5, S = 0.5$) for differential privacy and then sent to the cloud together with the auto-correlation coefficient. Once received by the remote DCM, the data is stored in the remote database.

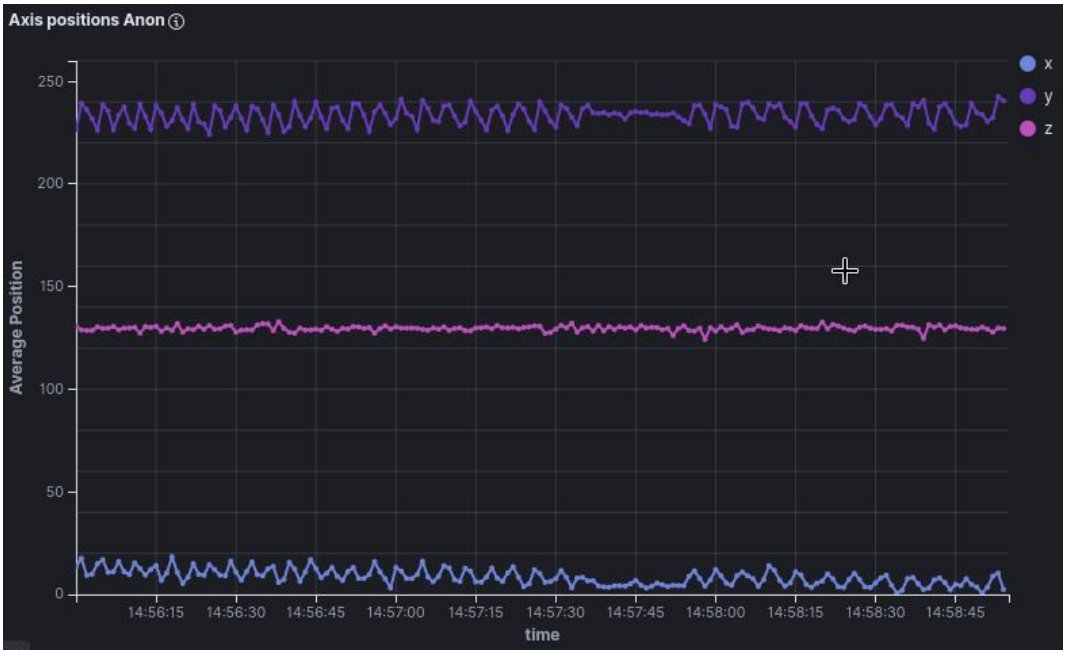
If machine chatter is detected with the auto-correlation coefficient, the tool path data can provide information on the reasons for the machine chatter. Note that also other data can provide similar insight, however, we use tool path variables for illustrative purposes. The examination is conducted by a human-in-the-loop in our experimental setup.

Figure 7.15 shows the tool path collected for this experiment. The figure shows two screenshots from the local and remote dashboards within our privacy-preserving data sharing platform. In Figure 7.15a, the screenshot from the local dashboard is showing the data prior to processing by the anonymization module. In Figure 7.15b, the anonymized data as visualized by the remote dashboard is shown.

Machine chatter is detected with the used method, a subsequent analysis of the reasons for it are subject to future work by other authors and out-of-scope for a



(a) Plain tool path data.



(b) Anonymized tool path data.

Figure 7.15: Recorded tool path data for Experiment 3 (Screenshots).

thesis focused on information security. The evaluation shows, however, that the transmission of reasonable anonymized data via our privacy-preserving data sharing platform is possible. This data can be used to conduct meaningful optimization operations within manufacturing.

For the remainder of this section, we iterate again on the application of differential privacy discussed by Section 7.5.1 in order to show the necessity of proper parameter configurations. Figure 7.16 shows the results of the simulation in Attack Scenario 2 (see Section 4.2.2.2, specifically see Figure 6.6) if it were anonymized by our application discussed here. The derived parameter configuration used is the same we used in this chapter, i.e., $(\epsilon, S) = (0.5, 0.5)$. Due to applied perturbation induced by differential privacy, the normal operation seen in Figure 6.6 cannot be reconstructed from the anonymized data set. In general, this is a desirable result for the data provider as its IP is protected. However, given the context of Attack Scenario 2, this is not desired if value for the data provider is to be provided. On the one hand, the critical threshold is reached at two points for normal operation making the results from this data a false positive. On the other hand, the critical threshold is reached several times during an ongoing attack. An anomaly or intrusion detection system operating only under data anonymized with $(\epsilon, S) = (0.5, 0.5)$ cannot make reasonable statements about the ongoing state of information security within the monitored environment. This demonstrates the importance of parameter estimation for individual use cases.

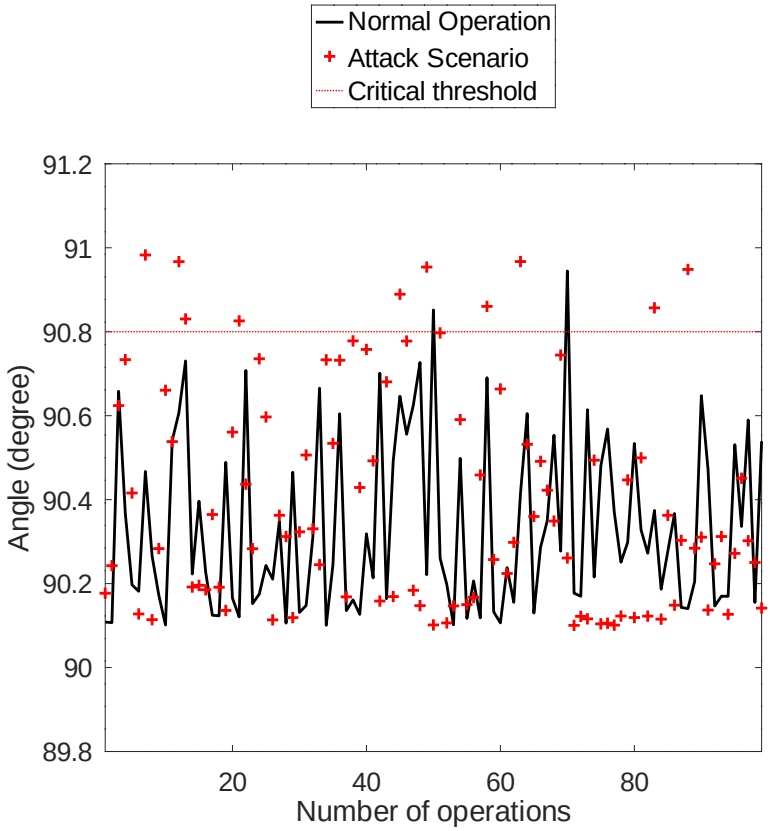


Figure 7.16: Simulation results of Experiment 2 ($\alpha_C = 90.8$) using anonymized data ($\epsilon = 0.5$ and $S = 0.5$).

7.6 Discussion

In this section we discuss the results our privacy-preserving data publishing application delivered when applied to industrial environments. We start by evaluating our application based on the results produced by experimentation with our attack scenario. This validation of our testbed is conducted primarily via prototyping and in accordance to the methodology laid out by us in Section 7.1. As mentioned in Section 6.5, prototyping in general involves building a functional but limited version of our application. For one, the prototype for our application is limited in the sense that not all of the concepts we envisioned are realized. That is, federated learning is not considered within our prototype. This is for several reasons, one of which being that we intended to focus on our core objectives that are laid out in the research project A4O under which the work presented in this chapter was funded (cf. also Section 1.1). Within this research project, our core objectives were anonymization of machine data and realization of an edge computing framework with a connected cloud infrastructure. The additional inclusion of PETs was not part of the project. Still, PETs are considered by us in this work as they potentially can provide a valuable addition to our application. However, due to the limited time frame we could access the shopfloor environment our research was based during the time span of 2019-2021, we decided to focus on the proper evaluation of the privacy model (see Section 7.5.1) rather then the conception of an additional algorithm for federated learning that may or may not have been properly implemented and evaluated. Another reason why our prototype may be limited is that our selected attack scenario (see Section 7.3.2.1) only reflects a small fraction of the possible attack scenarios that might be considered within privacy-preserving manufacturing [182, 145]. That is to be expected and providing a thorough validation of a privacy-preserving application that covers a wide range of possible scenarios appears impractical and may also not be achievable as new attacks for the extraction of IP from machine data may be possible at the time but remain yet unknown. However, the attack scenario selected by does reflect upon a general trend in manufacturing for the development of new business cases [1, 8, 3] (cf. also Section 1.1). We selected a specific optimization problem that is, however, built upon a general infrastructure of tooling machine, edge computing, and cloud-based services. This allows us to demonstrate the performance of our application under a broad perspective. In addition to this, we constantly assessed the performance, functionality, and adherence to requirements

for our application. This is part of an iterative refinement process based on the evaluation results, thus, ensuring that our testbed can improve further on from the prototyping stage.

The privacy model of differential privacy that is used by us in order to perform the anonymization showed satisfying performance. For one, the anonymization can be performed close to real-time, which also includes storage and transmission of the data. We note that the transmission rate of the data from the network edge at the company's site towards the externally hosted cloud provider is subject to the overall availability of the public internet, which is a factor that cannot be influenced by us. However, during our experiments, we experienced no delay in the transmission via the internet, which may be influenced by the fact that both, the cloud provider and the shopfloor are connected to the same provider and are located in close physical proximity. While different use cases typically may require either a different set of parameter configuration or another privacy model entirely, the basic infrastructure in conjunction with differential privacy seemed more than capable of handling our use case and attack scenario. Our privacy-preserving approach, i.e., differential privacy as privacy model with LaPlacian probability distribution as algorithm, may be suitable for other applications that are similar. However, this needs to be evaluated in the context of the potential new use cases. In our experiments, we established a set of parameters that can be used for data acquisition of machine tool data [47] (see Section 7.5.1). The results of our experiments can be used for the data described and the experiments suggest that they can be reasonably used for other data variables that behave in a similar manner. However, slight variations can cause issues with data utility even for similar appearing use cases. One extension to our privacy-preserving scheme to address this can involve machine learning in conjunction with federated learning as described in Section 7.3.1.1 with its automated training processes [109, 148]. This eliminates the need for requiring non-anonymized training datasets, which can result in acceptance problems by the data producers as highlighted in Section 1.2. The process for estimating the anonymization parameters themselves can be further extended. Possible approaches are related to the automated estimation of anonymization parameters [184] or to using the inherent noise in a given data set [268].

