Towards Secure Collaborative AI Service Chains

Vida Ahmadi Mehri

Blekinge Institute of Technology licentiate dissertation series
No: 2019:XX

Towards Secure Collaborative AI Service Chains

Vida Ahmadi Mehri

Licentiate Dissertation in
Telecommunication Systems

Faculty of Computer Sciences
Department of Computer Science
Blekinge Institute of Technology
SWEDEN

Faculty of Computer Sciences
Department of Computer Science
Publisher: Blekinge Institute of Technology,
SE-371 79 Karlskrona, Sweden
Printed by Lenanders, Kalmar, Sweden 2019XX
ISBN 123-21332-13423-123
ISSN 1650-2140

“If we knew what we were doing, it wouldn’t be called research. Would it? “

Albert Einstein

Abstract

At present, Artificial Intelligence (AI) systems have been adopted in many different domains such as healthcare, robotics, automotive, telecommunication systems, security, and finance for integrating intelligence in their services and applications. The intelligent personal assistant such as Siri and Alexa are examples of AI systems making an impact on our daily lives. Since many AI systems are data-driven systems, they require large volumes of data for training and validation, advanced algorithms, computing power and storage in their development process. Collaboration in the AI development process (AI engineering process) will reduce cost and time for the AI applications in the market. However, collaboration introduces the concern of privacy and piracy of intellectual properties, which can be caused by the actors who collaborate in the engineering process.

This work investigates the non-functional requirements, such as privacy and security, for enabling collaboration in AI service chains. It proposes an architectural design approach for collaborative AI engineering and explores the concept of the pipeline (service chain) for chaining AI functions. In order to enable controlled collaboration between AI artefacts [1] in a pipeline, this work makes use of virtualisation technology to define and implement Virtual Premises (VPs), which act as protection wrappers for AI pipelines. A VP is a virtual policy enforcement point for a pipeline and requires access permission and authenticity for each element in a pipeline before the pipeline can be used.

Furthermore, the proposed architecture is evaluated in use-case approach that enables quick detection of design flaw during the initial stage of implementation. To evaluate the security level and compliance with security requirements, threat modeling was used to identify potential threats and vulnerabilities of the system and analyses their possible effects. The output of threat modeling was used to define countermeasure to threats related to unauthorised access and execution of AI artefacts.

[1] AI artefact is a general term for AI functions such as preprocessing data and training functions and it refers to the implementation of such functionality as a service in an AI service chain.

Preface

This thesis consists of four papers that have been peer reviewed and published at conferences. The author has been the main contributor for all the publication where she is listed as first author. The studies in all papers have been developed and designed under the guidance of the supervisors and have been co-authored with supervisors. The formatting of included papers has been changed to maintain a consistent style through the thesis but the content is unchanged.

Included Papers

Paper I Tutschku, K. Ahmadi Mehri, V., Carlsson, A., Chivukula, K. V., Christenson, J. "On Resource Description Capabilities of on-board Tools for Resource Management in Cloud Networking and NFV Infrastructures". In *IEEE International Conference on Communication Workshops(ICC)*, 2016, Kuala Lumpur, Malaysia.

Paper II Ahmadi Mehri, V., Tutschku, K. "Flexible Privacy and High Trust in the Next Generation Internet: The Use Case of Cloud-based Marketplace for AI". In *13th Swedish National Computer Networking Workshop(SNCNW)*, 2017, Halmstad, Sweden.

Paper III Ahmadi Mehri, V., Ilie, D., Tutschku, K. "Privacy and DRM Requirements for Collaborative Development of AI Application". In *13th International Conference on Availability, Reliability and Security, ARES 2018: Workshop On Interdisciplinary Privacy and Trust.*, 2018, Hamburg, Germany.

Paper IV Ahmadi Mehri, V., Ilie, D., Tutschku, K. "Designing a Secure IoT System Architecture from a Virtual Premise for a Collaborative AI Lab". In *Network and Distributed System Security Symposium (NDSS): Workshop on Decentralized IoT Systems and Security (DISS)*, February, 2019, San Diego, CA, USA.

Related Papers

Paper V Tutschku, K.Ahmadi Mehri, V., Carlsson, A. "Towards Multilayer Resource Management in Cloud Networking and NFV Infrastructures". In *12th Swedish National Computer Networking Workshop (SNCNW)*, 2016, Sundsvall, Sweden.

Paper VI Ahmadi Mehri, V., Tutschku, K. "Privacy and Trust in Cloud-Based Marketplaces for AI and Data Resources". In *11th IFIP International Conference on Trust Management(TM)*, Springer International Publishing, 2017, Gothenburg, Sweden. p.223-225.

Paper VII Ahmadi Mehri, V., Ilie, D., Tutschku, K. "Towards Privacy Requirements for Collaborative Development of AI Applications". In *14th Swedish National Computer Networking Workshop (SNCNW)*, 2018, Karlskrona, Sweden.

Acknowledgements

I would like to thank the people, who have supported me in my research education and without whom this thesis would not have been possible. First and foremost, I would like to express my deepest and sincere gratitude to my main supervisor Professor Kurt Tutschku for his invaluable guidance, encouragement, patience and continuous support throughout my research adventure. It has been a pleasure that you trusted me and gave me the opportunity to work with a larger research community.

I would like to extend my sincere gratitude to my second supervisor Dr. Dragos Ilie for providing full-time support, sharing deep experiences, inspiring, and encouraging me during my research education.

My sincere gratitude goes to Dr. Anders Carlsson for his continuous support, invaluable guidance and encouragement.
I would like to thank all my friends and colleagues in the department of Computer Science who have helped me through discussions and through sharing their knowledge.

I would like to thank the Bonseyes consortium for extensive discussion especially Tim Llewellyn, Lorenzo Keller, Professor Samuel Fricker, and Yuliyan Maksimov.
I would like to thank *City Network Hosting AB* for providing the opportunity to perform resource management tests in their Cloud infrastructure.

I am thankful to my wonderful family for their never-ending support and motivation. Special thanks to my loving husband Mohammad who never gave up on me and has been my best friend and the source of inspiration. I

am grateful to my children, Sina and Sarina, who always forgive my absence in favour of my job.
A very special gratitude goes out to my friend Benzi Craig who makes me see the world differently.

This work was partly founded by Bonseyes project which has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 732204 (Bonseyes). This project is supported by the Swiss State Secretariat for Education Research and Innovation (SERI) under contract number 16.0159. The opinions expressed and arguments employed herein do not necessarily reflect the official views of these funding bodies.

Karlskrona, August 2019
Vida Ahmadi Mehri

Contents

List of Abbreviation

ABAS	Attribute-Based Architecture Style
AI	Artificial Intelligence
BL	Bonseyes Layer
BM	Bonseyes Module
BPDR	Bonseyes Private Docker Repository
CA	Certificate Authority
CMP	Cloud-based MarketPlace
CPU	Central Processing Unit
DRM	Digital Rights Management
EPT	Encryption and Privacy Tracker
GDPR	General Data Protection Regulation
IA	Information Assurance
IoT	Internet of Things
KWS	Keyword Spotting
LXC	Linux Container
ML	Machine Learning
MP	Marketplace
NFV	Network Function Virtualisation
OPEX	Operational Expenditures

OS	Operating Systems
SAAM	Scenario-based Architecture Analysis Method
SM	Security Manager
SoC	Separation of Concerns
SVP	Secure Virtual Premise
TFX	TensorFlow Extended
VM	Virtual Machine
VP	Virtual Premise

1

Introduction

The term Artificial Intelligence (AI) was created in 1956 at the Dartmouth College summer conference where AI was recognised as a research fields in science and engineering. AI research aims to train the machine to perform human-like tasks and build intelligent entities. Currently, AI has become more popular in a variety of sub-fields from the general learning and perception to the specific applications such as chess playing, self-driving car, and disease diagnosis. Since AI is compatible with any intellectual task it is considered as a global research field [1].

The industrial view of AI focuses on data-driven system which enables digital capabilities such as perception, reasoning, learning, and autonomous decision making for machines and humans [2]. A data-driven AI system focuses on analysing collected data intelligently to provide useful predictions of future data or new insights into existing data.

A data-driven AI system requires the large volumes of data for the learning process to identify the useful patterns in the data. The social media and Internet of Things produce a huge amount of data every day but only the giant corporations, such as Google and Apple have privileged access to such data. Therefore, collaborative application developments have become a major concern in data-driven AI system as one of a solution to obtain sufficient data volumes.

Collaboration across organisations facilitates the design of sophisticated applications and involves specialisation in the engineering process and reusability of AI objects such as data sets and deep learning models. The outcome of such a collaboration process is significantly reduced cost and time of development [3, 4].

The engineering process of an AI data-driven system consists of multiple

phases in a workflow to produce the proper output. The AI workflow is a sequence of AI functions which here is referred to as service chains or pipeline. These terms are used interchangeably in this work. The concept of service chains or pipeline is used in the design and implementation of distributed services. To enable collaboration and re-usability of AI objects, the AI pipeline needs to build on components that can be reused in other pipelines. This approach leads to a component-based development process, where each component in the AI pipeline is a reusable AI service. This is the appealing features of the Bonseyes[1] marketplace for AI [4] in comparison with other open source machine learning [5–7] and data visualisation tools like Orange[2].

Although collaborative service chains bring significant benefits, as described above, they also impose some fundamental challenges in the design process. First, data protection regulations, such as GDPR, have strong ramifications on handling of private data. A particular concern for organizations and individuals is when data is being shared outside the data provider?s premise. This is described in more detail in Section 2.4. A data-driven AI system builds on learning relationships within data sets while the typical privacy enforcement mechanisms, such as anonymisation techniques or minimum data collection policy removed or do not capture the inherent relationship within data sets. Therefore, such data sets are less valuable for learning purposes in data-driven AI systems.

Furthermore, re-usability of the AI component in collaborative design needs the establishment of trust between the AI components' providers and the users of the components. The trust might extend from following terms of usage or licenses for the services, as required by their providers, to obeying local laws such as the European General Data Protection Regulation (GDPR) [8]. Section 2.5 explains the trust in the context of collaborative AI service chains.

This work aims to address the aforementioned challenges in collaborative AI service chains in several ways. First, we derived privacy and security requirements for such a system through interviews with the stakeholders. Secondly, we designed the security architecture to enable a top-down approach in compliant collaboration (i.e., from artefact's provider requirements

[1] www.bonseyes.eu
[2] Orange Data Mining Fruitful & Fun available at https://orange.biolab.si/

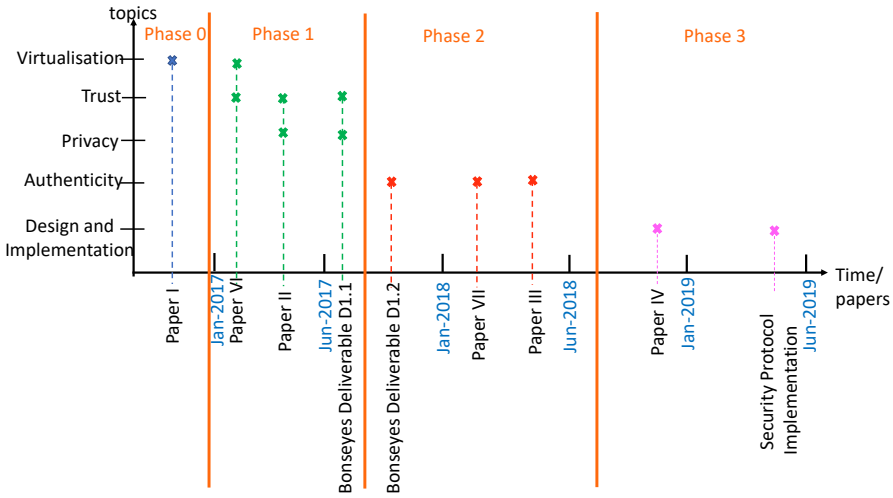

Figure 1.1: The road-map of my thesis

to the computational resources). The architecture uses the marketplace as an interaction point between the actors and adopts the cloud computing pay-per-use features for trading AI components. Additionally, it uses virtualisation techniques to facilitate the mobility of AI components in distributed environments.

The main contribution of this work is to investigate the non-functional requirements for collaborative AI service chains and find the right technology to be applied in order to facilitate secure collaborative AI service chains.

My journey as a PhD. student consists of four phases which describe my road map until this point. Figure 1.1 represents the different phases of my research adventure and development process.

- *Phase 0* was devoted towards understanding the concept of virtualisation as a technology to enable collaboration, and towards its applications in cloud computing and cloud networking. *Paper I* evaluates the capabilities of on-board tools for estimating resource utilisation for the purpose of enabling Operational Expenditure (OPEX) savings in a cloud networking and Network Function Virtualsation (NFV) environment. The results were based on the measurements performed

in a Cloud infrastructure.

- *Phase 1* was used to define the requirements of an AI marketplace in terms of security and privacy. These requirements were obtained through interviews with stakeholders and by use-cases explored within the project consortium and are described in project deliverable D1.1 [9]. The main concern of consortium members was to protect the AI artefacts from illegitimately distributed while collaborating with developers in other organisations. This phase led to understanding this problem and to defining the security interfaces in the AI service chains. *Paper II* investigates the privacy and trust issues in a cloud-based marketplace for AI and defines the knowledge gap within this scope. Additionally, it introduces some ideas on how to address these issues.

- *Phase 2* aimed at formulating the problem from Phase 1 in more details, defining specific research questions to specific use cases associated with the problem. The main contribution from this phase is a system architecture aligned with the privacy requirements discovered during Phase 1 and able to support secure interaction between artefacts in the AI service chains. Paper III describes the AI engineering process and workflow and focuses on the concept of Digital Rights Management (DRM) in conjunction with privacy and specific regulation such as GDPR, when applied to AI pipelines. Additionally, the paper explores the use of DRM technology for enforcing fine-grained access policy to support controlled collaboration. The threats against proposed methods and the possible mitigation techniques are described as well as the security architecture for AI marketplace [10, 11].

- *Phase 3* focused on the solution for secure collaboration in the Bonseyes environment and evaluated in detail the security architecture proposed during Phase 2. *Paper IV* generalises the security architecture of collaborative AI service chains by extending it into the use-case of Internet of Things (IoT) systems. The Virtual Premise (VP) and the security protocol implemented by its components are specified. The goal of the security protocol is to enable robust authentication and authorisation of virtual functions and components in AI service chains.

2

Background

The development of data-driven AI system faces the data wall problem [4]. This problem caused by the needs of data driven AI to access huge amounts of data in order to generate models of appropriate quality. Typically, only large companies can collect and manage data in great volumes, because they have a huge customer base and/or a large number of deployed products acting as data sources. The data wall problem manifests itself in smaller organisations, which find themselves incapable of engineering AI functions on the required quality level due to lacking access to sufficient amounts of data.

Moreover, access to insufficient data even prevents small organizations that may have innovative ideas based on AI, from putting them into practice. Therefore, a potential consequence of overcoming the data wall problem is that better AI functionality can be engineered and overall AI data-driven systems can be improved. However, collecting a large amount of data is time consuming and requires special techniques and sophisticated methods for handling such data. Bonseyes marketplace for AI [1] attempts to solve some of these issues by enabling collaboration among stakeholders in AI engineering. Typical stakeholder types are companies who collects data, AI specialists or developers that want to integrate AI into their applications.

Bonseyes focuses on collaboration and industrialization of a distributed AI engineering process in the context of pipelines or service chains. A pipeline or chain is a sequence of elements through which data flows, where each element processes data in a specific way that contributes to the overall goal of the system. A more detailed description of service chains is a available in Section 2.1. Bonseyes embraces the cloud infrastructure approach to reduce

[1] https://www.bonseyes.eu/

the operational cost and to facilitate the access to computational resources by a pay-per-use business model.

This work contributes to the Bonseyes project by proposing a system architecture with a strong focus on security architecture to support secure collaboration. The general questions listed below aim to give an overview of the areas addressed in this work. However, the specific research questions pursued in this work are located in Section 3.1 to address specific features in the research contribution.

l) How to design the Bonseyes architecture to address privacy and security challenges in collaborative data-driven AI?

l) What are the privacy and trust requirements to enable collaboration among actors?

l) What technologies should be used to address the Bonseyes' requirements?

This section provides an overview of the technologies and the concepts to be used for answering the general questions. It also helps in understanding the mechanisms used by Bonseyes to enable secure collaborative development.

2.1 AI Pipelines and AI Marketplace

AI pipelines and AI marketplaces are engineering concepts that have recently emerged in data-driven AI system engineering. AI pipelines have been developed in course of the lately developed machine learning (ML) frameworks, e.g. TensorFlow, cf. [12]. They describe the structure of the steps and the information flow among the design phases in data-driven AI model development. An AI marketplace enables users to re-use AI applications, frameworks, service, or functions, cf. [5–7]. We will first explain and provide examples of AI pipelines such that it can be understood how pipelines enable collaboration when using ML frameworks. Secondly, we will extend the view towards AI marketplaces and how they contribute to collaborative AI design.

2.1.1 AI Pipelines

TFX Pipeline: One of the currently most popular ML pipeline is depicted in Figure 2.1. It shows the components of Google TFX (TensorFlow Extended)

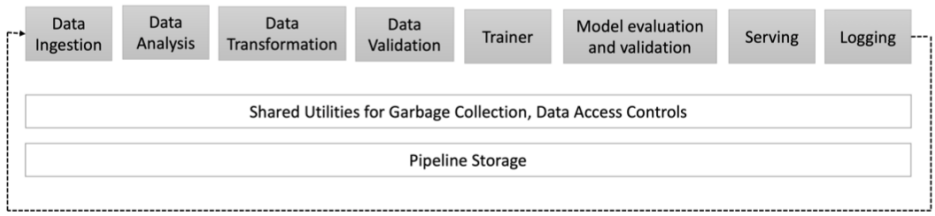

Figure 2.1: Overview of Google's ML platform for AI application design [13]

pipeline [13]. The pipeline consists of eight steps: *Data Ingestion:* obtains and imports data from data lake [2] or stream, *Data Analysis:* descriptive statistics of included features, *Data Transformation:* mapping feature-to-integer, *Data Validation:* validate the expected properties of dataset, *Training:* apply learning algorithm on dataset , *Model evaluation and validation:* investigate model prediction quality and safety to serve, *Serving:* serve multiple ML models concurrently, and *Logging*.