We further presented a software module called DCM that secures communication via a TLS channel and a corresponding pair of public and private keys. Via the DCM, it is possible to send a delete instruction to the DCM to remove all previously stored

data transmitted by the same public key. This function is intended to enhance the trust of the data producers. However, the data producer can, at the current level of implementation not independently verify that the data is in fact deleted. This can be further improved upon by employing a verifiable open-source implementation. As our implementation relies exclusively on open-source software components, this can be added with manageable effort. We furthermore showed, that our privacy-preserving data sharing architecture can be used to collect and extract data from manufacturing environments. As our approach for extraction is secure and privacy-preserving, this can increase the willingness of machine tool operators to partake in collaborative data sharing. While subsequent analysis of the specific use case at hand is subject to future work by other authors it still demonstrates the viability of our approach. Other applications like anomaly detection can profit from the same method [46]. Furthermore, our testbed for information security evaluations can be embedded within the the cloud. This is possible via the use of Docker containers (see Section 7.4) that allow to execute the simulation environment and by sending PCAP files (see Section 6.3) with anonymized process parameters. This can pave the way to a cloud-based information security evaluations environment where a digital twin of a real manufacturing environment is used to execute attack scenarios.

7.6.1 Future Work

For evaluation of our application, we implemented an attack scenario that is based on the description of a discrete manufacturing process (see Section 3.1.1 and Section 4.2.2.3). As outlined in Section 6.5.1, discrete manufacturing and the processes are reasonable a choice for simulation and evaluation. Furthermore, discrete manufacturing processes allow for the generation of structured data from the simulation environment. This is useful in the implementation of our privacy-preserving data sharing framework as it allows for a more accurate and straightforward evaluation of our control. Also, it provides a foundation for data-driven privacy techniques such as federated learning, which is a future possibility for extension of our scheme. However, we should consider the application of privacy-preserving techniques for continuous manufacturing processes. This way, we can cover more possible use cases for our data sharing architecture. The evaluation of a continuous process with the same quality as our evaluation here in this chapter does, however, require a real-life continuous process. Continuous processes, however, are more difficult to control than their discrete counterparts. Also, the may be irreversible, which could make

verification of privacy models and PETs more difficult. More research into applying our framework to those processes is required.

We proposed an architecture that is capable of providing federated learning to machine data. However, due to the reasons outlined above, federated learning algorithms and models are not realized within our architecture at the moment. For the most, this is owed to temporal and spatial constraints when accessing our testing facility. Federated learning is considered by us to provide a valuable contribution to the field of privacy-preserving data acquisition within competitive environments and their subsequent usage for shared, value-added services. Therefore, we should consider the construction of a new testing environment with an extended version of our framework.

The use case we evaluated, that is, chatter detection in tooling machines, is a relevant use case for the construction of high-quality tooling machines. Other use cases in the field of mechanical engineering that can be considered are condition monitoring or predictive maintenance. However, in Chapter 6, we discussed the application of the concept of digital twinning within testbeds for the realization of use cases related to information security. As we showed in Section 1.1 and Section 2.1.3, information security is an important aspect within connected manufacturing. Therefore, we should also consider use cases from the field of information security as the purpose for our privacy-preserving application. One such use case that is well-suited in this context is anomaly detection [49, 216]. Data extracted from the production environment can be used to train models for anomaly and intrusion detection.

8 Conclusion

In this chapter, we draw conclusions on our work discussed in this thesis and the results we gathered. We provided some conclusions on the individual topics of this thesis in the corresponding section: the broad topic of information security in manufacturing together with attacker models, attack vectors, and our overall system architecture is discussed by Section 4.4. The topic of modeling manufacturing environments with digital twins and information security is discussed in Section 5.4. The conclusions on the development of our testbed for information security with support for digital twins is the topic of Section 6.5. The results for our privacy-preserving data sharing platform that enables collaborative analysis and data extraction from competitive environments are discussed by 7.6. In this section, we discuss the common outlook of our thesis, which is first conceptualized by the overall system architecture in Section 4.3.

Our overall system architecture combines digital twins with privacy-preserving techniques for their usage within the domain of manufacturing while being grounded on a novel modeling approach. In Chapter 4 we provide a thorough examination of information security in the domain of connected manufacturing. This examination of Chapter 4 is different than the background provided on it in Section 2.1.3 or the related work discussed by Section 3.1 as it provides tangible results for the usage within our thesis. First, we provide a clear and detailed description of our attacker model that is used within this thesis. We derive a generic attack vector model that does, together with the attacker model, provide an understanding for our attack scenarios. These attack scenarios are the basis for the evaluation of the evaluation of our work in the remaining chapters. Our modeling approach proposed in Chapter 5 is based on UML state machine charts, which allows for a better understanding of complex systems such as digital twins or manufacturing environments in general. They provide a visual representation of the behavior of a system and the interactions that occur within it. Complex systems such as manufacturing processes can be abstracted by compartmentalization into more manageable subcomponents. UML

state machine charts provide a view on such systems that is focused on software and data, key enablers in connected manufacturing (see Section 2.1.1). We can model how data is processed by the system, including how it is collected, stored, and deleted. This provides insights into the data connection between physical and digital twin and can further enhance our capabilities of data acquisition from the modeled systems. We use our modeling approach to provide a reference scenario of a manufacturing environment on which the attack scenarios detailed in Section 4.2.2 are realized on.

In Chapter 6 we discuss the design, implementation, and evaluation of our testbed for information security evaluations in connected manufacturing [55, 60]. The testbed here is the realization of one part of the conceptual system architecture (cf. Figure 4.6). In particular, we consider digital twinning and its usage within information security within the testbed. For the implementation of the digital twin we considered design guidelines for the conception of testbeds and applied those to connected manufacturing, information security, and digital twins [33]. This approach enabled us to built our testbed and the digital twin on a sound technological basis. The digital twin is often not considered in the context of a wider production facility. The integration of a digital twin within a testbed provides tooling for a proper simulation of remaining components [55]. Furthermore, we consider the topic of fidelity for the digital twin in depth and provide definitions for different levels of fidelity [54, 42] (cf. Section 6.1.5.4). With our testbed, we achieved the implementation of a medium-fidelity digital twin that is evaluated in the context of our defined attack scenarios. We evaluated our digital twin within two attack scenarios: Attack Scenario 1 (Sorting, see Section 4.2.2.1) and Attack Scenario 2 (Sawing, see Section 4.2.2.2). These are inspired by real-life scenarios and implemented within a laboratory setup by using real industrial equipment [69, 11, 9]. This allowed us to test our digital twin in the presence of an active adversary that executed attacks against the twin. This is related to several potential use cases a digital twin can be used for when promoting information security within manufacturing, such as security testing or system testing [28]. Our comparison with the results from the digital and physical twins showed that we achieved a medium-fidelity digital twin that performed with a sufficient level of detail in our attack scenarios that are related to these use cases. Compared to related work (see Section 3.1), the evaluation of how well a digital twin meets its expected behavior and accurately represents its physical twin in context of real-life attack scenarios is difficult to identify in literature [42, 156, 28] (cf. also Section 3.1).

In Chapter 7 we discuss the design, realization, and evaluation of our privacy preserving data sharing architecture for manufacturing [46, 47]. The architecture is designed for enabling collaborative data analysis within competitive environments (see also Section 1.1 and Section 1.2). For this, we develop a privacy control for the anonymization of sequential machine data, that is, time-series data. This represents the realization of the second of our conceptual system architecture as seen by Figure 4.6. In order to provide a reasonable privacy control, we raised requirements among plant operators that employ tooling machines within their production facilities and derived design guidelines from those requirements (see Section 7.1). Our privacy model of differential privacy and the corresponding PET of federated learning enable us to provide strong privacy guarantees with a scalable architecture. The evaluation of our privacy model is conducted with real-life data from a production environment. The subsequent implementation of our privacy-preserving architecture is also conducted within the same shopfloor training facility with real-life tooling machine and other OT equipment. As test case we implemented Attack Scenario 3 (Data Sharing, see Section 4.2.2.3) that is designed to provide optimization of machine tool construction while maintaining the intellectual property of participating data providers. Our results showed that it is possible to prevent the loss of intellectual property from production facilities while still enabling domain-specific optimization solutions.

In summary, for the evaluation of our work we used real industrial equipment, that is, a robotic arm and a tooling machine, and took specific scenarios from real-life challenges experienced within the domain of manufacturing into account [46, 69, 11, 9]. Such practical aspects are rarely considered by other authors and provides a justification for our concepts developed within this thesis [28, 149]. Also, we support the basic principles of scientific experimentation within our testbed as discussed in detail by us in Section 6.1.5.2. Here, we point out that only about 10% of published information security testbeds for manufacturing *at best* consider all four required principles for scientific experimentation [33]. In fact, it appears that only [27] comprehensively follows the rigorously scientific approach outlined by themselves according to the survey of [33]. For digital twins, to the best of our knowledge, no such work exists (cf. Chapter 3).

8.1 Outlook & Future Work

In this section we discuss possible future work of our thesis and provide an outlook on the continuation of our platform. We discuss possible future work for our different concepts and their respective implementations already at other sections within this thesis. The future work for our modeling approach for modeling information security and digital twins in manufacturing is the topic of Section 5.4.1. The future work for our testbed is given by Section 6.5.1. The future work for our privacy-preserving data sharing platform for collaborative analysis and data extraction from competitive environments is discussed by Section 7.6.1. Therefore, we focus our discussion of future work in this section on the overall picture as it is given by Section 4.3.