Figure 2.1 also shows parts of the software infrastructure of the TFX pipeline. The infrastructure consists of utilities for memory management and access control. In addition, it shows where the pipeline elements. e.g. code or data, are stored. It is assumed that this infrastructure is typically implemented in Google data centers.

Usually, data scientists prepare each pipeline component alone then combine components to an end-to-end pipeline, which is a time and labor-intensive process [14]. The use of a structured pipeline, however, permits a transition to a concurrent development process. This parallel development is also evident in [13]. The paper provides for each step a rather detailed description. Each step can be refined individually. The feasibility of individual and dedicated refinements permits concurrency in engineering. Hence, this parallelism subsequently opens up for *collaboration among multiple and concurrently working developers*.

Finally, it was outlined in Section 7 of [13] that the TFX pipeline can be deployed on Google infrastructure with the aim of improving the recommendation and ranking system in the Google Play app store. The

[2] A system or repository of data that stores data in its natural or raw format

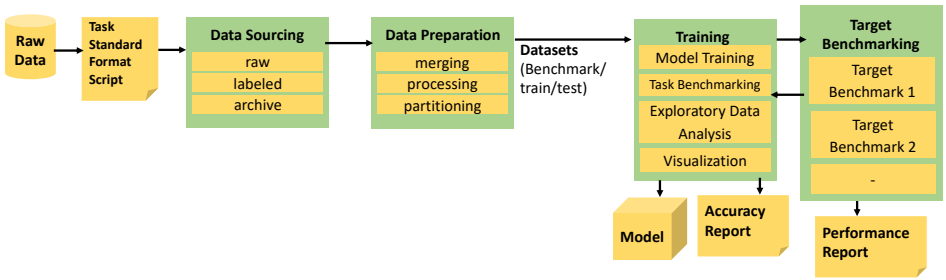

Figure 2.2: AI pipeline for data collection and training [16]

use of the improved recommending system increased the installation rate of apps from the app store by 2%.

Technically, an AI pipeline consists of various steps that have to be connected in order to execute and produce meaningful results in the AI engineering process. Each element of a pipeline supplies the input for the next element in the pipeline [15]. This requires defining the structure of input and output to and from each element, essentially specifying an API. **Bonseyes' KWS Pipeline**: Figure 2.2 shows another example of an AI pipeline. It depicts an end-to-end pipeline for keyword spotting (KWS) from spoken sentences [16]. This pipeline was used in Bonseyes as an example to study the feasibility of collaboration when training AI tasks. Again, this pipeline consists of five engineering steps. These steps are: *Raw Data processing:* download, parse, and format for compatibility, *Data Sourcing:* classified data based on its source, *Data Preparation:* processing,merging and partitioning data for next step, *Training:* feed the dataset to the model, and *Target Benchmarking:* evaluation of ML algorithm.

It is obvious that the pipeline of Figure 2.2 differs from the one of Figure 2.1 in terms of specific detailed steps. However, it is also evident that similar steps, especially for advanced data processing, are applied in both pipelines.

Moreover, Figure 2.2 introduces the Bonseyes' concept of *AI artefacts*. The Bonseyes concepts assume that AI engineering steps in a pipeline can be implemented by dedicated code or as programs, tools or functions. Hence, an engineering step becomes an object with a tangible manifestation. This

manifestation can be placed in a container, i.e. in a reusable, movable, and executable software object. Such a software object can even act also as a data source, e.g. by accessing a remote database upon request (i.e., acting as an interface towards the database) or by encapsulating the full database in the object.

The use of software objects now permits the re-use of AI engineering as well as the individual development of steps, i.e. software objects. Later on, these software objects or software components can be supplied and demanded in a marketplace.

In the Bonseyes concept the re-usability, mobility, and executability of software objects are achieved by containerisation [17], i.e. light-weight process-oriented operating system virtualisation. Such containers can be easily deployed and executed on a large range of hardware. The communication among Bonseyes' containerized AI artefact uses the HTTPS protocol and polymorphic interfaces, i.e. interfaces that can adapt their syntax and semantic during runtime.

2.1.2 Bonseyes' Marketplace and Collaboration Concepts

The general definition of marketplace is that of a place for trading products or services [18]. Bonseyes marketplace (MP) for artificial intelligence is a complex system for enabling a large variety of capabilities, workflows, and functions to support modern data-driven AI system engineering [4]. This MP provides the environment where each component of a pipeline registers and identifies. Each component of a pipeline is considered as an artefact in a Bonseyes MP.

MP is a collaboration enabler that facilitates the interaction between the AI artefact providers and the AI developers. Artefact providers publish the AI artefact as a product in a MP while registering it. MP provides the functionality to support publish, update, and remove AI artefacts as products and control visibility and access to the products. MP also wraps the artefact in a general purpose framework by using virtualisation technology to facilitate mobility, flexibility, and security of the artefact in each pipeline. A user (AI application developer or data scientist) selects the AI artefact and MP provides the secure purchase procedure and grants access to the artefact.

Figure 2.3 shows some examples of how MP enables supply and demand in collaboration and re-usability of the artefact in the different pipeline. A developer in AI pipeline 1 in the Figure 2.3 has registered its own artefact in MP which has been used in AI pipeline 2. The computational resources for each AI pipeline or each artefact in an AI pipeline might reside at different geographical locations and jurisdictions. An AI pipeline might be constructed on federated authorised resources which are trusted by Bonseyes MP (e.g., AI pipeline 1 in the Figure 2.3) or on the users' resources that are under users' control (e.g., AI pipeline 2 in the Figure 2.3) or similar to AI pipeline N in the Figure 2.3 using different possible resources. These possibilities raise concerns about the privacy and GDPR compliance in such collaboration. Also, each artefact in pipeline might have different access and execution policy that increases the complexity of a collaborated AI pipeline policy. It raises the question of how policy for the entire pipeline can be made compliant with each element's policy. This question is concerned with legal matters more than with technological aspects and there is no clear answer for it up to the time of writing this thesis.

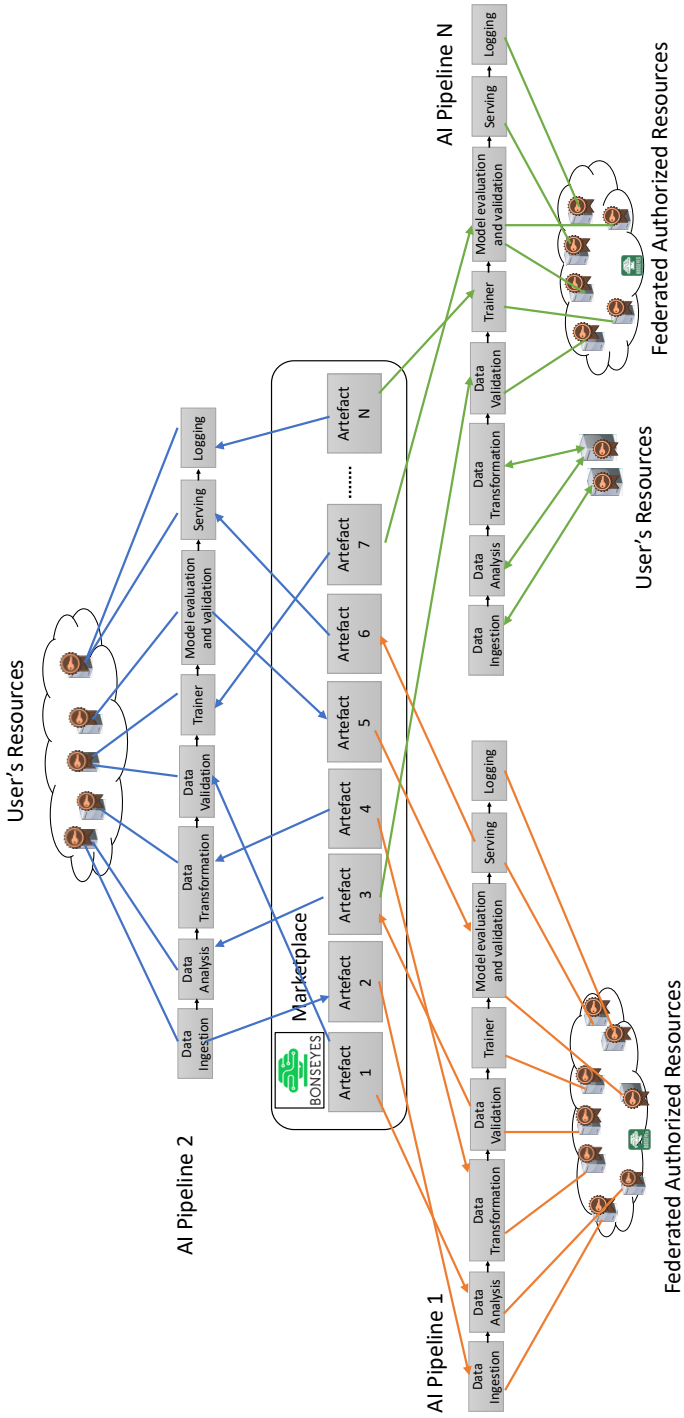

Figure 2.3: The Bonseyes Marketplace approach to enable re-usability of artefact in AI pipeline

The Bonseyes MP considers the artefact as an asset to be protected from unauthorised access while being used in different pipelines. The MP carries the policy enforcement of artefact usage based on the agreement enacted with the artefact provider at the registration time. Moreover, MP aims to offer services such as AI -artefact-as-a-service, AI-pipeline-as-a-service, and secure-pipeline-as-a-service. To achieve these goals, MP embraces the virtualisation techniques described in the next section to deliver its services and to ensures mobility and flexibility of artefact and pipeline construction.

2.2 Virtualization Techniques

The capabilities of virtualisation technologies increased dramatically during the last few years, and they became a major technology driver in the design and operation of information systems. Moreover, virtualisation techniques are required not only to be efficient but also to provide high flexibility and security [19]. These features are used by Bonseyes MP to assist collaboration between artefact providers and AI developers. The encapsulation of artefact by virtualisation techniques facilitates the mobility of artefacts. Application virtualisation are classified into two major approaches:

1. **Container-based** virtualisation is considered as a lightweight virtualisation technology as it uses the host kernel to run multiple virtual environments. The general architecture of the container-based virtualisation is presented in Figure 2.4. This approach is often referred to as container or operating system virtualisation. This type of virtualisation is considered as an efficient virtualisation technique because it allows multiple applications to operate without redundantly running other operating system kernels on the host. Also, container images tend to be small in size (tens of MBs) and start almost instantly.

   A container wraps the application with its dependencies (e.g. based image, libraries, and codes) yielding reliable mobility for the application. Docker is one of the most popular and widely used container technologies which is based on the Linux container (LXC). Docker is an open source container technology that facilitates the building, shipment, and running of distributed applications. It received the large support from the industry, for example, Microsoft built Windows

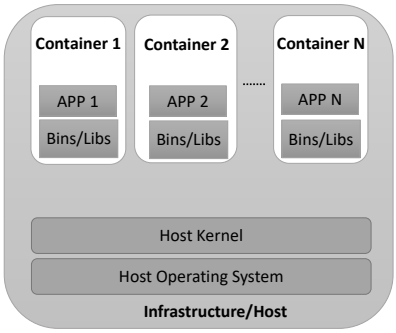

Figure 2.4: The architecture of container-based virtualisation

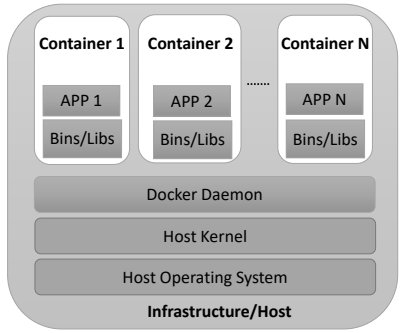

Figure 2.5: The architecture of Docker container

containers based on Docker techniques and Google facilitates running Docker containers in the Google cloud platform by Kubernetes [20].

Docker provides an additional abstraction layer which makes it compatible with different types of Operating Systems (OS) such as Linux and Windows. Docker containers enable independence between the application and the infrastructure to ease collaboration and innovation. Figure 2.5 illustrates the general architecture of a Docker container.

Docker containers run on top of the OS kernel, which minimises the computational resources required to promote a lightweight executable package. The standalone executable package of software, is called container image. In addition to the software itself, it carries its dependencies and has the ability to run in several types of infrastructure, such as bare metal servers, Virtual machines, or public cloud instances. Docker uses the kernel namespace and control groups to isolate containers from each other and from the host and is able to provide a default configuration that is fairly secure.

2. **Hypervisor-based** virtualisation provides virtualisation at the hardware level. In this approach, the hypervisor establishes complete virtual machines (VMs), denoted as guests, that represent the logical guest system with all dependencies, the entire operating system and the dedicated kernel. Figure 2.6 shows the architecture of a type 1 hypervisor, also known as a bare metal hypervisor, that runs on top of the underlying hardware of the host. The administration and monitoring of the VMs (guests) are performed by the first Guest's OS,

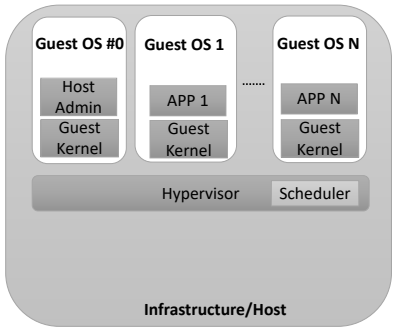

Figure 2.6: The architecture of Type 1 hypervisor-based virtualisation

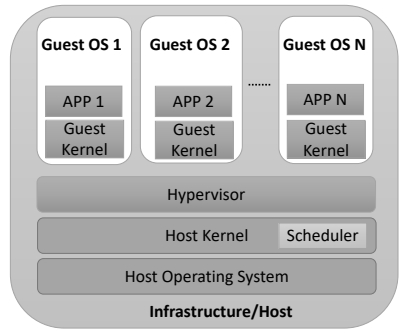

Figure 2.7: The architecture of type 2 hypervisor-based virtualisation

which runs in CPU ring 0. The first VM is responsible for creation and provision of the other VMs at the host hardware. Xen is an example of a bare-metal hypervisor.

Figure 2.7 illustrates the architecture of a type 2 hypervisor which works on top of the host's kernel like usual software. The hypervisor is responsible for monitoring and creating of VMs on the host in type 2 hypervisor. KVM is an example of a type 2 hypervisor. The differences in the architecture predetermine the pros and cons in each model. For example, a type 1 hypervisor provides better performance than a type 2 hypervisor as it does not include extra layers for the Host OS and kernel.

The aforementioned container-based and hypervisor-based virtualisation technologies, offer various capabilities based on their architectures. Container-based virtualisation could provide a higher density of containers per host as a container does not include an entire OS. This type of virtualisation is considered as lightweight virtualization technique because the size of an image and the required resources to run the application are less than what is required by a VM to run the same application. Therefore, more containers can be deployed in the infrastructure in comparison with traditional VMs [21, 22].

Thus, the above advantages associated with container-based virtualisation makes Docker the technology of choice for Bonseyes MP [10]. While the container-based techniques reduce platform incompatibility errors and

required resources for execution, they introduce privacy and security challenges. Basically, a Docker image could be executed in any Docker host that has access to the Docker image and a Docker host user is authorised to create/change/copy/remove any file inside the running container. Therefore, Bonseyes MP must provide a method to control the access and the integrity of an AI artefact that is encapsulate inside a Docker container.

2.3 Security

One of the main concepts of security is that there is not a perfectly secure system. Security is conditional and case-specific to provide adequate protection. Any security solution needs to clarify the two critical points of information: potential attackers (e.g., unauthorised user) and assets to be protected (e.g., confidential data) [23].

The core attributes of security are: a) *confidentiality:* the process to protect information from an unauthorized (not eligible)actor[3], b) *integrity:* the process to ensure the consistency, accuracy, and trustworthiness of information through its entire life cycle, c) *availability:* accessibility of resources and information in a reliable manner. In addition, one may include also the supporting or side attributes, which are d) *authenticity:* ensure the identities of the actors who participate in an exchange process, and e) *non-repudiation:* the procedure to ensure the transmission between actors cannot be denied at a later time. These five attributes together are considered the pillars of information assurance that help in defining the security requirements for the design of a system [24, 25].

All five pillars of information assurance were taken into consideration when designing the Bonseyes MP security architecture for secure collaboration. Additionally, the design work was guided by the well-known Saltzer and Schroeder [26] principles for designing secure protection mechanism.

1. *Economy of mechanism* expresses the aim to keep the design of the system simple, but still be able to meet security requirements while reducing the probability of errors and mistakes.

[3] Actor refers to the physical object consist of people or system

2. *Fail-safe defaults* embrace the white-listing principle where all access is denied by default except for specific access features that are given explicit permission.

3. *Complete mediation* enforces the idea that access control must be applied to each access to an object or resource (i.e, decisions are not cached).

4. *Open design* expresses intent to have the security mechanism open to critical observation.

5. *Separation of privilege* is a mechanism to protect the system by requiring more than one person, resource, secrets or tokens (e.g., keys, access cards) to grant access. This principle restricts access to the system by applying separate keys [27]. This principle is similar to the *separation of duty* and the process for launching nuclear weapons is a well-known example of this principle.

6. *Least privilege* is known as a principle of minimal privilege or principle of least authority which states that the access rights should be limited to the minimum that is necessary for actors to perform their tasks.

7. *Least common mechanism* expresses the limitation of mechanisms shared between multiple parts in a system to reduce the impact that a failed component may have on others.

8. *Psychological acceptability* states that the security mechanism should not be a barrier to use the system. This principle emphasizes on designing an easy to use system.

All of these theoretical principles should be considered when designing a system to support the required level of security. The required level of security for the Bonseyes project development process was determined through informal use-case interviews with stakeholders in the Bonseyes consortium. These requirements are reflected in the system architecture described in *Chapter 9*.

2.4 Privacy

This work considers the notion of privacy in information systems, which is defined as the right of individuals to know what is known about them and where this private information is stored, as well as the right to control whom to share the private data with and to what extent [28]. This definition leads the investigation of privacy requirements in different contexts from high-level policy description to low-level system implementation. The different dimensions of privacy such as identity privacy, location privacy, communication privacy, access privacy, and data processing privacy are identified and described in *Chapter 7*. The thesis aim is to address access privacy in collaborative service chains where the artefact owners/providers specify to whom and to what extent their artefact is to be shared. Therefore, Bonseyes MP should provide a way that allows artefact owners/providers to specify their privacy requirements during artefact registration. Then, MP must facilitate the policy enforcement for each artefact to ensure the controlled collaboration.

Moreover, the Bonseyes MP is compliant with the General Data Protection Regulation (GDPR), which is a European regulation for processing data [8]. Since Bonseyes MP is not directly involved in collection and processing of data in collaborative service chains, a set of requirements was put forward for data providers. MP should require the following features from data artefact providers before allowing them to register their artefact:

- Data providers must have consents from data subjects for collecting and processing data by third parties.
- Data providers should declare for which purpose data is collected and can be used.
- If the data-sets contain the private identifiable information of a data subject, the data providers must apply the privacy-preserving technology to make the processing of data compliant with GDPR requirements before registering data-sets in a MP.
- If a data subject requests for deleting her/his information from a data-set, data providers must remove data and inform the MP about changes.

Finally, Bonseyes MP should incorporate with the entities in its security architecture to address privacy requirements by artefact providers and regulations.

2.5 Trust and Trust Management

Trust is a fundamental requirement for collaborative service chains. The entities in the chains might not have direct knowledge of each other to establish on their own mutual trust for collaboration. In general, the establishment of trust between entities in a distributed system requires a dedicated trust mechanism. Trust in information usage is defined as confidence in a system's functionality to perform as it claims it should [28]. Jones et al. in [29] defined trust as the property of a business relationship. They listed the requirements related to quality and protection of digital assets for enabling trust in digital business as: a) confidentiality of sensitive information, b) integrity of critical information, c) availability of critical information, d) identification of digital object, e) prevention of unauthorized copying, f) traceability of digital objects, g) quality of digital goods, h)risk management to critical information, i) authentication of payment information [29]. According to trust definitions and the aforementioned requirements for establishing trust, this work reveals two dimensions of trust which need to be considered in the MP:

1. *Operational trust* relies on system functionality. This needs transparency and tractability of any interaction between entities in a service chain. Operational trust is influenced by the usability of a system and by the protection mechanism applied to safeguard the artefacts.

2. *Communication trust* relies on confidentiality, integrity and authenticity in each transmission between entities. The artefacts in a service chain should communicate securely with each other and they need a mechanism or protocol to enable secure communication.

Bonseyes MP should establish trust by providing a transparent procedure for granting and revoking fine-grained access to the artefacts (digital objects). This access needs to be defined and controlled based on the providers' access requirements to engaged collaboration. The transparent security protocol should safeguard the artefacts from unauthorised access and be able to

revoke the access if misbehaviour occurs. The process of assessment and decision making based on trust relationship is known as trust management. Trust management is a process which provides trustworthiness estimation in high quality and reliable services [30].

This work aims to address the two aforementioned trust dimension focused on security policy and credentials. A secure architecture for enabling collaborative AI service chain is described in *Chapter 7* and *Chapter 8*. Our trust management approach allows access to the artefacts by asserting appropriate credentials signed by the Bonseyes trusted certificate authority. This approach processes the access request based on the provider policy and requester credential. The details of our proposed method are described in Section 9.4.2.

3

Scientific Approach

3.1 Research Questions

The outcome of this work is a security architecture that enables collaboration in AI service chains. The results from this thesis are integrated with the overall MP architecture developed under the Bonseyes project. Some of the major difficulties encountered in the project are due to security requirements being defined at a very coarse level in the beginning of the project. Therefore, some of the research questions outlined below aim at identifying the threat model and at formulating the requirements for implementing an AI MP for collaborative design using the pipeline concept described earlier. The remaining questions deal with the performance and flexibility of the system in a Cloud environment.

Question 1- How can the performance and efficiency of a virtual environment in a Cloud system be described based on utilized resources?
Answering this question permits to judge whether a certain virtualisation environment is efficient enough to support the cloud-based AI system. **Chapter 6** provides the fundamental knowledge on capabilities of resource description of on-board tools to monitor and correlate the resource usage in cloud infrastructure in terms of CPU's load and utilisation. It also illustrates the predictability of resource usage by comparing measured performance between virtual and physical environments.

Question 2- How can one map the requirement of privacy and trust to the needs of cloud-based MP for AI?

This research question is addressed in **Chapter 7** by providing a definition of privacy and trust, specifically adapted to the context of a cloud-based

AI MP. The chapter describes the different privacy dimensions (e.g., location privacy, access privacy) and trust dimensions (e.g., device trust, operational trust) from a general point of view, but also specifically for the context of the cloud-based marketplace for AI. **Chapter 7** investigates the possibility of using the existing methods such as privacy by design, to achieve appropriate privacy and trust dimensions in the AI marketplace. It also introduces initial ideas about concept of a virtual premise. Later work helped in refining the concept of virtual premise. The outcome of this work is described in **Chapter 8** and **Chapter 9**. **Chapter 8** describes a trend from privacy requirements for enabling collaborative concepts in AI system engineering to a practical solution for facilitating collaboration between different stakeholders.

Question 3- What are the threats against Digital Rights Management (DRM) and privacy enabling mechanisms used for collaborative AI system engineering based on AI pipelines?

This question is addressed in **Chapter 8** by describing the ordinary DRM constraints to address law-driven privacy requirements (e.g., GDPR). **Chapter 8** proposes the adaptation of DRM concepts and fine-grain access control to the artefacts. It uses a threat modeling approach to investigate the possible threats against the proposed DRM in a collaborative AI pipeline. **Chapter 8** also suggests the mitigation techniques for the investigated threats in a DRM scheme.

Question 4- How should a flexible architecture for virtual premise be designed to enable robust authentication, access and execution control in virtual and containerised environments?

Chapter 9 provides the answer to this research question by generalising the architecture of virtual premise to IoT service chains and showing its flexibility to address different security and privacy policies. In addition, it proposes a security protocol to control the access and execution of each AI pipeline element in the containerised environment.

3.2 Research Methodology

As the main focus of this work is to address the secure collaboration in the system design, security is a key feature to enable such a system. Security science incorporates various disciplines such as social, political, and hard sciences. This mixture of disciplines requires an adaptation of existing research methodology [31]. This section provides an overview of four general research methods in security science and describes the research methodology that is used in this work.

3.2.1 Security Science Methodology

Since security science is a relatively new discipline and it is not a well-established science field, there is a lack of a method to measure the quantity of all security attributes mentioned in Section 2.3 except confidentiality as described in [32]. Therefore, security science largely relies on qualitative measurement [23, 31]. The four general research methods that are widely used in the security science are summarised as follows in [31] (p.70-75):

1. **Observational Research** is the method to understand the behaviour of a real-world system. It is used for the analysis of the system behaviour during special events such as cyberattacks behavior to understand target and tactics. It includes a) *exploratory studies* that collect and analyse known system design to determine a general theory of behaviour, b) *descriptive study* which focuses on the specific subset of system in details, and c) *machine learning* that uses applied mathematical techniques to detect correlations in large volumes of data.

2. **Theoretical Research** is a logical investigation of a system. It involves the definition of the system and its behaviour. This type of research is valuable to generate theories about the behaviour of the system under study, but for various reasons (e.g., complexity, cost) does not validate them in practical environment. Theoretical research consists of a)*formal theory* where mathematics or other logical languages are used to model and define the possible behaviour of the system, b) *formal theoretical research* which is a theoretical model that was formally proven and traceable, and c) *simulation* to explore the complex system for understanding the theoretical model with

enough confidence. The theoretical approach is not limited to the aforementioned fields, it can also involve use cases to define a system and its environment behaviour.

3. **Experimental Research** is a quantitative research method in which hypotheses are defined for the system under study and experiments are carried out to obtain evidence about their validity. If the researchers have full control over the experimental setups and configurations the method is called *hypothetico-deductive* otherwise it is called *quasi-experiment*. The second approach required a validity threat analysis to determine potential sources of error.

4. **Applied Research** is the method which aims at applying scientific knowledge to solve some problem. Applied research is the process to quantify the effectiveness of the research solution which is the main differences of this method with other aforementioned methods. This method consists of a) *applied experimentation* that is used to compare the performance of different solutions, and b) *applied observational study* which studies the system behaviour in different situations.

3.2.2 Methodological Approach

This section describes our approach towards architectural design for Bonseyes MP. The aim of Bonseyes project is to provide a MP for an AI engineering system by enabling collaboration between different actors. This thesis focuses on controlled collaboration in AI engineering system and considers architecture design for compliance control in AI function chain. Such an architecture must follow the system requirements outlined in Section 3.3 while at the same time the five pillars of information assurance are addressed properly.

The Bonseyes MP is considered a complex system because it must satisfy a mixture of requirements from AI functions, actors, datasets, infrastructures, and others which are part of collaborations. The requirements need to be well defined before moving to the design and implementation stage. However, this was not the case at the start of the project as described in the beginning of this chapter. Therefore, we used a design methodology that relies on the spiral model [33] which facilitates iterative development. The success of the

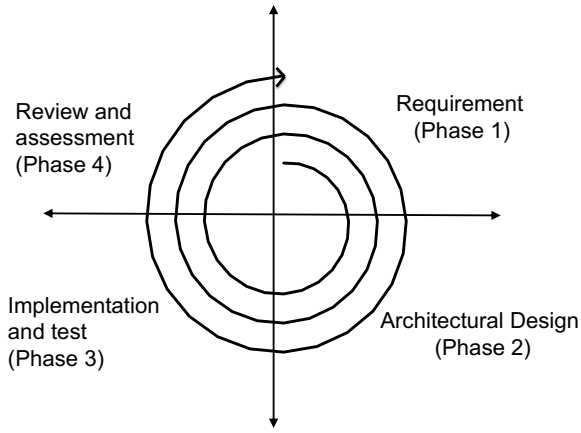

Figure 3.1: Spiral model of system design

spiral model has been previously proved in infrastructure design projects such as GENI [34].

The design spiral model is depicted in Figure 3.1 which is inspired by [35]. Each revolution compromises the four phases of system design development. The first phase of revolution identifies the system requirements. The second phase consists of architectural design with respect to identified requirements. The third phase represents the implementation of an architecture that needs to be tested which is subsequently evaluated in the fourth phase. However, new requirements might arise during phase three and four which need to be addressed in the next iteration. One advantage of this approach is that it allowed the stakeholders in the project to refine the requirements based on the output from each iteration.

In Bonseyes project, the requirements were not well-defined in phase 1. Hence, Only coarse requirements were available for the initial phase of architectural design. This created significant difficulties in understanding what are the right design choices during phase 2. Therefore, our design is drived from a use-case approach in theoretical research in which we reason about the validity of specific design choices. This approach does not require a complete implementation of the use-case under study. An full implementation can be complicated, time-consuming and not necessarily

expanding the scientific knowledge. The proposed security architecture is analysed in Section 9.4 using the approach mentioned above. Next, we applied the experimental research in phase 3 to verify the applicability of the system design. The security protocol is implemented as described in Section 9.4 in our lab environments and other relevant environments. The results of the first implementation illustrated in Section 4.4. At the time of writing this thesis, phase 4 is ongoing, and the security architecture is under review and assessments by the Bonseyes consortium.

3.3 Requirements

Bonseyes MP for AI has a complex design system because it needs to address a wide variety of requirements that depend on the artefact type. For example, the requirements of data artefacts are dependent on the types of data accessed through the artefact. For instance, within EU processing of private data must be compliant with GDPR and local legislation. On the other hand, medical data, for example, must additionally follow data protection and privacy regulations enforced within the medical community. These types of requirements can differ between countries, to the point where they can be in contradiction. Thus, requirements cannot be easily generalised, being instead rather specific to the use-case. Bonseyes [10] provides a list of requirements obtained through interviews with consortium stakeholders that show some of these issues.

The main concerns of all consortium members was to protect the digital assets from being exfiltrated while retaining the re-usability of AI artefacts in a distributed and collaborative AI engineering process. This brings piracy concerns into the marketplace design process. Therefore, *Chapter 7* and *Chapter 8* contribute to privacy, trust, and digital rights requirements in collaborative AI service chains. The result of requirement analysis in [36] was input to the initial system architecture design which facilitates the secure execution of artefact in edge and Cloud. The knowledge about practical requirements for enabling secure collaboration got matured while the project was running, and [11] provides the detail requirements and challenges in security architecture.

STRIDE Threats	**IA Attributes**	**Definition**
Spoofing	Authenticity (Authentication)	masquerade as another
Tampering	Integrity	modify the information on a network, on a disk or in a memory
Repudiation	Non- Repudiation	someone claims that a past event did not happen or that claimant was not responsible for it
Information Disclosure	Confidentiality	expose information to an entity that is not authorised to have it
Denial of Service	Availability	consume all available resources necessary to provide service
Elevation of Privilege	Authenticity (Authorisation)	a user or program that can technically do things they are not supposed to do

Table 3.1: The relation between STRIDE threats and attributes of Information Assurance(IA) [37]

3.4 Threat Modeling

Threat modeling is used to model an existing or to be designed system to find security risks. Threat modeling provides a view of the threats that could impact on the system [37]. There are multiple ways to do the systematic threat modeling such as STRIDE and attack trees which can be described as follows:

- *STRIDE* was invented by Loren Kohnfelder and Praerit Garg at Microsoft [1] and it stands for Spoofing, Tampering, Information Disclosure, Denial of Service, and Elevation of Privilege. STRIDE is designed to help initial target users in identifying the types of attack the soft-

[1] https://en.wikipedia.org/wiki/STRIDE_(security)

ware might experience. It is built on the five pillars of information assurance [37].

- *Attack tree* is a formal way to describe the security of a system based on different attacks against an asset. It represents the attacks against the system in a tree structure where the goal of the attack, the asset at risk, is the root node of the tree. The leaf nodes represent various ways to reach the asset. The nodes on the path from a leaf node to the root node are various sub-goals that need to be accomplished in order for the attack to succeed. Once a root node has been determined, the next level of the tree is defined by identifying potential threats against the root node. This can be done, for example, by using brainstorming, STRIDE, or literature reviews [37]. The attack tree will be completed through iteration over nodes, until the analyst is satisfied with the threats identified or is unable to identify additional threats. The resulting attack tree can be shared within the security community, to enable its reuse in some other software or use-cases.

Both threat modeling approaches rely on having access to a model of the system protecting the asset. The Bonseyes architecture still is under development and its components are not yet well defined due to the complexity of the requirements. For this type of moving target, we felt that an approach similar to STRIDE would work better than attack trees in defining the threat model for the Bonseyes MP. Table 3.1 shows the definition of each element in the STRIDE approach and the relationship between the STRIDE threats and the attributes of Information Assurance (IA). **Chapter 8** uses a STRIDE-like method to analyze the possible threats against the proposed model of the AI MP which is summarised in the Table 3.2.

STRIDE Threats	**Threat example**	**Mitigation**
Spoofing	Spoofing time source	Defeat by using time server authentication, e.g. NTP Autokey
Tampering	1-Tampering architecture elements 2- User modify license contents 3- Rolling back the time on the artefact host to use artefact past expiration date	1- Use trusted computing environment to protect elements integrity and actual development of AI pipeline is done by third party 2- BL checks the signature in the artefact license 3- After an initial connection to the time source, the BL is able to detect deviations from monotonically increasing time.
Repudiation	MP repudiates the issuance of license	Signing the artefact license by MP private key
Information Disclosure	1- Artefact breach against license purpose 2- Host breach against license purpose 3- Artefact execution outside a VP 4- Artefact execution on wrong VP 5- artefact execution outside allowable geographic region	1- BM on host executing artefact must verify information encoded in the artefact license to ensure its use case is allowed 2- Host must have the digitally signed license from Bonseyes CA that approve use scope and tied to the host's ID 3,4- The host and the artefact license contains a set of virtual premise identifiers that must match for artefact execution 5- Encoding allowed geographic regions in the license which must be matched with the current geolocation
Denial of Service	1- Denial of service against license server 2- Denial of service against license revocation server 3- Denial of service against SM 4- Denial of services against actual pipeline	Accept the request from authorized user only and assign enough resources to the server [2]
Elevation of Privilege	1- Artefact execution by unlicensed user 2- Artefact execution outside allowable geographic region 3- Artefact execution outside a VP 4- Artefact execution on wrong VP	1- Users have to authenticate to the BL with a digital user certificate signed by a Bonseyes certification authority (CA) 2,3,4- Legitimate skilled Bonseyes users who able to modify code of BL or BM that elevate the access and unauthorized usage. They could instrument the host to copy protected artefacts

Table 3.2: Analysis of STRIDE inspired threat modeling for Bonseyes MP [11]

3.5 Validation Method

This section aims to describe the validation of results of this thesis, in particular proposed techniques and methods.

We applied the validity classification scheme from [38] which is similar to the method used in controlled experiments in software engineering [39]. The classification scheme considers four aspects of the validity: *Construct validity:* determine if the operational measures reflect what is investigated by the research questions , *Internal validity:* determines to what degree found cause-effect relationships are trustworthy, *External validity:* limits on generalisation and applicability of the results , and *Reliability:* determines the level to which an experiment and its results can be replicated by other researchers. Here we choose to focus on internal and external validity.