We use the attack scenarios described in Section 4.2 for evaluation of our implementations in Section 6.3 and Section 7.4. Those attack scenarios are limited by the assumptions of the reference scenario given by Section 5.3, which they are based upon. In particular, in that reference scenario we narrow the potential industrial processes down to discrete assembly processes. While this does cover a substantial fraction of the overall manufacturing processes, this leaves continuous manufacturing processes out of scope for our thesis (see also Section 2.1 and Section 3.1.1). Our reasoning for using discrete assembly processes rather than continuous manufacturing processes is motivated by two aspects. First, discrete processes or easier to control and can be reversed to some degree, which makes them in general a reasonable choice for the realization of an evaluation use case. Second, the equipment available to us is designed for the use in discrete manufacturing setups and, thus, it seemed reasonable to us to use that equipment for their intended purposes. Nevertheless, the extension towards continuous processes can be achieved within our architecture as we orient ourselves on different related design goals such as modularity or flexibility during our work on this thesis [33].

Using digital twins and privacy controls for information security evaluations in manufacturing offers huge potential for a number of use cases [28]. We conducted research on security testing and privacy within this thesis, however, some use cases still remain unexplored by us. In order to evaluate our work, it should be extended to cover more use cases. Several use cases for information security and privacy exist in connected manufacturing especially when considering digital twins [28]. For example, digital twins can be used for anomaly and intrusion detection systems (IDS) [37, 49]. The application of digital twins seems especially promising within

the context of physical-based IDS [269, 28]. Physical-based IDS are characterized by their use of physical simulation models, such as co-simulation [30], in order to predict a target system's behavior for the detection of anomalies. Digital twins can be equipped with a model of the target system and a corresponding simulation of the physical processes present within the production environment. This way, digital twinning could be leveraged for anomaly detection within manufacturing devices. For example, sensor readings deviating from expected values could be detected by such an IDS. Also other use cases, such as training of personal [95] or honeypots [13], could be realized with digital twins. A digital twin can be included within a virtual environment that is developed for training purposes. With such a training environment that features an accurate representation of a target system, individuals can test and experience the consequences and potential impact a cyber attack can have on the target system first-hand. Furthermore, using a digital twin as a honeypot, attackers can be diverted from the real, productive system. This way, attacks can be detected early on and data from real-world attacks can be collected for further research purposes. Realization of such new use cases can be supported by the continued integration of our work. The privacy preserving architecture described by Chapter 7 can be combined with the information security testbed proposed by Chapter 6. This is possible due to use of common interfaces such as PCAP, OPC UA, and supporting technologies like Docker. With such a combined architecture, enhanced use cases can be envisioned, for example, privacy-preserving data extraction for training of IDS within a cloud infrastructure that receives input from diverse manufacturing environments.

Our domain of application is connected manufacturing (see Section 2.1). We focus on the domain of manufacturing and we take the specifics of that domain into account. However, manufacturing shares some properties with other domains. This is why our work can be transferred to some of those domains with lower effort as compared to other domains. The properties that aid in the application of our scheme to other domains are of technical and economical nature. For technical properties, those properties that are related to the long-term usage in harsh environments can be considered for this. For the usage in harsh environments, such requirements related to the dependability of the equipment need to be considered [48]. These include robustness and reliability among others. Furthermore, real-time capabilities need to be considered for such domains as well [21]. Also, domains where a large number of market participants are present can be considered as an economical requirement.

These market participants can be both: suppliers or users of equipment and services. Our privacy-preserving data sharing architecture allows collaboration in such competitive environments to some degree. Domains that share these requirements are, for example, building automation or medical devices. To summarize, any market relying on embedded systems and acquisition of data can benefit from the concepts developed in this thesis.

List of Figures

List of Tables

Acronyms

AI	artificial intelligence
AiF	Arbeitsgemeinschaft industrieller Forschungsvereinigungen
APT	advanced persistent threat
ARPANET	Advanced Research Projects Agency Network
BMBF	Federal Ministry of Education and Research
CNC	computer numerical control
CPS	cyber-physical system
DCM	Data Control Module
DoS	Denial-of-Service
DP	Differential Privacy
FSM	Finite State Machine
HIPAA	Health Insurance Portability and Accountability Act
HITECH	Health Information Technology for Economic and Clinical Health Act
ICS	industrial control system
IDS	intrusion detection system
IIoT	Industrial Internet-of-Things
IoT	Internet-of-Things
IP	intellectual property
ISO	International Organization for Standardization

IT	information technology
JSON	JavaScript Object Notation
MES	manufacturing execution system
MITM	man-in-the-middle
OPC UA	Open Platform Communications Unified Architecture
OT	operational technology
PET	privacy-enhancing technology
PLC	programmable logical controller
ROS	Robot Operating System
SM	subtractive manufacturing
SME	small- and medium-sized enterprise
SSL	Secure Sockets Layer
TLS	Transport Layer Security
TOE	target of evaluation
TUM	Technical University of Munich
UML	Unified Modeling Language

Bibliography

[1] H. Kagermann, J. Helbig, A. Hellinger, and W. Wahlster. Recommendations for implementing the strategic initiative INDUSTRIE 4.0: Securing the future of german manufacturing industry; final report of the Industrie 4.0 working group. Technical report, Forschungsunion, 2013.

[2] M. Speringer and J. Schnelzer. Differentiation of Industry 4.0 Models - The 4th Industrial Revolution from different Regional Perspectives in the Global North and Global South. Technical report, United Nations Industrial Development Organization (UNIDO), 2019.

[3] IUNO Consortium. Forschung für mehr IT-Sicherheit in Industrie 4.0 - IUNO Projektergebnisse. Technical report, IUNO Consortium, 2019.

[4] T. Philbeck and N. Davis. The fourth industrial revolution. *Journal of International Affairs*, 72(1):17–22, 2018.

[5] O. Bongomin, E. O. Nganyi, M. R. Abswaidi, E. Hitiyise, and G. Tumusiime. Sustainable and dynamic competitiveness towards technological leadership of industry 4.0: implications for east african community. *Journal of Engineering*, 2020:1–22, 2020.

[6] Y. Wang, O. Anokhin, and R. Anderl. Concept and use case driven approach for mapping it security requirements on system assets and processes in Industrie 4.0. *Procedia CIRP*, 63:207–212, 2017.

[7] IUNO Consortium. D2.1-2: Analysen, Anforderungsdefinition und Konzeptentwicklung für den Technologiedatenmarktplatz. Technical report, IUNO Consortium, 2017.

[8] Giehl, Alexander and Bossert, Daniel and Brenner, Gerd and Shaabany, Ghaidaa and Bock, Hans-Peter and Blume, Marco and Schwarz, Reinhard and Frisch, Simon and Wang, Yübo and Gomes, Marcel Ely and Beuttler,

Manuel and Görg, Christian. D2.2-1: Projektergebnisse. Technical report, IUNO Consortium, 2017.

[9] A. Giehl and N. Wiedermann. Security verification of third party design files in manufacturing. In *10th International Conference on Computer and Automation Engineering Proceedings*, pages 166–173, New York, NY, USA, 2018. ACM. Best Presentation Award. URL: `https://doi.org/10.1145/3192975.3192984`, `doi:10.1145/3192975.3192984`.

[10] Esslinger, Ernst and Baier, Uwe and Krespach, Felix and Vinski, Marijo and Bantle, Matthias and Köhne, Hermann and Reers, Volker and Hilbig, Yana and Yalcin, Özgün and Giehl, Alexander and Wiedermann, Norbert. D1.2-2: Simulationsplattform für vernetzte Fertigungsanlagen. Technical report, IUNO Consortium, 2018.

[11] Baier, Uwe and Krespach, Felix and Müller, Patrick and Köhne, Hermann and Nakic, Kai and Scheuermann, Dirk. D1.3-1: Projektergebnisse. Technical report, IUNO Consortium, 2018.

[12] L. J. Wells, J. A. Camelio, C. B. Williams, and J. White. Cyber-physical security challenges in manufacturing systems. *Manufacturing Letters*, 2(2):74–77, 2014.

[13] S. D. Antón, D. Fraunholz, C. Lipps, F. Pohl, M. Zimmermann, and H. D. Schotten. Two decades of SCADA exploitation: A brief history. In *Application, Information and Network Security (AINS), 2017 IEEE Conference on*, pages 98–104. IEEE, 2017.

[14] A. E. Elhabashya, L. J. Wellsb, and J. A. Camelioc. Cyber-physical security research efforts in manufacturing–a literature review. *Procedia manufacturing*, 34:921–931, 2019.

[15] L. Pietre-Cambacédes, M. Tritschler, and G. N. Ericsson. Cybersecurity myths on power control systems: 21 misconceptions and false beliefs. *IEEE Transactions on Power Delivery*, 26(1):161–172, 2010.

[16] S. McLaughlin, C. Konstantinou, X. Wang, L. Davi, A.-R. Sadeghi, M. Maniatakos, and R. Karri. The cybersecurity landscape in industrial control systems. *Proceedings of the IEEE*, 104(5):1039–1057, 2016.

[17] S. Plaga, N. Wiedermann, S. D. Anton, S. Tatschner, H. Schotten, and T. Newe. Securing future decentralised industrial iot infrastructures: Challenges and free open source solutions. *Future Generation Computer Systems*, 93:596–608, 2019.

[18] H. Kayan, M. Nunes, O. Rana, P. Burnap, and C. Perera. Cybersecurity of industrial cyber-physical systems: a review. *ACM Computing Surveys (CSUR)*, 54(11s):1–35, 2022.

[19] M. Rocchetto, A. Ferrari, and V. Senni. Challenges and opportunities for model-based security risk assessment of cyber-physical systems. In *Resilience of Cyber-Physical Systems*, pages 25–47. Springer, 2019.

[20] A.-R. Sadeghi, C. Wachsmann, and M. Waidner. Security and privacy challenges in industrial internet of things. In *Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE*, pages 1–6. IEEE, 2015.

[21] A. Giehl and S. Plaga. Implementing a performant security control for industrial ethernet. In *2018 International Conference on Signal Processing and Information Security*, pages 1–4. IEEE, 2018. URL: `https://doi.org/10.1109/CSPIS.2018.8642758`, `doi:10.1109/CSPIS.2018.8642758`.

[22] S. Plaga, N. Wiedermann, M. Niedermaier, A. Giehl, and T. Newe. Future Proofing IoT Embedded Platforms for Cryptographic Primitives Support. In *2018 12th International Conference on Sensing Technology (ICST)*, pages 52–57. IEEE, 2018.