Internal validity: is a vital consideration in a quantitative study, where the measurement concerns the casual relationship. This stipulates that the dependent variables and results are not affected by confounding variables [38]. *Paper I* was the quantitative studies where the correlation between dependent variables is calculated. The study focuses on the relationship between virtual and physical CPUs in terms of load and utilisation. The impact of other factors and processes such as measurement tools on load and utilisation in a test environment was calculated and it was observed that it is negligible. Also, the statistic analysis is based on computing average over 50 independent measurements for each experiment to account for randomness. We collected the data every second in the five minutes interval to study the capability of the on-board tools on a small time scale.

External validity: refers to the generalisation of finding and to what extent the result would be applied in other cases [38]. The result from *Paper I* could be applicable for estimating resource usage in a Cloud and obtained OPEX savings. Hence, the statistic analysis is on the steady-state measured interval which defined based on the test environment behaviour.*Paper II* investigates the general aspects of privacy and trust requirements for the cloud-based marketplace for AI. It identified the privacy and trust issues

[2] We did not consider general DoS attacks in [11] because defending against DoS attack is difficult and the defense mechanism depends on the type of DoS attack considered. However, the proposed mitigation method works when a registered user attempts to overload hosts by simultaneously launching many containers.

in the context of Cloud, which could be used in any Cloud-based use-cases. *Paper III* identifies the constraints in the proposed DRM in general, but the threat analysis is very specific for the collaborative AI environments. Hence, the identified constraints could be used in collaborative design. *Paper VI* describes the concept of a VP in collaborative AI engineering system which could be applied in other use-cases that have a similar security requirement (e.g., IoT future architecture).

4

Results

This chapter provides a summary of the research findings and highlights the main results of included papers in a related subsection below. The author explains the relationship between the obtained results in each paper and the overall context of the thesis.

4.1 Resource Description for Performance Management of Cloud Infrastructure

In [40], we provide a method to monitor, describe, and correlate load and performance relationships using on-board tools for CPU monitoring in Cloud. This method can be applied to general-purpose servers, which are typically used in Cloud infrastructure, to evaluate their off-the-shelf capability in the virtualised environment. Figure 4.1 shows how we model the relationship between physical and virtual environments. We compare and correlate the synthetic load as experienced by the virtual appliance ("inside" or "Guest") and physical environment ("outside" or "Cloud operator") in Figure 4.2. The graphical analysis illustrates the correlation between inside and outside view where each scattered point represents the relationship.

Also, the comparison of CPU usage inside and outside envision the predictability of resource usage by on-board tools. Figure 4.2 shows the variation in CPU usage in the different synthetic load in the range of 10-100% that applied inside. As Figure 4.2 shows, increasing load in the inside raises the CPU usage in the outside which provides evidence on the predictability of resource usage especially in the range of 25-50% load.

The above methodology shows how to characterise the relation of workload on Cloud infrastructure. In the AI design process described in Section 8.2, multiple AI pipelines will implement this methodology as a way to

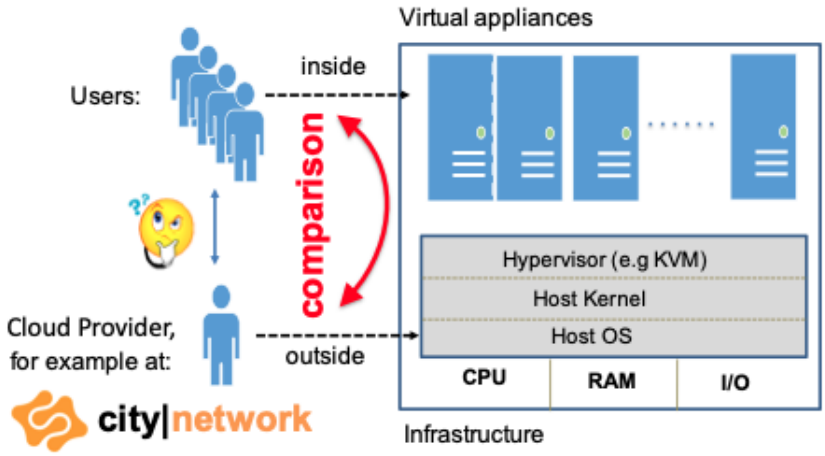

Figure 4.1: Modeling the load relationship between virtual and physical environments

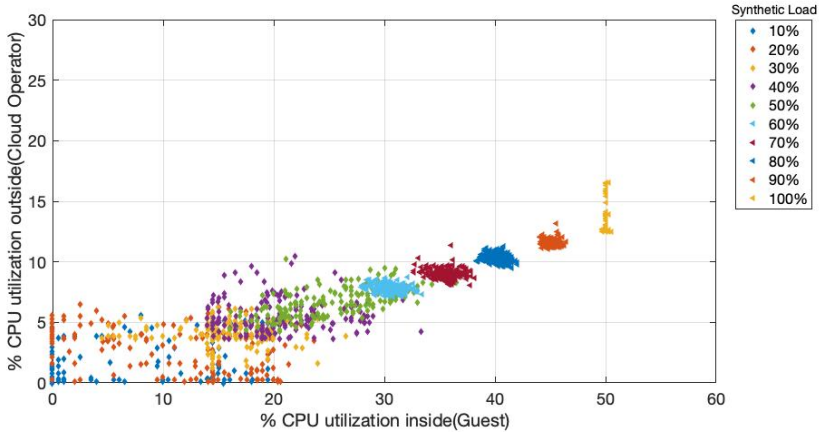

Figure 4.2: The correlation relationship of CPU utilisation between inside and outside

estimate load. The understanding of performance enables a discussion on whether the Cloud infrastructure is capable of executing parallel training in parallel AI pipelines and what amount of computational resources are needed for conducting AI design processes. The graphical results can be used to estimate the required computational power for AI pipelines executing on Cloud infrastructures.

4.2 Privacy and Trust Requirements for Cloud-based Marketplace

In [36], we investigate the requirements for privacy and trust in a Cloud-based MarketPlace (CMP) for AI. It identifies the privacy and trust functionalities that are not addressed adequately in the Cloud such as:

(a) *Traceability* of data sets and the model, which determines the reliability of the CMP and the quality of shared AI resources.

(b) *Controllability* which is the ability to control the privileged access to the datasets and model and generate the access report when needed

(c) *Location privacy* which is enabled the access control to the data, based on geographic location especially for private datasets.

The functionalities as mentioned above define the core concept of Virtual Premise (VP) to enable the privacy by design in CMP. The description of the early version of VP outlines the transparency and controllability as appealing features to facilitate the flexibility of artefacts in distributed and virtualised environments.

Figure 4.3 represents the early concepts of the VP which aims at fine-grained monitoring of data usage at distributed locations. It employs encryption for confidentiality of data transferred between trusted data centres and suggests the use of a privacy tracker for controlling the data access.

At that stage of the research, we focused on enabling a CMP to overcome the data wall problem in AI engineering process by sharing data between different data providers. After that, the concept of AI pipeline was introduced to the Bonseyes project as described in [13, 16], which aimed at enhancing

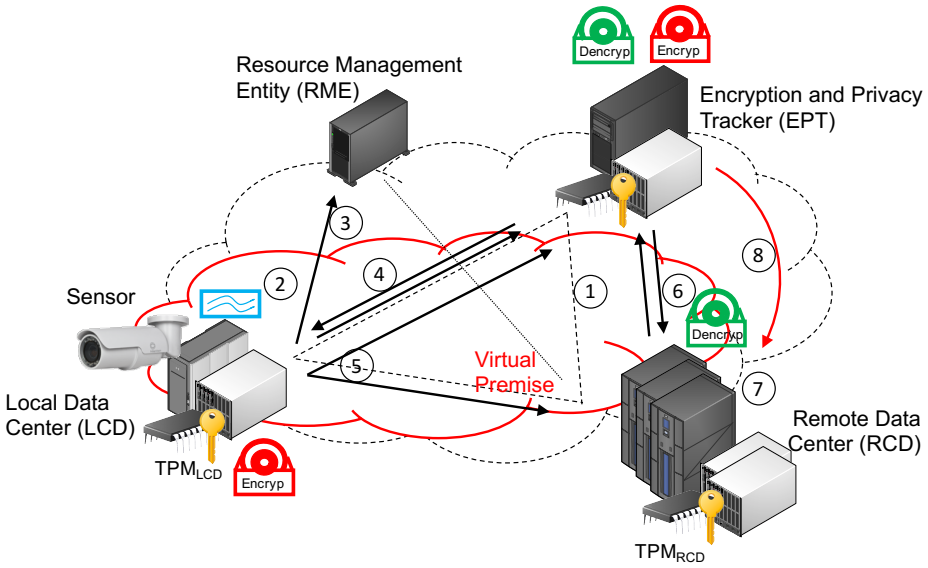

Figure 4.3: The early concept of VP using Encryption and Privacy Tracker(EPT)

the VP, for example in monitoring data usage and in securing AI pipelines. Next, we studied how to define the privacy and security requirements for each element in the pipeline in order to facilitate secure AI pipelines.

4.3 Requirements for Collaborative AI engineering

We describe the legal and technical requirements of privacy to enable collaboration in the pipeline based AI engineering system in [11]. The paper considers DRM as a potential technology to address the privacy challenges and outlines the relationship between privacy and DRM in Section 8.3. [11] adapted DRM concepts and suggested a fine-grained access control approach for artefacts. Also, it identified the potential constraints of the artefact usage that can be embedded in a license features such as:

(a) *Validity period:* allowed access duration to the artefact

(b) *Beneficiary:* unique identifiers of the licensed consumer

(c) *Purpose:* an encoded metadata of permitted scope for artefact use

License Features	Threats
Validity	• Blocking communication with the time source • Blocking communication to the license revocation server • Spoofing a time source • Rolling back the time on the artefact host to use artefact past expiration date
Beneficiary	• Artefact execution by unlicensed user
Purpose	• Host breach against license purpose • Artefact breach against license purpose
Location	• Artefact execution outside a virtual premise, i.e. host is lacking a valid BM • Artefact execution on wrong virtual premise • Artefact execution outside allowable geographic region
Peering	• Unauthorized peering

Table 4.1: The category of simple threats against license features

(d) *Location:* topologic or geographic place where artefact can be utilized

(e) *Peering:* regulates which entities the artefact is allowed to interact with

Furthermore, [11] analyses the threats against the artefact usage constraints mentioned above by considering simple and advanced threats. Table 4.1 categorises the simple threats against license features. The mitigation techniques to address these simple threats are described in Section 8.4.1.

Finally, [11] investigated the advanced threats caused by skilled malicious users who are able to modify the protected code running inside the container

(BL) and inside the host (BM). These are presented in Section 8.4.2. Since the outlook for defending against advanced threats looks gloom, we present two complementing approaches to raise the difficulty of mounting a successful attack: implementing a trusted computing environment to protect the integrity of BM and BL, and outsourcing the actual implementation of AI pipeline to a third party. Moreover, in [11], we present the components of the security architecture and describe the message exchanges taking place between the components in security architecture. The detailed communication protocol is illustrated in [41].

4.4 Robust Authentication and Authorisation of AI Artefacts

In [41], we developed and refined the flexible architecture of the VP and proposed a robust authentication and authorisation procedure for AI artefacts. The primary outcome of this work is mutual authentication and authorisation of the involved entities. Figure 4.4 shows the six discrete phases of this process. The red circles in Figure 4.4 highlight the separation between policy enforcement point in the artefact (BL) and in the host (BM), respectively. The details of each step in the process are described in Section 9.4.2.

The authentication and authorisation protocol aims to control access and execution of the AI artefact in an AI pipeline. It verifies the host eligibility to execute the artefact and the artefact permission to run on the given host.

Moreover, we proposed and refined VP as a solution to solve the technical challenges in a collaborative and distributed AI pipeline. Our solution can be applied to the other cases, in particularly to IoT scenarios that require similar security concepts for collaborative services. Section 9.2 discusses the future IoT architectures and outlines the need of authentication for elements in an IoT service chain, which is similar to an AI pipeline. Figure 4.5 shows the adaptation of VP to an IoT architecture. The VP provides flexibility in each pipeline through programming, policies, and orchestration. The concept of VP in IoT service chains are complementing edge computing concept such as multi-access edge computing [42].

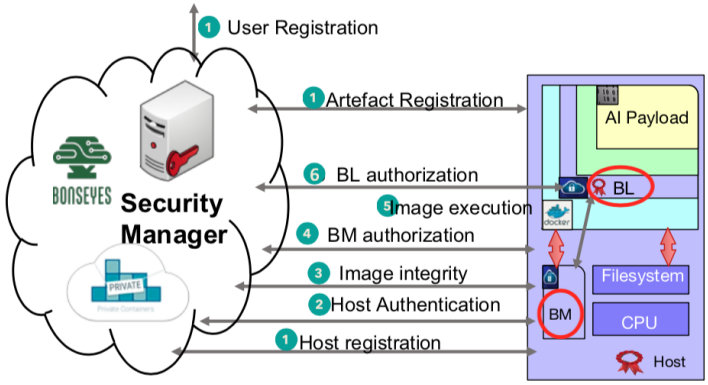

Figure 4.4: The discrete phases of mutual authentication and authorization

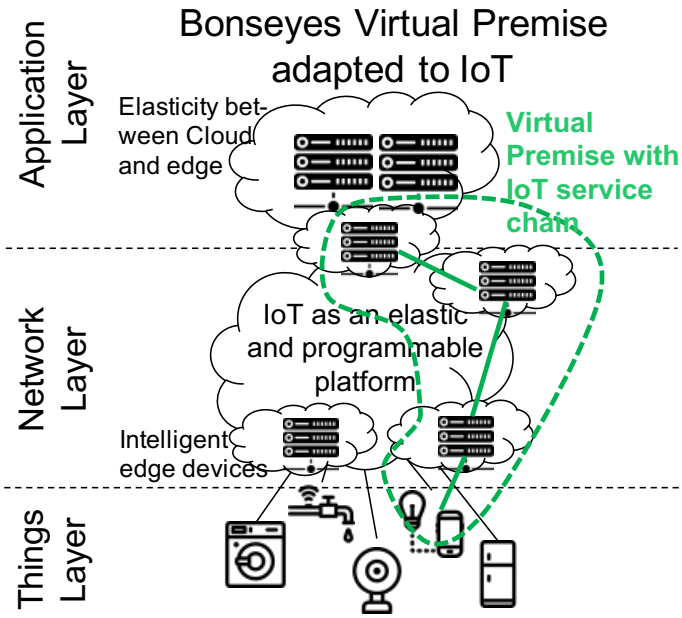

Figure 4.5: The adaptation of VP in IoT architecture

TRL	Task Description	Achievement Rate
3: Experimental proof of concept	Experimental proof achieved	100%
4: Technology validated in lab	implementation of Figure 4.4 in Cloud infrastructure using OpenStack	100%
5: Technology validated in relevant environment	Implementing the subset of the protocol in Figure 4.4 in Verne Global	70%

Table 4.2: Technical Readiness Level(TRL) for authentication and authorisation mechanisms

Verification of the Technical Readiness Level

After the publication of [41], we aimed at the verification the maturity of the proposed robust authentication and authorisation mechanism using an approach similar to Technical Readiness Level (TRL) as described in [43, 44]. TRL scales from 1 to 9, where 9 is the highest and 1 is the lowest maturity level. TRL verification is required by European Union H2020 projects that partly fund this research work.

At first, we implemented a minimal pipeline and the authentication, authorisation, including BL and BM, in an experimental environment. According to TRL requirements this experimental proof yields TR level of 3, cf. Table 4.2. After that, the experimental implementation was refined and implemented in an OpenStack Cloud environment at BTH. The components of security architecture: Security Manager (SM), Bonseyes Certificate Authority (CA), MarketPlace (MP), and BM instantiate in virtual machines in the OpenStack Cloud environment. The pipeline elements (artefacts) were encapsulated in a Docker container as described in Section 9.4.2. Since TR level of 4 requires successful implementation and validation in a lab environment, we conclude that the needs for TRL 4 have completely been achieved. Finally, the considered mechanism was transformed and adapted together with nViso[1], who is a technical coordinator of the Bonseyes project, to a commercial Cloud environment. This Cloud environment was provided by the company Verne Global[2], which is not associated with the Bonseyes project. Verne Global offers commercially a relevant environment where an AI pipeline can be implemented. A snapshot of the Verne Global environment is depicted in Figure 4.6. The environment comprises a large number of Cloud server racks, which are equipped with CPU and multiple

[1] https://www.nviso.ai/en
[2] https://verneglobal.com/

NVIDIA Tesla V100 Tensor Core GPUs[3] for fast AI processing. Since Verne Global provided only two bare-metal servers for the trial implementation, we simplified the authentication and authorisation mechanisms for host. As we had only one host with four GPUs to execute the minimal AI pipeline, the step 4 in Figure 4.4 was not implemented in Verne Global infrastructure. Nevertheless, it showed the execution of a minimal AI pipeline successfully as defined by nViso. The trial was completed in April 2019. According to the concept of TRL, the successful implementation of the mechanism in the industry relevant environment yields a TRL 5.

In summary, we assume by this comparison of achieved TRL the current maturity of the proposed solution for authentication and authorisation of AI artefact is approximately on TRL 4. The TRL 4 is a promising level for future extension and development of the security mechanism in collaborative AI pipelines.

[3] https://www.nvidia.com/en-us/data-center/tesla-v100/

Figure 4.6: The racks with GPU in Verne Global data centre where the AI pipeline instantiated

5

Conclusion and Future Work

This licentiate thesis investigated the requirements, mechanisms, and capabilities of the virtual environments to address the privacy and security challenges in collaborative AI service chains. We started this work by analysing the capability of the on-board tools to monitor and correlate the resource usage in Cloud infrastructure, which can facilitate the reduction of operational cost. It also helps in estimating the required computational resources to conduct AI pipelines.

Next, we studied the non-AI functional requirements, particularly privacy and trust, to enable collaboration in Cloud-based MarketPlace (CMP). The study highlights the existing data security problems and helps to understand, formulate, and refine the research questions. Then, we investigated how to transfer the requirements as mentioned earlier to a technical implementation for building trust in AI pipelines. This study led to the first definition of a Secure Virtual Premise (SVP).

Furthermore, we investigated the constraints of license management systems to address the technical and legal requirements of privacy in AI system engineering. We adapted the concept of DRM technologies towards obtaining fine-grained access control. Our approach enabled control over artefact execution on host. Then, we highlighted the security problems in the proposed solution by threat modelling. Threat modelling helped us to understand the possible threats and to find mitigation techniques for reducing the severity of attacks. Also, it improves our understanding of the security components, their role and vulnerabilities.

The current understanding of SVP's role in enabling secure collaboration in AI service chains is presented in [41]. The SVP utilises the mutual authentication and authorisation for each element in AI Pipelines. An SVP acts as a policy enforcement point for AI artefacts in collaborative service

chains.
Finally, we demonstrated a reasonably simplified version of the security architecture in an industry relevant Cloud infrastructure. The implementation comprises the essential security components of Bonseyes Layer (BL), Bonseyes Module (BM), Security Manager (SM), and CA, cf. Section 9.4. According to our analysis, we reached approximately level four in the TRL scale at the time of writing this thesis. We expect to achieve a higher technical capability of the solution in the future.

Outlook

From the result obtained in this work, we believe the work for secure collaboration should continue and might be spanning into three research directions in the future. The first direction is to refine the mechanism and concept of security towards more secure SVP concepts. In a current security mechanism, a Security Manager (SM) is a central decision point for AI artefacts that can be a single point of failure. In the future, the decision point can be distributed to improve the availability of the SM services for SVP by using blockchain technology [45] or multiple SMs [46]. The second direction could be the generalisation of secure collaboration concepts from this thesis towards non-AI service chains which have the similar security requirements, such as future IoT services in medical IoT environment where wearable and implant devices are being used. Our secure collaboration approach could be adopted to address the information collaboration challenge in medical IoT as described in [47]. The third research direction is the improvement of verification methods, and concepts for the security architecture suggested in this thesis. The verification could be done by systematic technology readiness assessment or by investigating the vulnerability of the running system against identified threats.

References

[1] S. J. Russell and P. Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited, 2016.

[2] Bis Data Value Association. *Data-driven Artificial Intelligence for European Economic Competitiveness and Societal Progress*. BDVA Position Statement paper available at http://www.bdva.eu/sites/default/files/AI-Position-Statement-BDVA-Final-12112018.pdf. 2018.

[3] I. Stoica, D. Song, R. A. Popa, D. Patterson, M. W. Mahoney, R. Katz, A. D. Joseph, M. Jordan, J. M. Hellerstein, J. E. Gonzalez, et al. "A Berkeley View of Systems Challenges for AI". In: *arXiv preprint arXiv:1712.05855* (2017).

[4] T. Llewellynn, M. Fernández-Carrobles, O. Deniz, S. Fricker, A. Storkey, N. Pazos, G. Velikic, K. Leufgen, R. Dahyot, S. Koller, et al. "BONSEYES: platform for open development of systems of artificial intelligence". In: *Proceedings of the Computing Frontiers Conference*. ACM. 2017.

[5] *PipelineAI - Home*. available at `https://pipeline.ai/`. (Visited on 04/06/2018).

[6] *Data Market Austria*. available at `https://datamarket.at/`. (Visited on 04/06/2018).

[7] *Industrial Data Space e.V.* de-DE. available at `http://www.industrialdataspace.org/`. (Visited on 04/06/2018).

[8] European Parliament and Council of the European Union. "Regulation (EU) 2016/679 of the European Parliament and of the council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)". In: *Official Journal of the European Union*. Vol. 59. L 119. ISSN: 1977-0677. May 2016.

[9] S. Fricker, Y. Maksimov, D. Oster, S. Koller, L. Wang, G. Velikic, M. Milosevic, T. Llewellyn, R. Tanner, J. Burger, T. Paulin, K. Tutschku, V. Ahmadi Mehri, O. Deniz, A. Elafrou, A. Storkey, R. Dahyot, E. Ozer, N. Pazos, N. Oerhani, and D. P. M. *Deliverable D1.1: Bon-*

seyes Requirements. Bonseyes – Information available on request at https://www.bonseyes.eu. 2017.

[10] K. Tutschku, V. Mehri, S. Fricker, Y. Maksimov, L. Keller, T. Llewellyn, and T. Paulin. *Deliverable D1.2: Initial Bonseyes System Architecture*. Bonseyes – Information available on request at https://www.bonseyes.eu. 2017.

[11] V. A. Mehri, D. Ilie, and K. Tutschku. "Privacy and DRM Requirements for Collaborative Development of AI Applications". In: *Proceedings of the 13th International Conference on Availability, Reliability and Security*. ARES 2018. event-place: Hamburg, Germany. New York, NY, USA: ACM, 2018.

[12] *TensorFlow*. en. URL: `https://www.tensorflow.org/` (visited on 04/30/2019).

[13] D. Baylor, E. Breck, H.-T. Cheng, N. Fiedel, C. Y. Foo, Z. Haque, S. Haykal, M. Ispir, V. Jain, L. Koc, et al. "Tfx: A tensorflow-based production-scale machine learning platform". In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2017.

[14] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, et al. "Mllib: Machine learning in apache spark". In: *The Journal of Machine Learning Research* 17.1 (2016).

[15] S. Das and U. M. Cakmak. *Hands-on Automated Machine Learning*. en. Packt Publishing Limited, 2018.

[16] M. de Prado, J. Su, R. Dahyot, R. Saeed, L. Keller, and N. Vallez. "AI Pipeline-bringing AI to you. End-to-end integration of data, algorithms and deployment tools". In: *arXiv preprint arXiv:1901.05049* (2019).

[17] S. Soltesz, H. Pötzl, M. E. Fiuczynski, A. Bavier, and L. Peterson. "Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors". In: *ACM SIGOPS Operating Systems Review*. Vol. 41. 3. ACM. 2007.

[18] *MARKETPLACE | meaning in the Cambridge English Dictionary*. en. URL: `https://dictionary.cambridge.org/dictionary/english/marketplace` (visited on 03/03/2019).

[19] N. M. K. Chowdhury and R. Boutaba. “Network virtualization: state of the art and research challenges”. In: *IEEE Communications magazine* 47.7 (2009).

[20] *Enterprise Application Container Platform*. en. URL: `https://www.docker.com/` (visited on 03/18/2019).

[21] M. G. Xavier, M. V. Neves, F. D. Rossi, T. C. Ferreto, T. Lange, and C. A. De Rose. “Performance evaluation of container-based virtualization for high performance computing environments”. In: *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*. IEEE. 2013.

[22] Z. Kozhirbayev and R. O. Sinnott. “A performance comparison of container-based technologies for the cloud”. In: *Future Generation Computer Systems* 68 (2017).

[23] T. W. Edgar and D. O. Manz. “Chapter 2 - Science and Cyber Security”. In: *Research Methods for Cyber Security*. Ed. by T. W. Edgar and D. O. Manz. Syngress, 2017.

[24] K. S. Wilson. “Conflicts among the pillars of information assurance”. In: *IT Professional* 15.4 (2013).

[25] M. N. Sadiku, S. Alam, and S. M. Musa. “Information assurance benefits and challenges: An introduction”. In: *Information & Security* 36 (2017).

[26] J. H. Saltzer and M. D. Schroeder. “The protection of information in computer systems”. In: *Proceedings of the IEEE* 63.9 (1975).

[27] *Separation of Privilege | US-CERT*. URL: `https://www.us-cert.gov/bsi/articles/knowledge/principles/separation-of-privilege` (visited on 01/31/2019).

[28] R. Shirey. *Internet security glossary, version 2*. Tech. rep. 2007.

[29] S. Jones, M. Wilikens, P. Morris, and M. Masera. “Trust requirements in E-Business: a conceptual framework”. In: *Communications of the ACM* 43.2 (1999).

[30] A. Xu and S. Q. Yang. “Customer Relationship Management as an Imperative for Academic Libraries: A Conceptual Model-121 E-Agent Framework”. en. In: *Handbook of Research on Managing and Influencing Consumer Behavior* (2015).

[31] B. Bailey. “Case studies: A security science research methodology”. In: *Proceedings of the 4th Australian Security and Intelligence Conference*. Australian Security Research Centre, Edith Cowan University, Perth, Western Australia, 2011.

[32] J. Katz and Y. Lindell. *Introduction to modern cryptography*. CRC press, 2014.

[33] B. W. Boehm. “A spiral model of software development and enhancement”. In: *Computer* 5 (1988).

[34] M. Berman, P. Demeester, J. W. Lee, K. Nagaraja, M. Zink, D. Colle, D. K. Krishnappa, D. Raychaudhuri, H. Schulzrinne, I. Seskar, et al. “Future internets escape the simulator”. In: *Communications of the ACM* 58.6 (2015).

[35] U. Zsolt. *Software development processes and software quality assurance*. University of Pannonia, 2014.

[36] V. Mehri and K. Tutschku. “Flexible Privacy and High Trust in the Next Generation Internet: The Use Case of a Cloud-based Marketplace for AI”. In: *Proceedings of 13th Swedish National Computer Networking Workshop (SNCNW)*. Halmstad University, Sweden, 2017.

[37] A. Shostack. *Threat Modeling: Designing for Security*. en-se. Wiley.com, 2014.

[38] P. Runeson and M. Höst. “Guidelines for conducting and reporting case study research in software engineering”. In: *Empirical software engineering* 14.2 (2009).

[39] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[40] K. Tutschku, V. A. Mehri, A. Carlsson, K. V. Chivukula, and J. Christenson. “On resource description capabilities of on-board tools for resource management in cloud networking and NFV infrastructures”. In: *IEEE International Conference on Communications Workshops (ICC)*. IEEE, 2016.

[41] V. Mehri, D. Ilie, K. Tutschku, et al. “Designing a Secure IoT System Architecture from a Virtual Premise for a Collaborative AI Lab”. In: *Workshop on Decentralized IoT Systems and Security (DISS) 24 February 2019, San Diego, CA,* 2019.

[42] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella. "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration". In: *IEEE Communications Surveys & Tutorials* 19.3 (2017).

[43] M. Héder. "From NASA to EU: The evolution of the TRL scale in Public Sector Innovation". In: *The Innovation Journal* 22.2 (2017).

[44] European Parliament and Council of the European Union. "20 General Annexes". In: *HORIZON 2020 – WORK PROGRAMME 2016-2017 General Annexes*. 2017.

[45] N. Lasla, M. Younis, W. Znaidi, and D. B. Arbia. "Efficient distributed admission and revocation using blockchain for cooperative ITS". In: *2018 9th IFIP international conference on new technologies, mobility and security (NTMS)*. IEEE. 2018.

[46] Y. Zhang, L. Cui, W. Wang, and Y. Zhang. "A survey on software defined networking with multiple controllers". In: *Journal of Network and Computer Applications* 103 (2018).

[47] W. Sun, Z. Cai, Y. Li, F. Liu, S. Fang, and G. Wang. "Security and privacy in the medical internet of things: a review". In: *Security and Communication Networks* 2018 (2018).

6

On Resource Description Capabilities of On-Board Tools for Resource Management in Cloud Networking and NFV Infrastructures

Kurt Tutschku ,Vida Ahmadi Mehri, Anders Carlsson, Krishna Varaynya Chivukula, Johan Christenson

Abstract: The rapid adoption of networks that are based on "cloudification" and Network Function Virtualisation (NFV) comes from the anticipated high cost savings of up to 70% in their build and operation. The high savings are founded in the use of general standard servers, instead of single-purpose hardware, and by efficiency resource sharing through virtualisation concepts.

In this paper, we discuss the capabilities of resource description of "on-board" tools, i.e. using standard Linux commands, to enable OPEX savings. We put a focus on monitoring resources on small time-scales and on the variation observed on such scales. We introduce a QoE-based comparative concept that relates guest and host views on "utilisation" and "load" for the analysis of the variations. We describe the order of variations in "utilisation" and "load" by measurement and by graphical analysis of the measurements. We do these evaluations for different host operating systems and monitoring tools.

6.1 Introduction

The anticipated high cost savings of up to 70% in build and operation of "cloudified" and *Network Function Virtualisation (NFV)* based networks [1] make their rapid adoption highly probable, although there is significant scientific criticism on the latter concept [2]. The high savings are founded at first in the use of general standard servers, instead of single-purpose

hardware. However, major additional savings are expected by the flexibility of these servers and by the efficiency of resource sharing on them through virtualisation concepts [3]. The flexibility and efficiency of the new infrastructure are expected to lead to a faster service deployment, thus generating new revenue streams for operators [4]. Simplified, cloudification and NFV enables network functions, such as HTTP-caches, VPN gateways or concentrators, MMEs (Mobility Management Entities in 4G mobile networks) or BRAS (Broadband Remote Access Servers), to be implemented as files rather than in hardware. These files can be executed on standard servers as well as in virtualization environments. So-called *virtual network functions (VNFs)* may now run at arbitrary locations. Practically, virtualisation is achieved by implementing the VNFs in *virtual machines (VMs; also denoted as "guests" which are executed on "hosts")* or by *containers*. These VNFs will be operated in large data centres (DCs), which host a very large number of servers. General virtualisation environments for computer systems, like XEN [5] or KVM [6], are used for the execution of VMs Containers have originated from high performance computing for Cloud services, e.g. Dockers [7]. However, ETSI's Industry Specification Group (ISG) on NFV [8], is currently preferring VMs over containers.

The *economics of scale in DCs* [9] are a key contribution to the cost savings. This feature assumes that it is easy to manage and share computing resources in DCs. The more resources are available in DCs, the more cost-effective the operation becomes. The efficiency of servers in DCs are often described by the terms "load" or "utilisation". These values are readily available as output from standard, so-called "on-board" tools, such as `mpstat` [10] or `top` [11] in modern operating systems. Their availability leads to simple and obviously practical resource management strategies, such as "put as many VNFs on a server until the system reaches a certain load". However, these tools typically haven't been designed with the stringent requirements, e.g. on small time scales, for resource efficiency management.

The aim of the paper is not to explain the nature and fundamentals of load monitoring. The aim is rather on evaluating the applicabilities of standards with respect to load variations resulting from the nature of the tools or of the operating systems. Good overview on load monitoring principles can be found in [12] or [13].

We will show in this paper evidence that a simple understanding of these tools, i.e. without considering their inherent variability, might not be appropriate for efficient resource management in cloudified infrastructures. We will argue that VNFs might require more comprehensive resource usage descriptions, e.g. *VNF profiles* [14]. We also demonstrate that the accuracy of the current on-board tools increases with a high utilisation in guest and host. However, users and performance evaluation typically consider high load as harmful. Thus, VMs are expected to be operated at low load levels and in turn, the accuracy and usefulness of the current load and utilisation monitoring tools in this scenario are in doubt.

We base our discussions on measurements that been carried out in experiments, which are close to the configurations of DCs operated by CityNetwork Webbhotell AB [15]. We will show results that outline the variability of load and utilisation monitoring. We suggest a correlation technique to investigate the variability, which is considered the QoE (as seen by the guest) and the QoS (as seen by the host) [16]. We also do these evaluations for different popular Linux distributions in DCs, such as `Ubuntu` [17] and `CentOS` [18].

We will discuss in Sec. 6.2 the time-scale requirements for resource monitoring, possible techniques for resource descriptions and monitoring, a comparative concept to identify load relationships between guests and hosts, and how to generate load for testing. Sec. 6.3 details the aims of the experiments and the experiment setup. Sec. 6.4 discusses the results, while Sec. 9.6 provides a brief summary and outlook to future work.

6.2 Requirements and Methods for Resources Descriptions

6.2.1 Timescale Requirements for Resource Management

We assume that the success of cloudified and NFV system's origins mainly from their anticipated cost-savings. Costs occurring in networks are usually classified into CAPEX (capital expenditures) and in OPEX (operational expenditures). CAPEX describes mainly the initial non-recurring expenditures in network equipment, infrastructure, hard-, software, buildings, or ground works, which are enjoyed over a long time. CAPEX savings in cloudified and NFV systems are related mainly to the dimensioning of the infrastructure,

e.g. the required number of servers to maintain a defined service level. CAPEX typically accounts for 20% of the total costs of networks [19].

OPEX costs are the ongoing costs of running networks. They comprise consumables (incl. spare parts), utilities (power or cooling), labor cost, but also maintenance and facility expenses. OPEX costs might sum up to 80% of the total costs. They occur regularly and might be controlled instantaneously, i.e. on seconds or minutes, e.g. by shutting down un-used servers and saving their utilities costs. Possible OPEX savings in cloudified and NFV systems are diverse. First, the DCs economics of scale lead to a smaller number of administrators, which need less specialised skills and thus, consequently might incur lower labor costs. Second, the independence from execution location permits operation at venues with lower cost, e.g. lowest energy costs. Finally, fine grained resource management in cloudified systems might lead to reduced energy consumption, e.g. VNFs can be moved to servers which are not fully loaded while unused servers are shutdown.

This short discussion shows that the full potential of cost savings in cloudified systems will probably achieved when OPEX reductions are implemented on short time-scales, i.e. seconds or minutes. Hence, the descriptions and monitoring tools for resource consumption to need to obey these time-scales.

6.2.2 Resource Descriptions

Next, we will outline which resources should be described and how this should be done. Therefore, we will discuss the type and nature of the resources (i.e their semantics) and formal description concepts (i.e. their syntax). The applied description concept will use statistical notations.

6.2.2.1 Resources Categories and Profiles

We describe the resources by assuming that virtualisation is done by the VMs. Hence, we derive the resource categories from the configuration options of the `virt-install` command, which is a typical tools to create a guest [20]. The configuration options can be classified into: a) CPU, b) memory, c) storage, d) input/ output (I/O), e) network, f) operating system and g) other fields, cf. Table 6.1.