[23] M. Brunner, H. Hofinger, C. Krauß, C. Roblee, P. Schoo, and S. Todt. Infiltrating critical infrastructures with next-generation attacks- W32.Stuxnet as a Showcase Threat. Technical report, Fraunhofer Institute for Secure Information Technology (SIT), Munich, 2010.

[24] P. H. Nguyen, S. Ali, and T. Yue. Model-based security engineering for cyber-physical systems: A systematic mapping study. *Information and Software Technology*, 83:116–135, 2017.

[25] T. Alladi, V. Chamola, and S. Zeadally. Industrial control systems: Cyberattack trends and countermeasures. *Computer Communications*, 155:1–8, 2020.

[26] C. Siaterlis and B. Genge. Cyber-physical testbeds: Scientific instruments for cyber security assessment of critical infrastructures. 2008.

[27] C. Siaterlis, B. Genge, and M. Hohenadel. EPIC: a testbed for scientifically rigorous cyber-physical security experimentation. *IEEE Transactions on Emerging Topics in Computing*, 1(2):319–330, 2013.

[28] M. Eckhart and A. Ekelhart. Digital twins for cyber-physical systems security: State of the art and outlook. *Security and Quality in Cyber-Physical Systems Engineering: With Forewords by Robert M. Lee and Tom Gilb*, pages 383–412, 2019.

[29] D. Angermeier, K. Beilke, G. Hansch, and J. Eichler. Modeling security risk assessments. In *17th Embedded Security in Cars (escar Europe)*, pages 133–146, 2019. doi:https://doi.org/10.13154/294-6670.

[30] A. Giehl. Development of a co-simulation framework to analyse attacks and their impact on smart grids. Master's thesis, Technische Universität München, 2013.

[31] D. Angermeier, A. Nieding, and J. Eichler. Supporting risk assessment with the systematic identification, merging, and validation of security goals. In *International Workshop on Risk Assessment and Risk-driven Testing*, pages 82–95. Springer, 2016.

[32] H. Holm, M. Karresand, A. Vidström, and E. Westring. A survey of industrial control system testbeds. In *Secure IT Systems*, pages 11–26. Springer, 2015.

[33] U. P. D. Ani, J. M. Watson, B. Green, B. Craggs, and J. R. Nurse. Design considerations for building credible security testbeds: Perspectives from industrial control system use cases. *Journal of Cyber Security Technology*, 5(2):71–119, 2021.

[34] C. Siaterlis, A. P. Garcia, and B. Genge. On the use of Emulab testbeds for scientifically rigorous experiments. *IEEE Communications Surveys & Tutorials*, 15(2):929–942, 2012.

[35] G. C. Rausser and S. R. Johnson. On the limitations of simulation in model evaluation and decision analysis. *Simulation & Games*, 6(2):115–150, 1975.

[36] H.-J. Bungartz, S. Zimmer, M. Buchholz, and D. Pflüger. *Modellbildung und Simulation: eine anwendungsorientierte Einführung*. Springer-Verlag, 2009.

[37] Y. Hu, A. Yang, H. Li, Y. Sun, and L. Sun. A survey of intrusion detection on industrial control systems. *International Journal of Distributed Sensor Networks*, 14(8):1550147718794615, 2018.

[38] F. Zhang, H. A. D. E. Kodituwakku, J. W. Hines, and J. Coble. Multilayer data-driven cyber-attack detection system for industrial control systems based on network, system, and process data. *IEEE Transactions on Industrial Informatics*, 15(7):4362–4369, 2019.

[39] R. Rosen, G. Von Wichert, G. Lo, and K. D. Bettenhausen. About the importance of autonomy and digital twins for the future of manufacturing. *Ifac-Papersonline*, 48(3):567–572, 2015.

[40] H. van der Valk, H. Haße, F. Möller, and B. Otto. Archetypes of digital twins. *Business & Information Systems Engineering*, 64:375—-391, 2022.

[41] D. Adamenko, S. Kunnen, R. Pluhnau, A. Loibl, and A. Nagarajah. Review and comparison of the methods of designing the digital twin. *Procedia CIRP*, 91:27–32, 2020.

[42] D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks. Characterising the digital twin: A systematic literature review. *CIRP Journal of Manufacturing Science and Technology*, 29:36–52, 2020.

[43] G. Breda and M. Kiss. Overview of information security standards in the field of special protected industry 4.0 areas & industrial security. *Procedia Manufacturing*, 46:580–590, 2020.

[44] D. Gaver. Time to failure and availability of paralleled systems with repair. *IEEE Transactions on Reliability*, 12(2):30–38, 1963.

[45] C. J. Lu, W. Q. Meeker, and L. A. Escobar. A comparison of degradation and failure-time analysis methods for estimating a time-to-failure distribution. *Statistica Sinica*, pages 531–546, 1996.

[46] A. Giehl, P. Schneider, M. Busch, F. Schnoes, R. Kleinwort, and M. F. Zaeh. Edge-computing enhanced privacy protection for industrial ecosystems in the context of SMEs. In *2019 12th CMI Conference on Cybersecurity and Privacy (CMI)*, pages 1–6. IEEE, 2019. URL: `https://ieeexplore.ieee.org/document/8962138`, `doi:10.1109/CMI48017.2019.8962138`.

[47] A. Giehl, M. P. Heinl, and M. Busch. Leveraging edge computing and differential privacy to securely enable industrial cloud collaboration along the value chain. In *2021 IEEE 17th International Conference on Automation Science and Engineering Proceedings*, pages 2023–2028. IEEE, 2021. URL: `https://ieeexplore.ieee.org/document/9551656`, `doi:10.1109/CASE49439.2021.9551656`.

[48] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE transactions on dependable and secure computing*, 1(1):11–33, 2004.

[49] P. Schneider and A. Giehl. Realistic data generation for anomaly detection in industrial settings using simulations. In *Computer Security*, pages 119–134. Springer, 2018.

[50] T. Alves, R. Das, and T. Morris. Virtualization of industrial control system testbeds for cybersecurity. In *Proceedings of the 2nd Annual Industrial Control System Security Workshop*, pages 10–14. ACM, 2016.

[51] T. Alves, R. Das, A. Werth, and T. Morris. Virtualization of scada testbeds for cybersecurity research: A modular approach. *Computers & Security*, 77:531–546, 2018.

[52] R. von Solms, H. Van Der Haar, S. H. von Solms, and W. J. Caelli. A framework for information security evaluation. *Information & Management*, 26(3):143–153, 1994.

[53] J. Lee, B. Bagheri, and H.-A. Kao. A cyber-physical systems architecture for Industry 4.0-based manufacturing systems. *Manufacturing Letters*, 3:18–23, 2015.

[54] R. Bitton, T. Gluck, O. Stan, M. Inokuchi, Y. Ohta, Y. Yamada, T. Yagyu, Y. Elovici, and A. Shabtai. Deriving a cost-effective digital twin of an ics to facilitate security evaluation. In *Computer Security: 23rd European Symposium on Research in Computer Security, ESORICS 2018, Barcelona, Spain, September 3-7, 2018, Proceedings, Part I 23*, pages 533–554. Springer, 2018.

[55] A. Giehl, N. Wiedermann, M. Tayebi Gholamzadeh, and C. Eckert. Integrating security evaluations into virtual commissioning. In *2020 IEEE 16th International Conference on Automation Science and Engineering Proceedings*, pages

1193–1200. IEEE, 2020. URL: https://ieeexplore.ieee.org/document/9217004, doi:10.1109/CASE48305.2020.9217004.

[56] F. Tao, B. Xiao, Q. Qi, J. Cheng, and P. Ji. Digital twin modeling. *Journal of Manufacturing Systems*, (64):372–389, 2022.

[57] A. Thelen, X. Zhang, O. Fink, Y. Lu, S. Ghosh, B. D. Youn, M. D. Todd, S. Mahadevan, C. Hu, and Z. Hu. A comprehensive review of digital twin — part 1: modeling and twinning enabling technologies. *Structural and Multidisciplinary Optimization*, 65(654):1–55, 2022.

[58] U. P. D. Ani, H. He, and A. Tiwari. Review of cybersecurity issues in industrial critical infrastructure: manufacturing in perspective. *Journal of Cyber Security Technology*, 1(1):32–74, 2017.

[59] M. Dietz, M. Vielberth, and G. Pernul. Integrating digital twin security simulations in the security operations center. In *Proceedings of the 15th International Conference on Availability, Reliability and Security*, pages 1–9, 2020.

[60] A. Giehl, N. Wiedermann, and S. Plaga. A framework to assess impacts of cyber attacks in manufacturing. In *2019 11th International Conference on Computer and Automation Engineering Proceedings*, pages 127–132, New York, NY, USA, 2019. ACM. URL: https://doi.org/10.1145/3313991.3314003, doi:10.1145/3313991.3314003.

[61] M. P. Heinl, A. Giehl, N. Wiedermann, S. Plaga, and F. Kargl. MERCAT: A metric for the evaluation and reconsideration of certificate authority trustworthiness. In *Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop*, pages 1–15, 2019.

[62] M. P. Heinl, A. Giehl, and L. Graif. Antipatterns regarding the application of cryptographic primitives by the example of ransomware. In *Proceedings of the 15th International Conference on Availability, Reliability and Security*, pages 1–10, 2020.

[63] S. Tatschner, F. Jarisch, A. Giehl, S. Plaga, and T. Newe. The stream exchange protocol: A secure and lightweight tool for decentralized connection establishment. *Sensors*, 21(15):4969, 2021.

[64] A. Giehl and P. Schneider. Sichere Prozesse für moderne Geschäftsmodelle. *Industrie 4.0 Management*, 34(1):55–59, 2018.

[65] A. Giehl and M. P. Heinl. Datenplattform nach dem Security-by-Design-Prinzip: Produktionsdaten sicher in der Cloud verarbeiten. *IT und Produktion*, (9):50–51, 2020.

[66] A. Giehl and M. P. Heinl. Drei Fragen an... Fraunhofer AISEC: Security als sich kontinuierlich weiterentwickelnder Prozess. Onlinebeitrag. *Computer und Automation*, 2020.