In this contribution, we will consider only available and used CPU capacity as system state (either by "utilisation" or "load"). Herby, the "host

Area	Option	Description	Used here
CPU	`-vcpus`	Number of vCPUs	X
memory	`-ram`	RAM to allocate (MB)	
storage	`-disk`	specify storage media	
	`-filesystem`	export a host directory to the guest	
	`-file`	Installation media location (local install)	
	`-location`	Installation via distribution tree (network install)	
network	`-network`	Host network	
operating system	`-os-type`	OS type	
	`-os-variant`	OS variantversion	
I/O	`-graphics, -nographics`	Graphical display method	
Other	`-name`	Virtual machine name	

Table 6.1: Resource categories and configuration options in `virt-install`

capacity" is the *available physical CPU* on the host running the VM and the "guest capacity" is the assigned number virtual CPUs to a guest. However, it is obvious from Table 6.1 that more complex resource descriptions are needed that comprise multiple resource categories. This need has led to the concept of a "VNF profile" [14]. We focus intentionally at CPU only in order to find initial relationships.

6.2.2.2 Sampling Concept and Statistical Characterisations

The requirement for describing the system state in the order of seconds to minutes in cloudified systems sets the scope of the monitoring and sampling concept. We adopt a sampling concept similar to typical steady-state simulation analysis [21]. The sampling concept is depicted in Figure 6.1. It comprises a start-up phase and stop phase in order to reach a state-steady and to avoid interference with the load generation, cf. Section 6.2.4. The inter-sampling interval Δt is currently chosen as constant, but easily be changed to random and independent intervals [21].

The statistical characterisations for status description are considered initial for this paper and might require further improvements. Due to the objective of simplicity of status monitoring, however, we start with simple but major statistical values: average *E[X]*, minimum *Min[X]* and maximum *Max[X]* to quantify the mean and the range of the variations in the sampling

interval. Hereby, X is the random variable of the observed state:

$$E[X] = \sum_{i=1}^{N} x_{s,i}/N \tag{6.1}$$

$$Min[X] = x_{\min} \quad \text{with} \quad \{\forall x_i \in X : x_{\min} \leq x_i\} \tag{6.2}$$

$$Max[X] = x_{\max} \quad \text{with} \quad \{\forall x_i \in X : x_{\max} \geq x_i\} \tag{6.3}$$

$$\sigma[X] = \sqrt{E[(X - E[X])^2]} \tag{6.4}$$

$$cov[X] = \frac{\sigma[X]}{E[X]} \tag{6.5}$$

with x_i is the observed state at $t = t_{\text{start\_collection}} + \Delta t * (i-1)$, $N = \lfloor (t_{\text{stop\_collection}} - t_{\text{start\_collection}})/\Delta t \rfloor$ and $i \in [1, ..., N]$.

6.2.3 Tools for Load and Utilisation Monitoring

6.2.3.1 Notions for CPU Utilisation and CPU Load

CPU load is the concurrent number of processes using the computational resources on a computer. It is computed by summing up the number of running threads and the number of waiting threads. Typically, the CPU load value should be between 0 and the numbers of available CPU cores. This requirement assures that no resources are wasted by queuing of processes. It should be noted that the CPU load output from monitoring tools, see

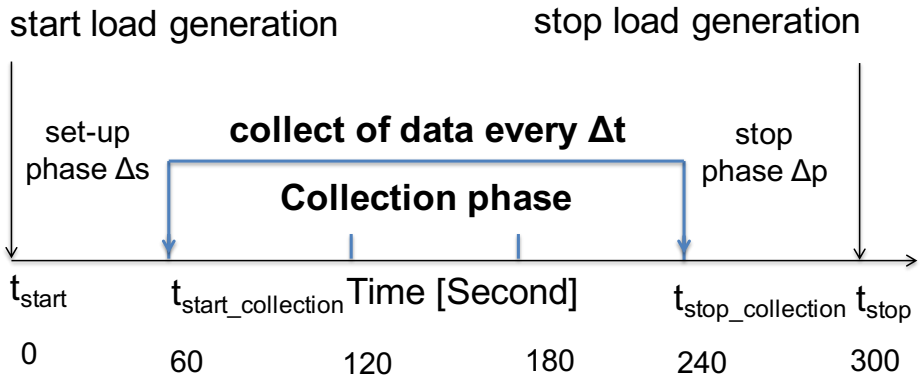

Figure 6.1: Sampling concept for CPU utilisation and load

below, is typically a weighted average of the number of served processes at subintervals during a measurement.

CPU utilisation is denoted by the amount of accumulated time a CPU is busy for handling work during a specific interval. It is reported as a percentage. It can be calculated as the time a CPU is idle, the time a CPU is running a user-level application or the system-level threads for each processor are served, cf. [22].

The CPU utilisation ratio is calculated by dividing amount of accumulated CPU time by the observation time interval of the measurement interval. Almost each tool applies its own calculation and may not necessarily calculate similar to other tools use. For purpose of simplification, we call the CPU utilisation ratio simply as CPU utilisation.

6.2.3.2 Tools

MPstat

The tool was used for monitoring utilisation and is part of the `sysstat` utilities [10]. It reports individual or combined processor statistics. The used `MPstat` command is: `mpstat -P ALL T1 T2`. The option `-P ALL` specifies that the usage of all processors will be monitored. The value `T1` details the inter-sampling interval and corresponds to the Δt in Figure 6.1. The parameter `T2` specifies the duration of the measurement. `MPstat` has a multi-column output, comprising the CPU number, percentages of CPU utilisation at `%user/%system/%iowait` levels, and `%idle` ratio. We used in our monitoring the value of (100 − `%idle`), since the `%idle` value was reported in percentages.

top

We used the tool to monitor load and utilisation [11]. It is the default tool in `Linux` distributions for real-time view of a system. It provides a summary and details a list of processes which are handled by the `Linux` kernel. The used `top` command is: `top -b -d T1 -n T2`. The option `-b` starts a batch for monitoring and it is used for sending output to a file. The value `-d T1` provides the sampling interval `T1`, which again corresponds to Δt in Figure 6.1. The parameter `-n T2` specifies the number of samplings. `top`

shows the load average of 1, 5, and 15 minute intervals. We consider the 1-minute-average in our measurements.

6.2.4 Load Generation

Imposing load and stress on computer systems has regained popularity with the improved resource sharing capabilities of virtual systems. Such stress tests aim at high loads and can be used for performance fine-tuning, testing of physical and virtual hardware components, or measuring power consumption, cf. [23]

For our investigations will use the `stress-ng` tool, which is a clean room implementation of the original `stress` tool, cf. [24]. It offers a wide range of tests for CPUs, storage and drive systems, I/O syncs, memory and processes.

Since our investigation focuses on load and utilisation of the physical/virtual CPUs or cores, we used the CPU test as defined by the `-c N, -cpu N` options. The used command was: `stress-ng -c N1 -l P -t N2`. The value `N1` is the number of processes, which load the (virtual)CPUs by calculating the computational demanding function `sqrt((double)rand())`. The value `P` specifies the load imposed on the CPU by a process and is given in percentage (%). It ranges between 0 (=sleep) and 100 (=full load). The value `N2` stops the stress test after $N2$ seconds.

6.2.5 Comparative Investigation of Load and Utilisation Relationships

A major aim of resource management in cloudified infrastructures is to reduce operational costs by increasing load and utilisation while maintaining expected performance. Levelling resource consumption and performance is already difficult in non-virtualised systems. However, finding such trade-offs in virtualised systems is even more complex due to the higher number of types of resource, their complex interactions (dependencies of network throughput on CPU capacity due virtual hardware) and due to the nesting of virtual resources in physical ones.

In order to make the analysis, independent from the type of the resource, relationships among resource categories and from nesting, we define a *comparative concept* to relate aims and relationships of users and operators in virtual infrastructures. The comparative concept is a kind of "black box"

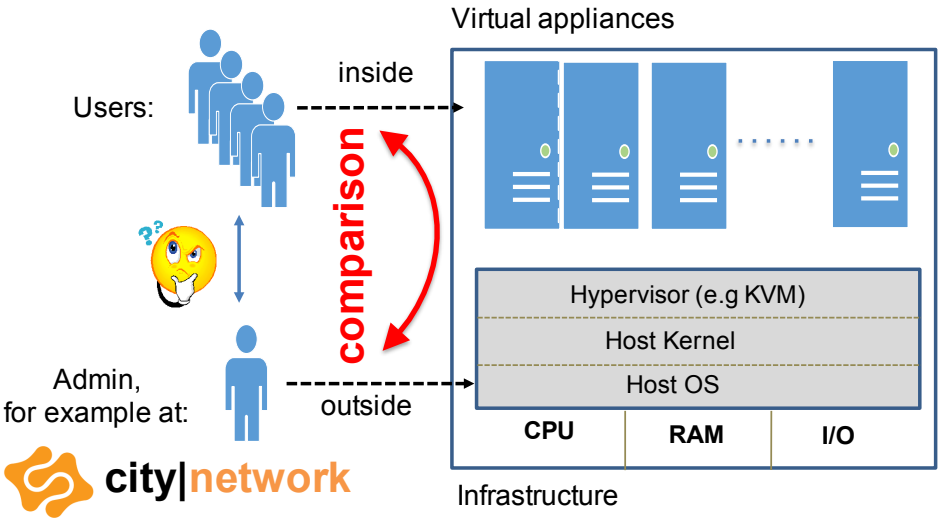

Figure 6.2: Comparative Concept to Investigate and Model the Utilisation and Load Relationship in virtual Environments

approach [25] for system analysis. Hence, it doesn't require any detailed knowledge about the investigated resource (sharing) mechanisms.

6.2.5.1 The Comparative Concept

The comparative concept correlates the views of a user of VM (i.e. a person or entity observing the VM's performance) and of the operator (i.e. administrator) surveying the physical infrastructure. We call the view from within the VM the "inside" or "Guest" view and the view on infrastructure as the "outside" or "Host" view. The aim of the concept is to compare the "inside" with the "outside" and to identify correlations between these views. The described the concept is depicted in Figure 6.2. The compact is a derivative of methods to investigate the QoE/QoS relationship in networks [16, 26].

6.2.5.2 Correlation Analysis Using Scatter Plots

The "inside/outside" relationships researched in this contribution are analysed by simple but powerful *scatter plots* [27]. Scatter plots are two-dimensional diagrams using Cartesian coordinates to display a set of points. These points

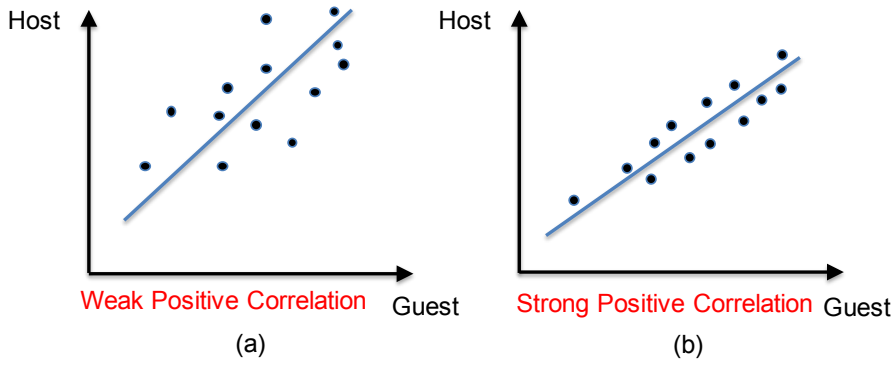

Figure 6.3: Scatterplots to Compare Guest and Host Relationships

are defined by a tuple (x_t, y_t), i.e. a pair of concurrently observed variables X and Y at time t.

The tuple $(sample_{\text{inside/ Guest},j}, sample_{\text{outside/Host},j})$ is the pair of simultaneous load or utilisation measurements at guest and host at time j. Examples of correlations identifiable by scatter plots are shown in Figure 6.3. Figure 6.3(a) shows a weak positive correlation, i.e. the points are loosely around the increasing line, and Figure 6.3(b) depicts a strong positive correlation, i.e. the points are quite close to the line.

6.3 Experiments

6.3.1 Aims and Research Questions

The experiments carried out with respect to these questions:

1. Is there a predictable, i.e. strong positive correlation of load and utilisation between guest and host? Here, "scaling" means, whether an increase in guest load results in a predictable increase of the host.
2. How strong do the observed utilisation values vary when observing them during a small sampling interval?
3. Does the accuracy of the prediction depend on guest load levels?
4. Does the scaling depend on the number of stressed vCPUs, i.e. on the resources available to the VM?

5. Does the scaling depend on host operating systems?

6. Are there significant differences between the monitoring tools?

6.3.2 Experiment Setup

We carried out the experiments on a single standalone server that is similar to the ones operated in DCs of CityNetwork Webhotell AB. The key host and guest specifications are depicted in Table 6.2. The server was running either `Ubuntu 12.04 LTS Desktop` [17] or `CentOS 6.6 Desktop` [18]. In both cases we used `kvm` [6] as the virtualisation environment. The guests were allocated two virtual CPUs, 2GB of virtual RAM, and 30GB of virtual disk. This assigning is based on the KVM provisioning model, which is based on CPU cores. The guest used `Ubuntu 14.04 LTS Server`.

Criteria	Specification
CPU	Intel ® Xeon ® CPU EC-1230 v2 @ 3.30GHz 8 cores
RAM	8GB
Hard disk	500 GB SAS
Hypervisor	KVM
Host OS	CentOS 6.6 Desktop / Ubuntu 12.04 LTS Desktop
Guest OS	Ubuntu 14.04 LTS server

Table 6.2: Specifications for host and guest.

6.3.3 Common Experimentation Parameters

Throughout the experiments, we increased utilisation in the guest by using the `stress-ng` command in 10% step from 10% to 100%. Each step was iterated and measured at least 50 times and the load was imposed for 5min in order to have a small time-scale. Finally, the set-up and stop phases Δs and Δp were 1min and the inter-sampling interval Δt was 1sec, cf. Figure 6.1. Guest and host did no other computing task other than standard OS tasks.

6.4 Results

6.4.1 Predictability of Scaling

The predictability of the scaling of utilisation was observed using `mpstat` and is depicted in Figures 6.4 and 6.5. Figure 6.4 shows if the VM is loaded with 50% (i.e. 100% load from one process) then the host is loaded at about

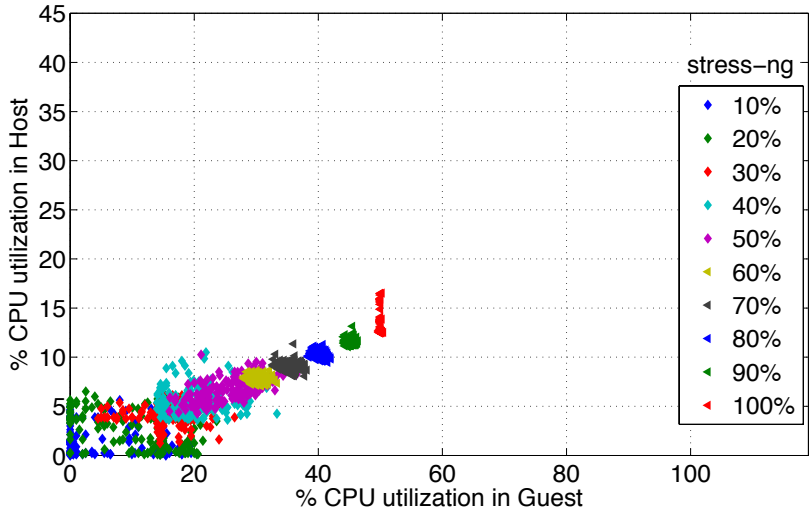

Figure 6.4: Relationship for utilisation; one process loading on guest; host: Ubuntu; tool: mpstat

12%. This behaviour can be expected for a eight core host and for a VM, which is configured with two vCPUs. However, the figures also reveal that the observed utilisations are not deterministic for the load levels and vary strongly. Hence, we conclude that there is predictable scaling, i.e. positive correlation, but this is weak and even weakens when the load on the guest is reduced. Tables 6.3 and 6.4 provide the numerical values to the statistical characteristics in Figures 6.4 and 6.5. These values shows that `stress-ng` is accurate in generating the expected load E[X] on the guest (in %; E[] is abbreviated by ∅ in the tables) but the *cov* is similarly high for guest and host (above 0.2) for guest loads less than 30%. Hence, we conclude that the prediction accuracy decreases significantly with lower guest load.

The measurements show that monitoring of utilisation is possible on small time-scales only as long as the VM is highly loaded. Moreover, the uncertainty might increase when stressing multiple vCPUs. This is evidenced higher *cov* values on the guest and host for two stressing process and guest utilisation below 20%. This observation shows that the accuracy depends on the guest load level, but also on the type of guest load. Hence, this might indicate that a more detailed description load is needed for describing VNFs. Thus, the suggest VNF profiles [14] need to address the load type as well.

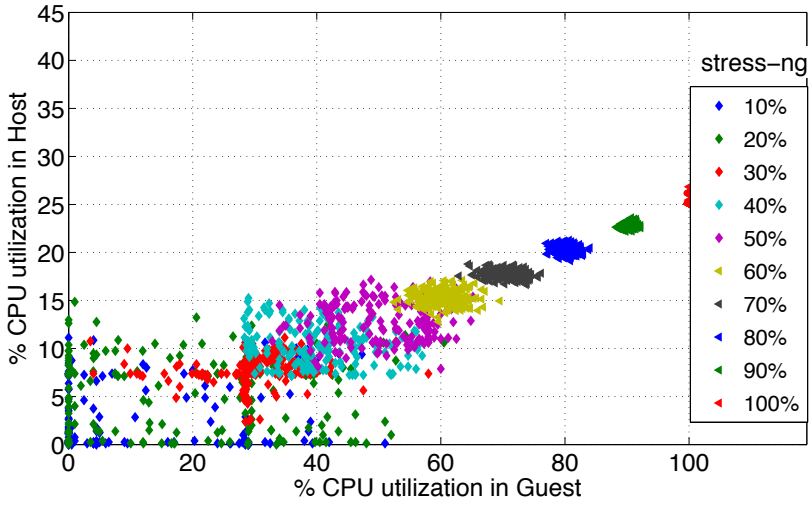

Figure 6.5: Relationship for utilisation; two processes loading on guest; host: Ubuntu; tool: mpstat

	Guest [X]				Host [Y]			
%	∅	cov	min	max	∅	cov	min	max
10	5,20	1,33	0,00	20,6	1,68	1,04	0,00	5,62
20	11,0	0,64	0,00	23,6	2,95	0,62	0,12	6,50
30	15,8	0,24	4,50	26,5	4,11	0,22	1,25	6,25
40	20,5	0,20	14,0	33,3	5,57	0,24	3,63	10,5
50	25,3	0,17	16,0	36,5	6,66	0,18	4,26	10,3
60	30,4	0,04	28,0	33,0	7,79	0,04	7,07	8,70
70	35,5	0,03	32,7	38,2	9,02	0,05	8,08	11,4
80	40,2	0,21	38,3	42,9	10,3	0,04	9,47	11,3
90	45,3	0,10	44,1	46,5	11,5	0,24	11,1	13,2
100	50,1	0,00	49,8	50,5	13,0	0,07	12,5	16,5

Table 6.3: Characterisation of utilisation at guest and host; one loading process; host: Ubuntu; tool: mpstat

6.4.2 Predictability of Load

The predictability of the scaling of load monitored by `top` is depicted in Figures 6.6 and 6.7. They confirm a similar weak positive correlation as for utilisation. However, other effects that lead to variations and which are systematic, are also observable. For example, the load at 100% in Figure 6.6 seems to be asymptotic. These effects need further research.

	Guest [X]				Host [Y]			
%	∅	cov	min	max	∅	cov	min	max
10	10,8	1,33	0,00	51,0	2,85	1,22	0,00	11,1
20	20,4	0,81	0,00	60,5	5,37	0,75	0,00	14,9
30	30,5	0,25	3,5	58,0	7,88	0,20	2,38	11,1
40	40,3	0,19	28,3	57,3	10,4	0,19	7,13	15,3
50	50,3	0,15	32,8	65,8	12,8	0,16	7,88	17,2
60	60,4	0,05	52,5	69,5	15,3	0,05	12,8	17,1
70	70,2	0,04	62,9	76,4	17,8	0,03	16,7	18,9
80	80,2	0,02	77,1	84,0	20,3	0,02	19,1	21,3
90	90,1	0,01	88,3	92,2	22,8	0,01	22,2	23,8
100	100	0,00	100	100	25,3	0,02	25,0	26,9

Table 6.4: Characterisation of utilisation at guest and host; two loading processes; host: Ubuntu; tool: mpstat

	Guest [X]				Host [Y]			
%	∅	cov	min	max	∅	cov	min	max
10	8,11	1,01	0,00	28,5	3,96	0,73	0,06	13,6
20	15,5	0,50	0,00	32,8	6,98	0,48	0,81	16,3
30	30,5	0,18	11,3	43,3	9,57	0,20	3,51	17,6
40	40,4	0,12	30,1	54,4	12,0	0,17	8,69	20,2
50	50,4	0,12	34,4	67,8	14,4	0,16	10,1	22,1
60	60,2	0,05	53,3	66,9	15,8	0,06	13,8	20,5
70	70,3	0,09	54,0	84,4	26,2	0,45	16,5	49,9
80	80,2	0,02	75,5	84,5	21,2	0,05	19,8	25,0
90	90,1	0,01	88,4	92,1	23,3	0,03	22,4	25,8
100	100	0,00	100	100	26,7	0,06	25,4	33,4

Table 6.5: Characterisation of utilisation at guest and host; two loading process; host: centOS; tool: mpstat

6.4.3 Dependency on Host OS

The dependency on the host operating system can be analysis by comparing Figures 6.5 and 6.8. Figure 6.8 and the related statistical characterisations in Table 6.5 reveal that `CentOS` has typically higher *cov* at higher utilisation values than `Ubuntu`. Although `CentOS` is typically considered as a better choice as a host operating system, our measurements show that it exhibits a higher variation and in turn a lower accuracy when describing utilisation. Therefore, `Ubuntu` seems to be a better choice when it comes to load optimisation.

6.4.4 Differences Between Tools

The comparison Figures 6.4 and 6.5 for utilisation and Figures 6.6 and 6.7 for load indicates that there might be an advantage for the `top` tool due lower visual variations. However, the advantage might be doubtful due to

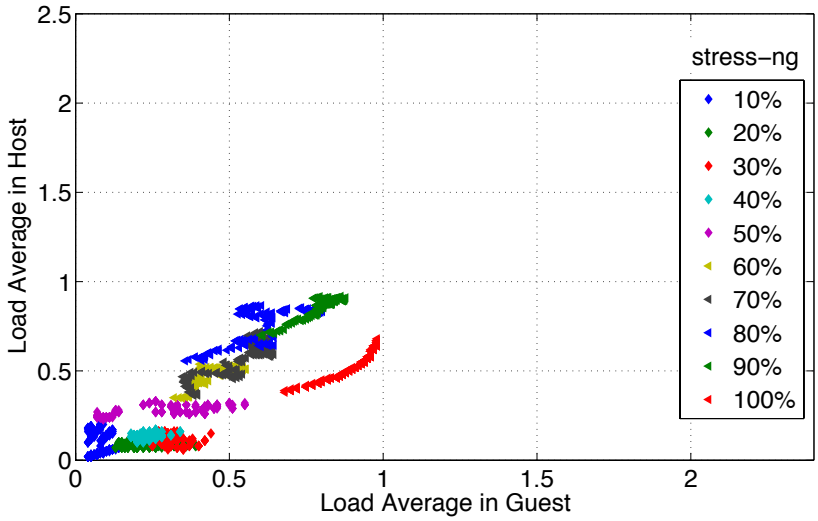

Figure 6.6: Relationship for load; one process loading on guest; host: Ubuntu; tool: top

the apparent systematics in the load relationships. Hence, the comparison of the tools requires further research.

6.5 Conclusion

This contribution outlines the capabilities of on-board Linux tools for describing of resource utilisation in cloud networking and NFV infrastructures. We argued that resource descriptions, monitoring and management need to address small time-scales in order to obtain high OPEX savings. Furthermore, we introduced for the first time a QoE-based comparative concept to relate guest and host views for utilisation and load.

We have shown by this concept that there is a predictable but weakly correlated relationship for utilisation and load between host and guest VM. Moreover, the accuracy decreases for low guest load levels. Hence, resource utilisation on the host might be overestimated , i.e. the infrastructure might not exhibit the required efficiency. Accurate monitoring on a small, i.e. 5min, time-scale is possible at high load level, but more accurate monitoring is suggested at low load levels. Our measurements outlined significant

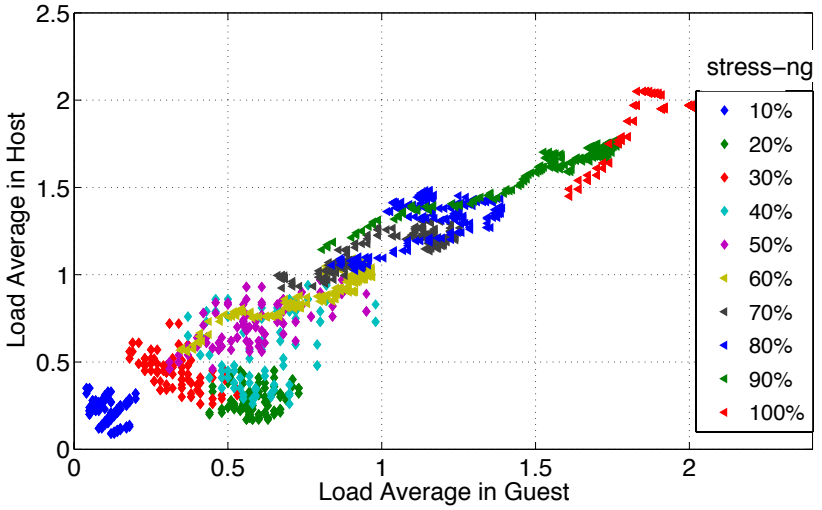

Figure 6.7: Relationship for load; two processes loading on guest; host: Ubuntu; tool: top

differences in scaling and predictability between host operating systems and between tools. Hence, `Ubuntu` seems to be a more appropriate choice as host OS when it comes to resource management. The comparison of tools, however, needs further research.

The level, but also the kind of stress, i.e. in terms of stressing processes, determines the utilisation of the infrastructure. Hence, future research should be devoted to identifying what kind of NFV profiles and benchmark loads should be defined!

References

[1] ETSI. *Network Functions Virtualisation – An Introduction, Benefits, Enablers, Challenges & Call for Action*. Information available at http://portal.etsi.org/NFV/NFV_White_Paper.pdf. 2012.

[2] J. Crowcroft. *j'accuse NFV*. Post to E2E mailing List, Information available at http://http://www.postel.org/pipermail/end2end-interest/2015-April/009289.html. 2015.

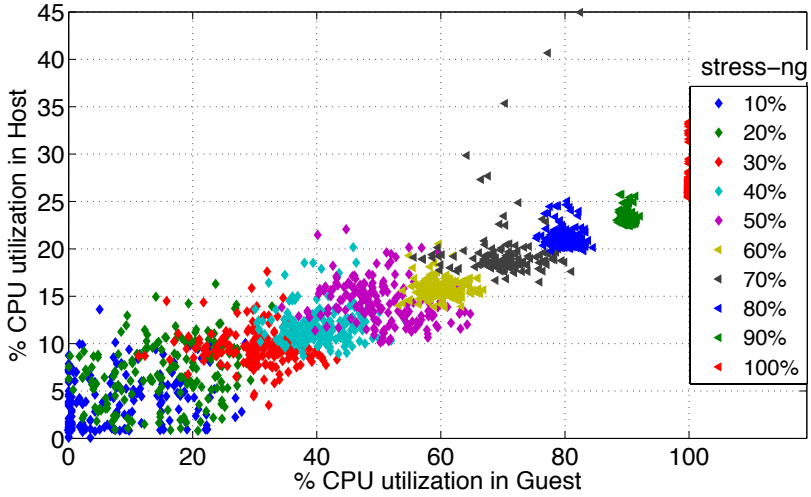

Figure 6.8: Relationship for utilisation; two loading processes on guest; host: CentOS; tool: mpstat

[3] B. Chatras. *Network Functions Virtualisation (NFV); Virtualisation Technologies; Report on the Application of Different Virtualisation Technologies in the NFV Framework*. ETSI GS EVE 004 V0.2.0 – draft available at `https://docbox.etsi.org/`. 2015.

[4] AT&T Inc. *AT&T Vision Alignment Challenge Technology Survey – AT&T Domain 2.0 Vision White Paper*. Information available at `http://www.att.com/Common/about_us/pdf/AT&T%20Domain%202.0%20Vision%20White%20Paper.pdf`. Nov. 2013.

[5] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. "Xen and the art of virtualization". In: *ACM SIGOPS Operating Systems Review* 37.5 (2003).

[6] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori. "kvm: the Linux virtual machine monitor". In: *Proceedings of the Linux Symposium*. Vol. 1. 2007.

[7] D. Merkel. "Docker: Lightweight Linux Containers for Consistent Development and Deployment". In: *Linux Journal* 2014.239 (Mar. 2014). ISSN: 1075-3583.

[8] ETSI. *Network Functions Virtualisation – Collaborative Portal*. Information available at http://portal.etsi.org/portal/server.pt/community/NFV/367. 2013.

[9] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel. "The Cost of a Cloud: Research Problems in Data Center Networks". In: *SIGCOMM Comput. Commun. Rev.* 39.1 (Dec. 2008).

[10] S. Godard. *Sysstat Utilities Home Page!* Information and code available at `http://sebastien.godard.pagesperso-orange.fr/index.html`. 2015.

[11] W. LeFebvre. *Top*. Information and code available at `http://sourceforge.net/projects/unixtop/`. 2003.

[12] R. Walker. "Examining Load Average". In: *Linux J.* 2006.152 (Dec. 2006). ISSN: 1075-3583.

[13] A. Lewis. *Understanding Linux CPU Load - When should you be worried?* Information at `http://blog.scoutapp.com/articles/2009/07/31/understanding-load-averages`. 2009.

[14] R. Rosa and C. Rothenberg and R. Szabo. *VNF Benchmark-as-a-Service – draft-rorosz-nfvrg-vbaas-00*. Information available at https://www.ietf.org/internet-drafts/draft-rorosz-nfvrg-vbaas-00.txt. 2015.

[15] City Network Hosting AB. *Agility Through Open IT-Infrastructure*. Information at `https://www.citynetworkhosting.com`. 2015.

[16] M. Fiedler, K. Tutschku, and P. Carlsson. "Identification of Performance Degradation in IP Networks Using Throughput Statistics". In: *the 18th International Teletraffic Congress - ITC18*. Berlin, Germany, Sept. 2003.

[17] Canonical Ltd. Ubuntu. *Ubuntu*. Information at `http://www.ubuntu.com/`. 2015.

[18] The CentOS Project. *CentOS*. Information at `https://www.centos.org`. 2015.

[19] CISCO Inc. *Network Procurement: The Journey from CAPEX through TCO to Business Value*. White paper available at http://www.cisco.com/c/dam/en/us/solutions/collateral/executive-perspectives/executive-perspectives/network-procurement-cfo.pdf. 2012.

[20] Anonymous. *libvirt – The virtualization API*. Information and code available at `http://libvirt.org`. 2015.

[21] J. Banks, J. C. II, B. Nelson, and D. Nicol. *Discrete-Event System Simulation*. Upper Saddle River, NJ: Prentice Hall, 2009.

[22] B. Gregg. *Systems Performance: Enterprise and the Cloud*. Pearson Education, 2013.

[23] V. Gite. *How To Stress Test CPU and Memory (VM) On a Linux and Unix With Stress-ng*. nixCraft – Linux Howto's Guide – Information available at http://www.cyberciti.biz/faq/stress-test-linux-unix-server-with-stress-ng/. Sept. 2015.

[24] C. King. *stress-ng*. GitHub – Information and software available at https://github.com/ColinIanKing/stress-ng. Nov. 2015.

[25] R. Patton. *Software testing*. Sams Pub., 2006.

[26] T. Hossfeld, D. Hock, P. Tran-Gia, K. Tutschku, and M. Fiedler. "Testing the IQX Hypothesis for Exponential Interdependency between QoS and QoE of Voice Codecs iLBC and G.711." In: *18th ITC Specialist Seminar on Quality of Experience*. Vol. s. 105-114. 2008.

[27] W. Cleveland. *Visualizing Data*. Hobart Press, 1993. ISBN: 0963488406.

7

Flexible Privacy and High Trust in the Next Generation Internet: The Use Case of Cloud-based Marketplace for AI

Vida Ahmadi Mehri, Kurt Tutschku

Abstract: Cloudified architectures facilitate resource access and sharing which is independent from physical locations. They permit high availability of resources at low operational costs. These advantages, however, do not come for free. End users might fear that they loose control over the location of their data and, thus, of their autonomy in deciding to whom the data is communicate to. Thus, strong privacy and trust concerns arise for end users.

In this work we will review and investigate privacy and trust requirements for Cloud systems in general and for a cloud-based marketplace (CMP) for AI in particular. We will investigate whether and how the current privacy and trust dimensions can be applied to Clouds and for the design of a CMP. We also propose the concept of a "virtual premise" for enabling "Privacy-by-Design" [1] in Clouds. The idea of a "virtual premise" might probably not be a universal solution for any privacy requirement. However, we expect that it provides flexibility in designing privacy in Clouds and thus leading to higher trust.

7.1 Introduction

Clouds became key elements of our daily lives. They are changing our social and economic behaviours. Today's Clouds are mostly infrastructure-centric, e.g. cheap storage solutions such DropBox [2]. The next generation of Clouds will be designed particularly for humans (cf. European Community (EC) Next Generation Internet initiative [3]) and should have mechanisms build-in that support social needs of users, such as privacy.

Clouds gain their benefit from the separation of resource location and resource availability. This permits automatic and autonomic processing, resource sharing, transmission, re-organisation or replication of data and resources. However, this characteristic leads also to concerns by end users: do I loose control over my data? Such concerns even become more importance when *digital trading* and a *digital economics* is considered [4].

Privacy issues determine the adoption of Clouds and have a direct impact on the growth of Cloud service providers[5]. TRUSTe reported in the "Consumer Privacy Index 2016" that 92% of US Internet user worry about their online privacy and 89% avoid a company that do not maintain privacy [6].

The General Data Protection Regulation (GDPR) [7] was formulated by the EC as a legal framework that specifies which end user requirements must to address by service provider. This regulation requires service provider to answer user questions such as: where is my data located, can other people read my data, is the access to my data controlled or traced, or can I revoke my permissions to use my data?

Engineering appropriate privacy levels for a specific application is challenging. Different applications might require different privacy levels. This challenge has lead, among other reasons, to the concept of "Privacy-by-Design (PbD)" [1]. PbD calls for an early consideration of privacy requirements in systems design. However, Clouds aim at general solutions while PdD might call for a specific solution. Hence, the question arises how can one design Clouds that support in a flexible way application-specific privacy levels?

Privacy is a broad concept and difficult to design. We will apply the concepts of *privacy and trust dimensions* for providing a more detailed specification of the requirements. We will use these dimensions and extend them towards Clouds. Furthermore, we will consider the use case of a CMP for Artificial Intelligence (AI) [8, 9] for the investigation of the dimenssions. In addition, we use this example for the specification of the concept of a "virtual premise". This concept aims at facilitating the idea of PbD.

In this work, Section 7.2 provides definitions for the terms of *privacy* and *trust* and specifies the considered *privacy and trust dimensions*. Section 7.3 discusses the privacy requirements of a CMP, introduces the concept of PbD, and provides a first gap analysis for privacy in a CMP. Section 7.4 introduces the concept of a "virtual premise" for addressing privacy issues. Finally,

Section 9.6 summarises our findings and provide an outlook on future work.