[67] A. Giehl. Kurzstatement. *DV-Dialog*, (1-2/2021):10, 2021.

[68] A. Giehl and M. P. Heinl. Cybersicherheit für kritische Infrastrukturen. *Linux Magazin*, 2:20–25, 2023.

[69] Esslinger, Ernst and Baier, Uwe and Krespach, Felix and Vinski, Marijo and Marschallek, Daniel and Nakic, Kai and Köhne, Hermann and Reers, Volker and Dukanovic, Sinisa and Bayarou, Kpatcha and Scheuermann, Dirk and Giehl, Alexander and Wiedermann, Norbert. D1.1-3: Sichere Kommunikation in der Fertigung. Technical report, IUNO Consortium, 2017.

[70] Giehl, Alexander and Bossert, Daniel and Brenner, Gerd and Shaabany, Ghaidaa and Bock, Hans-Peter and Blume, Marco and Schwarz, Reinhard and Frisch, Simon and Wang, Yübo. D2.1-1: Marktplatzsicherheit: Analyse vorhandener Technologien und Empfehlungen für den IUNO-Technologiedatenmarktplatz; Stand der Technik, Bedrohungs- und Risikoanalyse zum Demonstrator Technologiedatenmarktplatz. Technical report, IUNO Consortium, 2016.

[71] Hormann, Ricardo and Teuber, Stephan and Kerber, Marten and Schwandt, Sven and Sinner, Nadine and Rohr, Sebastian and Antón, Simon Duque and Fraunholz, Daniel and Pohl, Frederic and Giehl, Alexander and Plaga, Sven and Matthes, Manuel and Fischer, Kai and Heintel, Markus. D4.1-1: Stand der Technik, Bedrohungs- und Risikoanalyse zum Demonstrator Visueller Security Leitstand. Technical report, IUNO Consortium, 2016.

[72] Hormann, Ricardo and Teuber, Stephan and Nikelski, Sebastian and Larbig, Pedro and Neff, Sebastian and Scheuermann, Dirk and Giehl, Alexander and Plaga, Sven and Norouzian, Mohammad Reza and Malkis, Alexander. D4.2-1: Anomalieerkennung im Wertschöpfungsnetzwerk; Datenaustauschplattform für das Wertschöpfungsnetzwerk. Technical report, IUNO Consortium, 2017.

[73] Antón, Simon Duque and Fraunholz, Daniel and Lipps, Christoph and Angermeier, Daniel and Filipovic, Bartol and Giehl, Alexander and Obermaier, Johannes and Schneider, Peter and Schuster, Dieter and Wiedermann, Norbert and Rudolph, Manuel and Brandl, Hans and Schreiner, Florian and Fischer, Kai and Friedrich, Daniela and Bierbaumer, Bruno and Norouzian, Mohammad Reza and Müller, Johannes and Fischer, Günther and Neifer Wolfgang. AP7: Werkzeuge. Projektergebnisse. Technical report, IUNO Consortium, 2018.

[74] S. Cook, C. Bock, P. Rivett, T. Rutt, E. Seidewitz, B. Selic, and D. Tolbert. Unified modeling language (UML) version 2.5.1. Standard, Object Management Group (OMG), December 2017. URL: `https://www.omg.org/spec/UML/2.5.1`.

[75] L. Li, Y. Fan, M. Tse, and K.-Y. Lin. A review of applications in federated learning. *Computers & Industrial Engineering*, 149:106854, 2020.

[76] P. N. Stearns. *The industrial revolution in world history.* Routledge, 2020.

[77] J. Bardeen and W. H. Brattain. The transistor, a semi-conductor triode. *Physical Review*, 74(2):230, 1948.

[78] W. F. Brinkman, D. E. Haggan, and W. W. Troutman. A history of the invention of the transistor and where it will lead us. *IEEE Journal of Solid-State Circuits*, 32(12):1858–1865, 1997.

[79] M. Xu, J. M. David, S. H. Kim, et al. The fourth industrial revolution: Opportunities and challenges. *International journal of financial research*, 9(2):90–95, 2018.

[80] M. Hollender. *Collaborative process automation systems.* ISA, 2010.

[81] T. Meudt, M. Pohl, and J. Metternich. Die automatisierungspyramide-ein literaturüberblick. 2017.

[82] Enterprise-Control System Integration – Part 1: Models and Terminology. Standard, The International Society of Automation, Research Triangle Park, North Carolina, USA, 2010.

[83] Enterprise-control system integration - Part 1: Models and terminology (IEC 62264-1:2013); German version EN 62264-1:2013. Standard, Deutsches Institut für Normung, Berlin, July 2014.

[84] Y. Wang, G. Wang, and R. Anderl. Generic procedure model to introduce Industrie 4.0 in small and medium-sized enterprises. In *Proceedings of the World Congress on Engineering and Computer Science*, volume 2, pages 1–6, 2016.

[85] I. Bojanova. The digital revolution: What's on the horizon? *IT Professional*, 16(1):8–12, 2014.

[86] International Labour Organization. Preparing for the future of work: National policy responses in ASEAN +6. ANNEX I Labour markets and technological change. Technical report, International Labour Organization, 2019.

[87] European Commission. *Digital Transformation Monitor Country: Portugal "Industria 4.O"*. European Commission, 2017.

[88] W. Naudé, A. Surdej, and M. Cameron. Ready for Industry 4.0? The Case of Central and Eastern Europe. In *Industry 4.0 and Engineering for a Sustainable Future*, pages 153–175. Springer, 2019.

[89] Ireland's Industry 4.0 Strategy 2020-2025. Supporting the digital transformation of the manufacturing sector and its supply chain. Technical report, Government of Ireland - Department of Business, Enterprise and Innovation, 2019.

[90] K. Schwab. *The fourth industrial revolution*. Currency, 2017.

[91] E. Negri, L. Fumagalli, and M. Macchi. A review of the roles of digital twin in cps-based production systems. *Procedia manufacturing*, 11:939–948, 2017.

[92] R. Buttrick. *The project workout: a toolkit for reaping the rewards from all your business projects*. Financial Times/Prentice Hall, 2000.

[93] C. Labuschagne and A. C. Brent. Sustainable project life cycle management: the need to integrate life cycles in the manufacturing sector. *International Journal of Project Management*, 23(2):159–168, 2005.

[94] M. Shafto, M. Conroy, R. Doyle, E. Glaessgen, C. Kemp, J. LeMoigne, and L. Wang. Modeling, simulation, information technology & processing roadmap. *National Aeronautics and Space Administration*, 32(2012):1–38, 2012.

[95] A. Bécue, E. Maia, L. Feeken, P. Borchers, and I. Praça. A new concept of digital twin supporting optimization and resilience of factories of the future. *Applied Sciences*, 10(13):4482, 2020.

[96] J. Lee, E. Lapira, B. Bagheri, and H.-a. Kao. Recent advances and trends in predictive manufacturing systems in big data environment. *Manufacturing letters*, 1(1):38–41, 2013.

[97] R. R. Dipert. Other-than-Internet (OTI) cyberwarfare: challenges for ethics, law, and policy. *Journal of Military Ethics*, 12(1):34–53, 2013.

[98] K. Thakur, M. L. Ali, N. Jiang, and M. Qiu. Impact of cyber-attacks on critical infrastructure. In *2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS)*, pages 183–186. IEEE, 2016.

[99] T. Reed. *At the abyss: an insider's history of the Cold War*. Presidio Press, 2005.

[100] P. J. DeSouza. The soviet gas pipeline incident: Extension of collective security responsibilities to peacetime commercial trade. *Yale J. Int'l L.*, 10:92, 1984.

[101] S. Colbourn. An interpreter or two: defusing nato's siberian pipeline dispute, 1981–1982. *Journal of Transatlantic Studies*, 18:131–151, 2020.

[102] J. Slay and M. Miller. Lessons learned from the maroochy water breach. In *International conference on critical infrastructure protection*, pages 73–82. Springer, 2007.

[103] K. E. Hemsley, E. Fisher, et al. History of industrial control system cyber incidents. Technical report, Idaho National Lab.(INL), Idaho Falls, ID (United States), 2018.

[104] M. Rocchetto and N. O. Tippenhauer. On attacker models and profiles for cyber-physical systems. In *European Symposium on Research in Computer Security*, pages 427–449. Springer, 2016.

[105] M. Rocchetto and N. O. Tippenhauer. Cpdy: extending the dolev-yao attacker with physical-layer interactions. In *Formal Methods and Software Engineering: 18th International Conference on Formal Engineering Methods, ICFEM 2016, Tokyo, Japan, November 14-18, 2016, Proceedings 18*, pages 175–192. Springer, 2016.

[106] P. C. Reich, S. Weinstein, C. Wild, and A. S. Cabanlong. Cyber warfare: a review of theories, law, policies, actual incidents–and the dilemma of anonymity. *European Journal of Law and Technology*, 1(2):1–58, 2010.

[107] R. Langner. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security & Privacy*, 9(3):49–51, 2011.

[108] R. Langner. To kill a centrifuge: A technical analysis of what stuxnet's creators tried to achieve. White paper, The Langner Group, 2013.

[109] P. Schneider. Do's and don'ts of distributed intrusion detection for industrial network topologies. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 3222–3231. IEEE, 2019.

[110] B. A. S. Al-rimy, M. A. Maarof, and S. Z. M. Shaid. Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions. *Computers & Security*, 74:144–166, 2018.

[111] G. Salviotti, N. Abbatemarco, L. M. De Rossi, and K. Bjoernland. Understanding the role of leadership competencies in cyber crisis management: A case study. *Proceedings of the 56th Hawaii International Conference on System Sciences*, pages 6068–6077, 2023.

[112] B. Oestereich and A. Scheithauer. *Analyse und Design mit der UML 2.5*. Oldenbourg Wissenschaftsverlag, 2013.

[113] G. Booch, I. Jacobson, J. Rumbaugh, et al. The unified modeling language. *Unix Review*, 14(13):5, 1996.

[114] Information technology — open distributed processing — unified modeling language (uml) version 1.4.2. Standard, International Organization for Standardization, ISO Central Secretariat, Geneva, Switzerland, 2005.

[115] J. Jürjens. Umlsec: Extending uml for secure systems development. In *UML 2002—The Unified Modeling Language: Model Engineering, Concepts, and Tools 5th International Conference Dresden, Germany, September 30–October 4, 2002 Proceedings*, pages 412–425. Springer, 2002.

[116] J. Jürjens. *Secure systems development with UML*. Springer Science & Business Media, 2005.

[117] T. Lodderstedt, D. Basin, and J. Doser. Secureuml: A uml-based modeling language for model-driven security. In *UML 2002—The Unified Modeling*

Language: Model Engineering, Concepts, and Tools 5th International Conference Dresden, Germany, September 30–October 4, 2002 Proceedings, pages 426–441. Springer, 2002.

[118] M. U. Khan. Representing security specifications in uml state machine diagrams. *Procedia Computer Science*, 56:453–458, 2015.

[119] K. Barker, M. Askari, M. Banerjee, K. Ghazinour, B. Mackas, M. Majedi, S. Pun, and A. Williams. A data privacy taxonomy. In *British National Conference on Databases*, pages 42–54. Springer, 2009.

[120] R. L. Finn, D. Wright, and M. Friedewald. Seven types of privacy. In *European Data Protection: Coming of Age*, pages 3–32. Springer, 2013.

[121] B. C. Fung, K. Wang, R. Chen, and P. S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Surveys (Csur)*, 42(4):1–53, 2010.

[122] C. Eckert. *IT-Sicherheit: Konzepte-Verfahren-Protokolle*. Walter de Gruyter, 2013.

[123] A. Majeed and S. Lee. Anonymization techniques for privacy preserving data publishing: A comprehensive survey. *IEEE access*, 9:8512–8545, 2020.

[124] L. Demir, A. Kumar, M. Cunche, and C. Lauradoux. The pitfalls of hashing for privacy. *IEEE Communications Surveys & Tutorials*, 20(1):551–565, 2017.

[125] V. Janmey and P. L. Elkin. Re-identification risk in hipaa de-identified datasets: The mva attack. In *AMIA Annual Symposium Proceedings*, volume 2018, page 1329. American Medical Informatics Association, 2018.

[126] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[127] L. Sweeney. k-anonymity: A model for protecting privacy. *International journal of uncertainty, fuzziness and knowledge-based systems*, 10(05):557–570, 2002.

[128] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek, and H. V. Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469, 2020.

[129] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 202–210. ACM, 2003.

[130] C. Dwork, A. Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[131] C. Dwork. Differential privacy. *Encyclopedia of Cryptography and Security*, pages 338–340, 2011.

[132] C. Dwork, G. N. Rothblum, and S. Vadhan. Boosting and differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 51–60. IEEE, 2010.

[133] G. Van Blarkom, J. J. Borking, and J. E. Olk. Handbook of privacy and privacy-enhancing technologies. *Privacy Incorporated Software Agent (PISA) Consortium, The Hague*, 198:14, 2003.

[134] A. Fahad, Z. Tari, A. Almalawi, A. Goscinski, I. Khalil, and A. Mahmood. PPFSCADA: Privacy preserving framework for SCADA data publishing. *Future generation computer systems*, 37:496–511, 2014.

[135] S.-C. Cha, T.-Y. Hsu, Y. Xiang, and K.-H. Yeh. Privacy enhancing technologies in the internet of things: Perspectives and challenges. *IEEE Internet of Things Journal*, 6(2):2159–2187, 2018.

[136] P. Hustinx. Privacy by design: delivering the promises. *Identity in the Information Society*, 3(2):253–255, 2010.

[137] F. Mosaiyebzadeh, S. Pouriyeh, R. M. Parizi, Q. Z. Sheng, M. Han, L. Zhao, G. Sannino, C. M. Ranieri, J. Ueyama, and D. M. Batista. Privacy-enhancing technologies in federated learning for the internet of healthcare things: A survey. *Electronics*, 12(12):2703, 2023.

[138] D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[139] A. Cavoukian. *Privacy by design, take the challenge.* desLibris, 2009.

[140] C. Meyer. *Years of SSL/TLS Research: An analysis of the Internet's security foundation.* PhD thesis, PhD thesis, Ruhr-University Bochum, 2014.

[141] S. Zuboff. Big other: surveillance capitalism and the prospects of an information civilization. *Journal of Information Technology*, 30(1):75–89, 2015.

[142] S. A. Tovino. The hipaa privacy rule and the eu gdpr: illustrative comparisons. *Seton Hall L. Rev.*, 47:973, 2016.

[143] J. Scheibner, J. L. Raisaro, J. R. Troncoso-Pastoriza, M. Ienca, J. Fellay, E. Vayena, and J.-P. Hubaux. Revolutionizing medical data sharing using advanced privacy-enhancing technologies: technical, legal, and ethical synthesis. *Journal of medical Internet research*, 23(2):e25120, 2021.

[144] M. U. Hassan, M. H. Rehmani, and J. Chen. Differential privacy techniques for cyber physical systems: a survey. *IEEE Communications Surveys & Tutorials*, 22(1):746–789, 2020.

[145] M. A. Husnoo, A. Anwar, R. K. Chakrabortty, R. Doss, and M. J. Ryan. Differential privacy for iot-enabled critical infrastructure: A comprehensive survey. *IEEE Access*, 9:153276–153304, 2021.

[146] B. Jiang, J. Li, G. Yue, and H. Song. Differential privacy for industrial internet of things: Opportunities, applications, and challenges. *IEEE Internet of Things Journal*, 8(13):10430–10451, 2021.

[147] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[148] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[149] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, D. Niyato, and H. V. Poor. Federated learning for industrial internet of things in future industries. *IEEE Wireless Communications*, 28(6):192–199, 2021.

[150] P. Boobalan, S. P. Ramu, Q.-V. Pham, K. Dev, S. Pandya, P. K. R. Maddikunta, T. R. Gadekallu, and T. Huynh-The. Fusion of federated learning and industrial internet of things: A survey. *Computer Networks*, 212:109048, 2022.

[151] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang. Blockchain and federated learning for privacy-preserved data sharing in industrial iot. *IEEE Transactions on Industrial Informatics*, 16(6):4177–4186, 2019.

[152] Y. Liu, R. Zhao, J. Kang, A. Yassine, D. Niyato, and J. Peng. Towards communication-efficient and attack-resistant federated edge learning for industrial internet of things. *ACM Transactions on Internet Technology (TOIT)*, 22(3):1–22, 2021.

[153] A. El Ouadrhiri and A. Abdelhadi. Differential privacy for deep and federated learning: A survey. *IEEE Access*, 10:22359–22380, 2022.

[154] A. Appari and M. E. Johnson. Information security and privacy in healthcare: current state of research. *International journal of Internet and enterprise management*, 6(4):279–314, 2010.

[155] A. Gkoulalas-Divanis, G. Loukides, and J. Sun. Toward smarter healthcare: Anonymizing medical data to support research studies. *IBM Journal of Research and Development*, 58(1):9–1, 2014.

[156] M. Conti, D. Donadel, and F. Turrin. A survey on industrial control system testbeds and datasets for security research. *IEEE Communications Surveys & Tutorials*, 23(4):2248–2294, 2021.

[157] C. Davis, J. Tate, H. Okhravi, C. Grier, T. J. Overbye, and D. Nicol. Scada cyber security testbed development. In *2006 38th North American Power Symposium*, pages 483–488. IEEE, 2006.

[158] M. Ciavotta, M. Alge, S. Menato, D. Rovere, and P. Pedrazzoli. A microservice-based middleware for the digital factory. *Procedia manufacturing*, 11:931–938, 2017.

[159] Y. Cai, B. Starly, P. Cohen, and Y.-S. Lee. Sensor data and information fusion to construct digital-twins virtual machine tools for cyber-physical manufacturing. *Procedia manufacturing*, 10:1031–1042, 2017.

[160] C. Gehrmann and M. Gunnarsson. A digital twin based industrial automation and control system security architecture. *IEEE Transactions on Industrial Informatics*, 16(1):669–680, 2019.

[161] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on information theory*, 29(2):198–208, 1983.

[162] V. Damjanovic-Behrendt and W. Behrendt. An open source approach to the design and implementation of digital twins for smart manufacturing. *International Journal of Computer Integrated Manufacturing*, 32(4-5):366–384, 2019.

[163] E. O'Connell, W. O'Brien, M. Bhattacharya, D. Moore, and M. Penica. Digital twins: Enabling interoperability in smart manufacturing networks. In *Telecom*, volume 4, pages 265–278. MDPI, 2023.

[164] R. Candell, K. Stouffer, and D. Anand. A cybersecurity testbed for industrial control systems. In *Process Control and Safety Symposium, International Society of Automation, Houston, TX*, pages 1–16, 2014.

[165] S. D. Schaber, D. I. Gerogiorgis, R. Ramachandran, J. M. Evans, P. I. Barton, and B. L. Trout. Economic analysis of integrated continuous and batch pharmaceutical manufacturing: a case study. *Industrial & Engineering Chemistry Research*, 50(17):10083–10092, 2011.

[166] and others. *Modern control engineering*, volume 5. Prentice hall Upper Saddle River, NJ, 2010.

[167] D. Witsch and B. Vogel-Heuser. Plc-statecharts: An approach to integrate uml-statecharts in open-loop control engineering–aspects on behavioral semantics and model-checking. *IFAC Proceedings Volumes*, 44(1):7866–7872, 2011.

[168] C. Secchi, M. Bonfe, and C. Fantuzzi. On the use of uml for modeling mechatronic systems. *IEEE Transactions on Automation Science and Engineering*, 4(1):105–113, 2007.

[169] K. C. Thramboulidis. Using uml in control and automation: a model driven approach. In *2nd IEEE International Conference on Industrial Informatics, 2004. INDIN'04. 2004*, pages 587–593. IEEE, 2004.

[170] G. F. Schneider, G. A. Peßler, and W. Terkaj. Knowledge-based conversion of finite state machines in manufacturing automation. *Procedia manufacturing*, 28:189–194, 2019.

[171] P. Muñoz, J. Troya, and A. Vallecillo. Using uml and ocl models to realize high-level digital twins. In *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, pages 212–220. IEEE, 2021.

[172] M. Azangoo, A. Taherkordi, and J. O. Blech. Digital twins for manufacturing using uml and behavioral specifications. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 1035–1038. IEEE, 2020.

[173] E. Brusa. Digital twin: Toward the integration between system design and rams assessment through the model-based systems engineering. *IEEE Systems Journal*, 15(3):3549–3560, 2020.

[174] D. Lehner, S. Wolny, A. Mazak-Huemer, and M. Wimmer. Towards a reference architecture for leveraging model repositories for digital twins. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 1077–1080. IEEE, 2020.

[175] I. Siveroni, A. Zisman, and G. Spanoudakis. A uml-based static verification framework for security. *Requirements engineering*, 15:95–118, 2010.

[176] K. Böttinger. *Fuzzing with Stochastic Feedback Processes*. PhD thesis, Technische Universität München, 2019.

[177] S. Uludag, S. Zeadally, and M. Badra. Techniques, taxonomy, and challenges of privacy protection in the smart grid. In *Privacy in a Digital, Networked World*, pages 343–390. Springer, 2015.

[178] R. Chen, B. C. Fung, N. Mohammed, B. C. Desai, and K. Wang. Privacy-preserving trajectory data publishing by local suppression. *Information Sciences*, 231:83–97, 2013.

[179] N. Notario, A. Crespo, Y.-S. Martín, J. M. Del Alamo, D. Le Métayer, T. Antignac, A. Kung, I. Kroener, and D. Wright. Pripare: integrating privacy best practices into a privacy engineering methodology. In *2015 IEEE Security and Privacy Workshops*, pages 151–158. IEEE, 2015.

[180] E. Schicker. *Datenbanken und SQL*. Springer, 2017.

[181] E. Shi, T. H. Chan, E. Rieffel, R. Chow, and D. Song. Privacy-preserving aggregation of time-series data. In *Proc. NDSS*, volume 2, pages 1–17. Citeseer, 2011.

[182] C. Yin, J. Xi, R. Sun, and J. Wang. Location privacy protection based on differential privacy strategy for big data in industrial internet of things. *IEEE Transactions on Industrial Informatics*, 14(8):3628–3636, 2017.

[183] B. Schmucker, M. Busch, R. Kleinwort, and F. Zäh, M. Edge- und cloudbasierte Prozessüberwachung: Systemidentifikation und Regelung von Werkzeugmaschinen unter Berücksichtigung der Datensicherheit. In *WT WERKSTATTSTECHNIK*, volume 110 of *3*, pages 113–118. 2020.

[184] Y. Wang, Z. Huang, S. Mitra, and G. E. Dullerud. Differential privacy in linear distributed control systems: Entropy minimizing mechanisms and performance tradeoffs. *IEEE Transactions on Control of Network Systems*, 4(1):118–130, 2017.

[185] T. Wang, Y. Mei, W. Jia, X. Zheng, G. Wang, and M. Xie. Edge-based differential privacy computing for sensor–cloud systems. *Journal of Parallel and Distributed computing*, 136:75–85, 2020.

[186] Q. Hu, R. Chen, H. Yang, and S. Kumara. Privacy-preserving data mining for smart manufacturing. *Smart and Sustainable Manufacturing Systems*, 4(2):99–120, 2020.

[187] P. C. M. Arachchige, P. Bertok, I. Khalil, D. Liu, S. Camtepe, and M. Atiquzzaman. A trustworthy privacy preserving framework for machine learning in industrial iot systems. *IEEE Transactions on Industrial Informatics*, 16(9):6092–6102, 2020.

[188] S. Jamil, M. Rahman, and Fawad. A comprehensive survey of digital twins and federated learning for industrial internet of things (iiot), internet of vehicles (iov) and internet of drones (iod). *Applied System Innovation*, 5(3):56, 2022.

[189] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang. Communication-efficient federated learning for digital twin edge networks in industrial iot. *IEEE Transactions on Industrial Informatics*, 17(8):5709–5718, 2021.

[190] C. Herley and P. C. Van Oorschot. Sok: Science, security and the elusive goal of security as a scientific pursuit. In *2017 IEEE symposium on security and privacy (SP)*, pages 99–120. IEEE, 2017.

[191] Y. Pan, J. White, D. C. Schmidt, A. Elhabashy, L. Sturm, J. Camelio, and C. Williams. Taxonomies for reasoning about cyber-physical attacks in IoT-based manufacturing systems. *International Journal of Interactive Multimedia & Artificial Intelligence*, 4(3), 2017.

[192] T. Hutzelmann, S. Banescu, and A. Pretschner. A comprehensive attack and defense model for the automotive domain. *SAE International Journal of Transportation Cybersecurity and Privacy*, 2(11-02-01-0001):5–20, 2019.

[193] Dukanovic, Sinisa and Matthess, Manuel and Schwarz, Reinhard and Frisch, Simon and Fischer, Kai and Borisov, Alexander. D6.1-1: Bedrohungsmodell für Wertschöpfungsnetzwerke der Industrie 4.0. Technical report, IUNO Consortium, 2016.

[194] Dukanovic, Sinisa and Matthess, Manuel and Schwarz, Reinhard and Frisch, Simon and Fischer, Kai and Lipps, Cristoph. D6.1-2: Risikomodell für Wertschöpfungsnetzwerke der Industrie 4.0. Technical report, IUNO Consortium, 2016.

[195] J. Slay and M. Miller. *Lessons learned from the maroochy water breach.* Springer, 2008.

[196] R. Candell, T. Zimmerman, and K. Stouffer. An industrial control system cybersecurity performance testbed. Technical report, National Institute of Standards and Technology. NISTIR, 2015.

[197] S. Cox, J. M. Tomás, A. Cheyne, and A. Oliver. Safety culture: The prediction of commitment to safety in the manufacturing industry. *British Journal of management*, 9:3–11, 1998.

[198] T. Bodström and T. Hämäläinen. A novel method for detecting apt attacks by using ooda loop and black swan theory. In *Computational Data and Social Networks: 7th International Conference, CSoNet 2018, Shanghai, China, December 18–20, 2018, Proceedings 7*, pages 498–509. Springer, 2018.

[199] D. Duggan, M. Berg, J. Dillinger, and J. Stamp. Penetration testing of industrial control systems. *Sandia national laboratories*, page 7, 2005.

[200] A. Cook, L. Maglaras, R. Smith, and H. Janicke. Managing incident response in the industrial internet of things. *International Journal of Internet Technology and Secured Transactions*, 8(2):251–276, 2018.

[201] P. H. Meland, Y. F. F. Bayoumy, and G. Sindre. The ransomware-as-a-service economy within the darknet. *Computers & Security*, 92:101762, 2020.

[202] H. Turner, J. White, J. A. Camelio, C. Williams, B. Amos, and R. Parker. Bad parts: Are our manufacturing systems at risk of silent cyberattacks? *IEEE Security & Privacy*, 13(3):40–47, 2015.

[203] K. Arbanas and N. Žajdela Hrustek. Key success factors of information systems security. *Journal of information and organizational sciences*, 43(2):131–144, 2019.

[204] S. Bandyopadhyay and R. Bhattacharya. *Discrete and continuous simulation: theory and practice*. CRC Press, 2014.

[205] N. Suzić, B. Stevanov, I. Ćosić, Z. Anišić, and N. Sremčev. Customizing products through application of group technology: A case study of furniture manufacturing. *Strojniški vestnik-Journal of Mechanical Engineering*, 58(12):724–731, 2012.

[206] J. Gradišek, A. Baus, E. Govekar, F. Klocke, and I. Grabec. Automatic chatter detection in grinding. *International Journal of Machine Tools and Manufacture*, 43(14):1397–1403, 2003.

[207] K. K. Singh, R. Singh, and V. Kartik. Comparative study of chatter detection methods for high-speed micromilling of ti6al4v. *Procedia Manufacturing*, 1:593–606, 2015.

[208] A. N. Novak and M. O. Vilceanu. "the internet is not pleased": twitter and the 2017 equifax data breach. *The Communication Review*, 22(3):196–221, 2019.

[209] C. Alcaraz and J. Lopez. Digital twin: A comprehensive survey of security threats. *IEEE Communications Surveys & Tutorials*, 24(3):1475–1503, 2022.

[210] P. Koller. Damit der Digital Twin nicht zum bösen Zwilling wird. Onlinebeitrag. *all-electronics*, 2022.

[211] OPC UA Companion Specification. Standard, MTConnect Institute, McLean, Virginia, USA, June 2019.

[212] OPC Unified Architecture – Part 1: Overview and Concepts. Standard, International Electrotechnical Commission, Geneva, CH, February 2010.

[213] M. Helu, T. Hedberg Jr, and A. B. Feeney. Reference architecture to integrate heterogeneous manufacturing systems for the digital thread. *CIRP journal of manufacturing science and technology*, 19:191–195, 2017.

[214] B. Saenz de Ugarte, A. Artiba, and R. Pellerin. Manufacturing execution system–a literature review. *Production planning and control*, 20(6):525–539, 2009.

[215] N. P. Mahalik and A. N. Nambiar. Trends in food packaging and manufacturing systems and technology. *Trends in food science & technology*, 21(3):117–128, 2010.

[216] P. Schneider and K. Böttinger. High-performance unsupervised anomaly detection for cyber-physical system networks. In *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy*, pages 1–12, 2018.

[217] Z. DeSmit, A. E. Elhabashy, L. J. Wells, and J. A. Camelio. An approach to cyber-physical vulnerability assessment for intelligent manufacturing systems. *Journal of Manufacturing Systems*, 43:339–351, 2017.

[218] M. D. Richardson, P. A. Lemoine, W. E. Stephens, and R. E. Waller. Planning for cyber security in schools: The human factor. *Educational Planning*, 27(2):23–39, 2020.

[219] M. Doiro, F. Fernández, M. Félix, and G. Santos. Machining operations for components in kitchen furniture: A comparison between two management systems. *Procedia Manufacturing*, 41:10–17, 2019.

[220] G. Dokter, S. Andersson, L. Thuvander, and U. Rahe. Co-creation—a facilitator for circular economy implementation? a case study in the kitchen industry. *Proceedings of the PLATE Product Lifetimes and the Environment, Berlin, Germany*, pages 18–20, 2019.

[221] L. Li, A. Haghighi, and Y. Yang. A novel 6-axis hybrid additive-subtractive manufacturing process: Design and case studies. *Journal of Manufacturing Processes*, 33:150–160, 2018.

[222] H. Paris, G. Mandil, et al. Process planning for combined additive and subtractive manufacturing technologies in a remanufacturing context. *Journal of Manufacturing Systems*, 44:243–254, 2017.

[223] A. Ekelhart, E. Kiesling, B. Grill, C. Strauss, and C. Stummer. Integrating attacker behavior in IT security analysis: a discrete-event simulation approach. *Information Technology and Management*, 16(3):221–233, 2015.

[224] Q. Qassim, N. Jamil, I. Z. Abidin, M. E. Rusli, S. Yussof, R. Ismail, F. Abdullah, N. Ja'afar, H. C. Hasan, and M. Daud. A survey of scada testbed implementation approaches. *Indian Journal of Science and Technology*, 10(26):1–8, 2017.

[225] C. Queiroz, A. Mahmood, and Z. Tari. SCADASim—a framework for building SCADA simulations. *IEEE Transactions on Smart Grid*, 2(4):589–597, 2011.

[226] S. N. T.-c. Chiueh and S. Brook. A survey on virtualization technologies. *Rpe Report*, 142, 2005.

[227] F. Sauer, M. Niedermaier, S. Kießling, and D. Merli. Licster–a low-cost ics security testbed for education and research. *arXiv preprint arXiv:1910.00303*, 2019.

[228] International Electrotechnical Commission. IEC 62443 Technical Specification, Edition 1.0, July 2009. Technical Specification 62443-1-1, International Electrotechnical Commission, Geneva, CH, July 2009.

[229] S. Almeaibed, S. Al-Rubaye, A. Tsourdos, and N. P. Avdelidis. Digital twin analysis to promote safety and security in autonomous vehicles. *IEEE Communications Standards Magazine*, 5(1):40–46, 2021.

[230] B. Sousa, M. Arieiro, V. Pereira, J. Correia, N. Lourenço, and T. Cruz. Elegant: Security of critical infrastructures with digital twins. *IEEE Access*, 9:107574–107588, 2021.

[231] M. Chetan, S. Yao, and D. T. Griffith. Multi-fidelity digital twin structural model for a sub-scale downwind wind turbine rotor blade. *Wind Energy*, 24(12):1368–1387, 2021.

[232] F. Akbarian, E. Fitzgerald, and M. Kihl. Intrusion detection in digital twins for industrial control systems. In *2020 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 1–6. IEEE, 2020.

[233] A. Stefanov and C.-C. Liu. Cyber-Physical system security and impact analysis. *IFAC Proceedings Volumes*, 47(3):11238–11243, 2014.

[234] J. Milošević, T. Tanaka, H. Sandberg, and K. H. Johansson. Exploiting submodularity in security measure allocation for industrial control systems. In

Proceedings of the 1st ACM Workshop on the Internet of Safe Things, pages 64–69. ACM, 2017.

[235] Road vehicles – Cybersecurity Engineering. Standard, International Organization for Standardization/ Society of Automotive Engineers, Geneva, CH, 2021.

[236] M. O. Ball, C. J. Colbourn, and J. S. Provan. Network reliability. *Handbooks in operations research and management science*, 7:673–762, 1995.

[237] T. Cruz, P. Simoes, and E. Monteiro. Virtualizing programmable logic controllers: Toward a convergent approach. *IEEE Embedded Systems Letters*, 8(4):69–72, 2016.

[238] P. Grim, R. Rosenberger, A. Rosenfeld, B. Anderson, and R. E. Eason. How simulations fail. *Synthese*, 190(12):2367–2390, 2013.

[239] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. ROS: an open-source Robot Operating System. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[240] M. Quigley, B. Gerkey, and W. D. Smart. *Programming Robots with ROS: a practical introduction to the Robot Operating System.* O'Reilly Media, Inc., 2015.

[241] L. Joseph and J. Cacace. *Mastering ROS for Robotics Programming: Design, build, and simulate complex robots using the Robot Operating System.* Packt Publishing Ltd, 2018.

[242] A. Koubâa. *Robot Operating System (ROS).*, volume 1. Springer, 2019.

[243] ABB Robotics. Datenblatt ABB IRB 120 Industrieroboter. `https://search.abb.com/library/Download.aspx?DocumentID=ROB0149DE_A`, 2019. Accessed on June 7, 2023.

[244] J. P. Kharoufeh, S. M. Cox, and M. E. Oxley. Reliability of manufacturing equipment in complex environments. *Annals of Operations Research*, 209(1):231–254, 2013.

[245] Dienkhah, Sahand and Dukanovic, Sinisa and Erhardt, Sascha and Giehl, Alexander and Mäser, Nils and Niemczik, Lars and Reti, Daniel and Schulze, Jan-Phillip . D4.2: Getestete und funktionsfähige Lösungen. Technical report, IUNO Insec Consortium, 2022.

[246] Y. Feng, Y. Yao, and N. Sadeh. A design space for privacy choices: Towards meaningful privacy control in the internet of things. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–16. ACM, 2021.

[247] Antón, Simon Duque and Dienkhah, Sahand and Dukanovic, Sinisa and Giehl, Alexander and Kern, Alexander and Schneider, Daniel and Fraunholz, Daniel and Laabs, Martin and Plaga, Sven and Schneider, Peter and Feuchtmüller, Sven. D2.1: Sicherheitslevel und abgeleitete Migrationspfade. Technical report, IUNO Insec Consortium, 2019.

[248] A. Ključnikov, L. Mura, and D. Sklenár. Information security management in smes: factors of success. *Entrepreneurship and Sustainability Issues*, 6(4):2081, 2019.

[249] M. Seregni, D. Opresnik, C. Zanetti, M. Taisch, and F. Voorhorst. Mini factory: a successful model for european furniture industry? In *IFIP International Conference on Advances in Production Management Systems*, pages 571–578. Springer, 2014.

[250] C. Zanetti, M. Seregni, M. Bianchini, and M. Taisch. A production system model for mini-factories and last mile production approach. In *2015 IEEE 1st International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, pages 451–456. IEEE, 2015.

[251] A. Barni, D. Corti, P. Pedrazzoli, D. Rovere, and G. Lucisano. Mini-factories for close-to-customer manufacturing of customized furniture: from concept to real demo. *Procedia Manufacturing*, 11:854–862, 2017.

[252] M. Zaeh, F. Schnoes, B. Obst, and D. Hartmann. Combined offline simulation and online adaptation approach for the accuracy improvement of milling robots. *CIRP Annals*, 69(1):337–340, 2020.

[253] Analytical Prediction of Stability Lobes in Milling. *CIRP Annals - Manufacturing Technology*, 44(1):357–362, 1995. doi:10.1016/S0007-8506(07)62342-7.

[254] Y. ALTINTAS. Analytical Prediction of Three Dimensional Chatter Stability in Milling. *JSME International Journal Series C*, 44(3):717–723, 2001. doi:10.1299/jsmec.44.717.

[255] G. W. Hart. Nonintrusive appliance load monitoring. *Proceedings of the IEEE*, 80(12):1870–1891, 1992.

[256] W. Shi and S. Dustdar. The promise of edge computing. *Computer*, 49(5):78–81, 2016.

[257] OPC UA for Machine Tools – Part 1: Machine Monitoring and Job Overview. Standard, VDMA, Frankfurt, Germany, 2020.

[258] T. Zhu, P. Xiong, G. Li, W. Zhou, and S. Y. Philip. Differentially private model publishing in cyber physical systems. *Future generation computer systems*, 108:1297–1306, 2020.

[259] L. Fan and L. Xiong. An adaptive approach to real-time aggregate monitoring with differential privacy. *IEEE Transactions on knowledge and data engineering*, 26(9):2094–2106, 2013.

[260] C. Zhao, S. Zhao, M. Zhao, Z. Chen, C.-Z. Gao, H. Li, and Y.-a. Tan. Secure multi-party computation: theory, practice and applications. *Information Sciences*, 476:357–372, 2019.

[261] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Computing Surveys (Csur)*, 51(4):1–35, 2018.

[262] H.-C. Möhring, P. Wiederkehr, K. Erkorkmaz, and Y. Kakinuma. Self-optimizing machining systems. *CIRP Annals*, 69(2):740–763, 2020.

[263] F. Schnoes and M. Zaeh. Model-based planning of machining operations for industrial robots. *Procedia CIRP*, 82:497–502, 2019.

[264] N. Holohan, S. Braghin, P. Mac Aonghusa, and K. Levacher. Diffprivlib: the IBM differential privacy library. arXiv preprint arXiv:1907.02444, 2019.

[265] S. AG. *SINUMERIK - SINUMERIK 840D sl - NC-Variable und Nahtstellensignale - Listenhandbuc*. Siemens AG, 2015.

[266] M. E. Gursoy, A. Tamersoy, S. Truex, W. Wei, and L. Liu. Secure and utility-aware data collection with condensed local differential privacy. *IEEE Transactions on Dependable and Secure Computing*, pages 1–13, 2019.

[267] S.-J. Shin, S.-H. Suh, and I. Stroud. Reincarnation of G-code based part programs into STEP-NC for turning applications. *Computer-Aided Design*, 39(1):1–16, 2007.

[268] J. Giraldo, A. Cardenas, and M. Kantarcioglu. Security and privacy trade-offs in cps by leveraging inherent differential privacy. In *2017 IEEE Conference on Control Technology and Applications (CCTA)*, pages 1313–1318. IEEE, 2017.

[269] J. Giraldo, D. Urbina, A. Cardenas, J. Valente, M. Faisal, J. Ruths, N. O. Tippenhauer, H. Sandberg, and R. Candell. A survey of physics-based attack detection in cyber-physical systems. *ACM Computing Surveys (CSUR)*, 51(4):1–36, 2018.