7.2 Definition of Privacy and Trust

Next, we present the existing definition and dimensions of privacy and trust.

7.2.1 Privacy Definitions and Privacy Dimensions

Westin defines *privacy* as *" the claim of individuals...to determine for themselves when, how and to what extent information about them is communicated to others ..."*[10]. R.S. Poore mentioned privacy as a required context of *Personal Identifiable Information (PII)* which have to be under control of the individual person who is the owner of it [11]. S. Guilloteau and V. Mauree consider privacy as *"... the right of individual to know what is known about them, be aware of stored information about them, control how that information is communicated and prevent its abuse."*[5] in the "privacy in cloud computing" report.

These definitions introduce different privacy dimensions[12, 13]:

a) *Identity privacy*, information that lead to the ability to identify a specific person

b) *Location privacy*, information that disclose the location of a person

c) *Communication privacy*, information that may breach the confidentiality when exchanging data

d) *Access privacy*, refers to control of access privileges

e) *Data processing privacy*, refers to information about the information flow in processing and dissemination of data

7.2.2 Trust Definitions and Trust Dimensions

IETF Internet security glossary defines *trust* as *"... the extent to which someone who relies on a system can have confidence that the system meets its specifications, i.e., that the system does what it claims to do and does not perform unwanted functions"*[14]. J. Huang[15] summarise trust definition for cloud computing as a *" mental state constitutes of expectancy, belief, and willingness to take a risk."* Trust dimensions[13] will summarise as following:

a) *Device trust*, refers to the reliability of devices to produce data correctly

b) *Operation trust*, considers traceability of data and analytics usage

c) *Communication trust*, builds on confidentiality, integrity, and authenticity in data transmission

d) *Infrastructure trust*, aims at the transparency and predictability of processing.

7.3 PbD Requirements for the Use Case of a CMP for AI

In this section we provide a description about the use case and discuss its privacy PbD requirements.

7.3.1 Description of the Use Case — CMP for AI

The MPs for AI resources, which is suggested by the H2020 project "Bonseyes" [8, 9], provides a platform for engaging various stakeholders, e.g. data providers, model, or application designer, into business among AI resources, i.e. data sets, models, training facilities, etc. This MP will be implemented in the Cloud in order to facilitate between different business models with massive amounts of data. The MP permits that AI models can be moved around. They can be trained and used at various compute facilities depending on the available computing power and the scope of the application.

A CMP, like any real world MP, requires mechanism to enforce privacy and trust. However, these mechanisms are more challenging than the ones in non-virtualised system due to separation of resource location and availability.

Another challenge for providing privacy and trust is the lack of standard definitions for these terms in Clouds. The general concept of aforementioned definitions are the right of individuals to control their data. Clouds, however, operate autonomously without individual interference. Therefore, we need to specify to what extent an individual would like to control the privacy of its data and how this extend can be enabled and protected in a CMP.

7.3.2 Privacy by Design — PbD

A. Cavoukian defined the concept of PbD in the 1990s [1]. She deduced it from the fundamental human rights of informational self-determination. PbD aims at considering privacy needs throughout the whole engineering process. In particular, privacy needs and mechanisms should be considered at early stages in the design.

In 2009, it was observed that general solutions for protecting privacy in Clouds are limited. The stakeholders in Clouds need to consider case-based requirements for engineering privacy [16]. PbD principles in WP168 [17] provide guidelines to evaluate PbD requirements for processing data during the planning and design stage. The general PbD aspects are:

1. *Data minimisation*: system providers should avoid collecting PII data or use it as less as possible

2. *Controllability*: system providers have to provide efficient control over PII to data subjects

3. *Transparency*: data subjects have to be informed about systems operation

4. *User Friendly systems*: system should have privacy functions that are easy to use by unexperienced users

5. *Data confidentiality*: system providers have to ensure that only authorised users have access to PII

6. *Data quality*: system providers have to guarantee the technical quality of data and they have to be able to proof this quality for lawful purposes.

7. *Use Limitation*: system providers have to ensure isolation of data and processing in multi-tenant and virtualised system.

7.3.3 Requirements and Gap Analysis for Privacy

After reviewing PbD, privacy and trust definitions and dimensions, we analysis them in the context of a CMP.

A CMP may host different types of AI resources like datasets and models with various level of sensitivity in different location. In general, PbD is applied for data sets with different protection levels. The level depends on the importance of the types of PII stored in it. For example medical records need the highest protection level while weather forecast does not. We introduce three privacy levels:

- *High level privacy:* for the private resources that have PII like medical records and all seven PbD aspects (see previous section) should be addressed.
- *Medium level privacy:* applied for licensed datasets and applications which do not required minimisation of data.
- *Low level privacy:* apply to public datasets and models which are open access.

The trust dimensions can be used to specify the privacy and trust challenges for a CMP that need to be addresses by the infrastructure provider that runs the CMP:

1. Traceability of data sets and models: A CMP needs to prove its reliability and the quality of AI resources. Here, traceability provides operation trust and device trust. However, these dimensions are typically not well addressed in Cloud architectures.
2. Controllability: A CMP should be able to control privileged access to the data sets and models. It needs to provide access based on requests with session time limitation. A CMP should record the access to resources, including time and duration. It should be be able to generate access reports when needed. These functionalities are often not addressed in Cloud systems.
3. Location privacy: A CMP is losing location awareness by its default cloud nature. This kind of privacy would be required for establishing device and infrastructure trust. However, it is yet not clear whether these features [18, 19] need to be considered in a CMP.

7.4 Virtual Premise

7.4.1 Concept

We suggest the concept of a *Virtual Premise* in order to allow for flexible privacy and to facilitate the concept of PbD. A *Virtual Premise (VP)* provides an area where privacy measures are enforced in a Cloud, i.e. where controlled access and monitoring of data usage is executed. The enforcement should be similar to physical premises, where access control is implemented by observable physical means or actions, e.g. locks or perimeter fences. The controllability and traceability of the access facility PbD for the users.

The VP aims at a fine-grained monitoring of data usage at distributed locations. The data access control is enforced in the VP by using *data encryption*.

7.4.2 Encryption and Privacy Tracker

The concept of a VP is closely intertwined with the idea of a "Encryption & Privacy Tracker" (EPT). The EPT establishes a network of trust among the involved entities, permits secure exchange of keys and software for data encryption and decryption, monitors the key and software distribution, records the data usage, and it can revoke execution and access right for decryption software by suspending the software licence for the decryption module. The EPT tracks the usage similar to [20]. However, it enhances the tracking concept by providing means for strong data encryption. Only encryption and decryption modules are requesting and providing data access, i.e. access is only permitted to entities in a trusted group of compute entities.

The EPT sets up a network of trust among processing entities in a trusted group. These entities are required to apply Trusted Platform Modules (TPMs) [21]. The use of TPM permits that only trusted software modules are allowed en- and de- crypt data. The trusted modules can be implemented such that the EPT can enforce them to stop their execution, e.g. when access rights are revoked. Furthermore, a regular checking of the modules with ETP can be required. This checking permits time-based licensing model for the decryption software similar to DRM models [22]. The use of TPMs also supports the mitigation of malicious tampering of the encryption and description software.

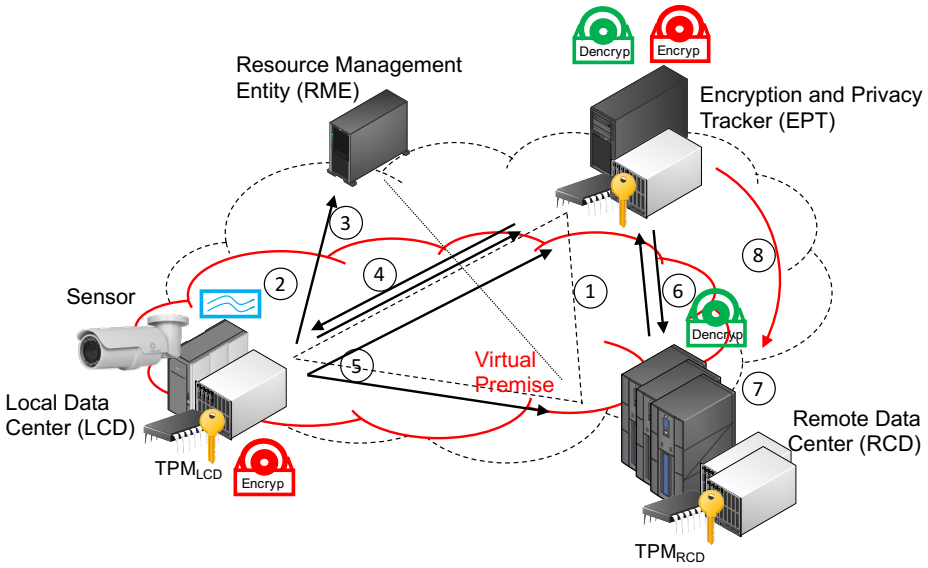

Figure 7.1: The Concept to a Virtual Premise using a Encryption & Privacy Tracker (EPT)

7.4.3 Implementation and Operation of the Virtual Premise

The implementation and operation of a Virtual Premise is shown in Figure 7.1. The operation comprises the following steps:

1. A network of trust among data centers (DCs) that use TPMs is established EPT.
2. The data collected at a Local DC (LDC) is watermarked for traceability.
3. If a Virtual Premise needed, i.e. data usage has to be extend to Remote Data Center (RDC), then LDC checks the Resource Management Entity (RME) for a available RDCs with trusted TPMs
4. The LDC requests the encryption program from the EPT with keys from RDC
5. The LDC encrypts the data, sends it to RDC, and updates the EPT about having forwarding the data
6. The RDC receives the encrypted data and requests decryption software from from EPT

7. The RDC uses a trusted time source, decrypts the data and informs regularly the EPT about a) the kind of data usage and b) the session time for the decryption.

8. If the session time has expired or a user revokes permissions then the ETP forces the software modules at trusted location to cease operation.

The concept of the VP is only used when local computation at the LDC is not sufficient and data needs to be transferred to a remote location, i.e. RDC. In this case the local premise is extend by the VP. Various encryption and decryption modules will offer by VP based on the data types and its privacy level. This possibility will facilitate the concept of PbD in distributed location of CMP.

7.5 Conclusion

Establishing privacy and trust and enabling PbD in the engineering of a CMP is difficult. However, we outlined in this paper that a flexible PbD design is possible when considering appropriate privacy and trust dimensions. The privacy and trust dimensions are roughly defined for Cloud systems, but a very vaguely defined for CMPs. Here, a more detailed investigations and specifications of the privacy and trust dimensions are needed.

The concept of a "Virtual Premise" permits for flexibility in designing privacy in a distributed and virtualised environment. We think that the VP is an appealing concept for increasing trust in Clouds due to the PbD aspects of "controllability" and "transparency".

In future, we will implement the concept of a VP in a real world environment and provide a deeper investigations of its privacy capabilities. In addition, we will investigate the security concerns, e.g. what happens when the EPT is corrupted and how can one protect this element?

Acknowledgment: This project has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 732204 (Bonseyes). This work is supported by the Swiss State Secretariat for Education Research and Innovation (SERI) under contract number 16.0159. The opinions expressed and arguments employed herein do not necessarily reflect the official views of these funding bodies.

References

[1] A. Cavoukian. "Privacy by Design [Leading Edge]". In: *IEEE Technology and Society Magazine* 31.4 (2012). ISSN: 0278-0097. DOI: 10.1109/MTS.2012.2225459.

[2] Dropbox Inc. *Dropbox*. Information available at https://www.dropbox.com. 2017.

[3] European Commission and D. Overton. *Next Generation Internet Initiative Consultation— Final report*. Information available at https://ec.europa.eu/futurium/en/next-generation-internet/document. Mar. 2017.

[4] The E15 Initiative. *Strengthening the Global Trade and Investment System for Sustainable Development*. Information available at http://e15initiative.org/. 2017.

[5] S. Guilloteau and V. Mauree. "Privacy in Cloud Computing". In: *ITU-T Technology Watch Report* (Mar. 2012).

[6] *2016 TRUSTe/NCSA Consumer Privacy Infographic - US Edition*. https://www.truste.com/resources/privacy-research/ncsa-consumer-privacy-index-us/. URL: https://www.truste.com/resources/privacy-research/ncsa-consumer-privacy-index-us/ (visited on 04/06/2017).

[7] European Parliament and Council of the European Union. "Regulation (EU) 2016/679 of the European Parliament and of the council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)". In: *Official Journal of the European Union*. Vol. 59. L 119. ISSN: 1977-0677. May 2016.

[8] T. Llewellynn, M. Fernández-Carrobles, O. Deniz, S. Fricker, A. Storkey, N. Pazos, G. Velikic, K. Leufgen, R. Dahyot, S. Koller, et al. "BONSEYES: platform for open development of systems of artificial intelligence". In: *Proceedings of the Computing Frontiers Conference*. ACM. 2017.

[9] *BONSEYES - Artificial Intelligence Marketplace*. https://www.bonseyes.eu/. URL: https://www.bonseyes.eu/ (visited on 03/21/2017).

[10] A. F. Westin. *Privacy and Freedom*. en. Bodley Head, 1970. ISBN: 978-0-370-01325-1.

[11] R. S. Poore. "Anonymity, Privacy, and Trust". In: *Information Systems Security* 8.3 (Sept. 1999). ISSN: 1065-898X. DOI: 10.1201/1086/43306.8.3.19990901/31071.4.

[12] Y. Cheng, M. Naslund, G. Selander, and E. Fogelstrm. "Privacy in machine-to-machine communications A state-of-the-art survey". In: *2012 IEEE International Conference on Communication Systems (ICCS)*. Nov. 2012. DOI: 10.1109/ICCS.2012.6406112.

[13] J. Daubert, A. Wiesmaier, and P. Kikiras. "A view on privacy and trust in IoT". In: *2015 IEEE International Conference on Communication Workshop (ICCW)*. June 2015. DOI: 10.1109/ICCW.2015.7247581.

[14] R. Shirey. *Internet security glossary, version 2*. Tech. rep. 2007.

[15] J. Huang and D. M. Nicol. "Trust mechanisms for cloud computing". en. In: *Journal of Cloud Computing: Advances, Systems and Applications* 2.1 (Dec. 2013). ISSN: 2192-113X. DOI: 10.1186/2192-113X-2-9. URL: https://link.springer.com/article/10.1186/2192-113X-2-9 (visited on 04/06/2017).

[16] A. R. Lombarte. *The Madrid Resolution*. International Standards on the Protection of Personal Data and Privacy. Nov. 2009. URL: https://icdppc.org/wp-content/uploads/2015/02/The-Madrid-Resolution.pdf/.

[17] *Article 29 Data Protection Working Party and Working Party on Police and Justice, The Future of Privacy*. Joint contribution to the Consultation of European Commission on the legal framework for the fundamental right to protection of personal data,WP168. Dec. 2009. URL: http://ec.europa.eu/justice/policies/privacy/docs/wpdocs/2009/wp168_en.pdf (visited on 03/21/2017).

[18] I. A. Ridhawi and M. Aloqaily. "A policy-based location-aware framework for personalized services in cloud computing systems". In: *2015 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*. Nov. 2015. DOI: 10.1109/AEECT.2015.7360583.

[19] C. Saravanakumar and C. Arun. "Location awareness of the cloud storage with trust management using common deployment model". In: *4th Int. Conf. on Computing, Communications and Networking Technologies (ICCCNT)*. July 2013. DOI: 10.1109/ICCCNT.2013.6726703.

[20] H. Gjermundrød, I. Dionysiou, and K. Costa. "privacyTracker: A Privacy-by-Design GDPR-Compliant Framework with Verifiable Data Traceability Controls". In: *Current Trends in Web Engineering: ICWE 2016 International Workshops, DUI, TELERISE, SoWeMine, and Liquid Web, Lugano, Switzerland, June 6-9, 2016. Revised Selected Papers*. Ed. by S. Casteleyn, P. Dolog, and C. Pautasso. Cham: Springer International Publishing, 2016. ISBN: 978-3-319-46963-8. DOI: 10.1007/978-3-319-46963-8_1.

[21] S. L. Kinney. *Trusted platform module basics: using TPM in embedded systems*. Newnes, 2006.

[22] A. Cooper and A. Martin. "Towards an Open, Trusted Digital Rights Management Platform". In: *Proceedings of the ACM Workshop on Digital Rights Management*. DRM '06. Alexandria, Virginia, USA: ACM, 2006. ISBN: 1-59593-555-X. DOI: 10.1145/1179509.1179525.

8

Privacy and DRM Requirements for Collaborative Development of AI Application

Vida Ahmadi Mehri, Dragos Ilie, Kurt Tutschku

Abstract: The use of data is essential for the capabilities of Data-driven Artificial intelligence (AI), Deep Learning and Big Data analysis techniques. This data usage, however, raises intrinsically the concerns on data privacy. In addition, supporting collaborative development of AI applications across organisations has become a major need in AI system design. Digital Rights Management (DRM) is required to protect intellectual property in such collaboration. As a consequence of DRM, privacy threats and privacy-enforcing mechanisms will interact with each other.

This paper describes the privacy and DRM requirements in collaborative AI system design using AI pipelines. It describes the relationships between DRM and privacy and outlines the threats against these non-functional features. Finally, the paper provides first security architecture to protect against the threats on DRM and privacy in collaborative AI design using AI pipelines.

8.1 Introduction

Collaborative application development across organisations has become a major focus in Data-driven Artificial Intelligence (AI) system design when aiming at sophisticated AI applications[1, 2]. This collaboration process builds on specialisation in AI engineering and on re-useable AI objects, e.g. data set or Deep Learning models. These objects have been gathered or developed by third-parties not designing the final application. The advantages of the process are potentially significant reductions of development cost and time and access to components that enable engineering for higher

AI performance. The appealing features are evidenced by the development of AI pipelines [3], open source machine learning and data visualisation tools such as Orange [4] and the emerge of data marketplaces [5, 6].

This collaborative approach, however, comes at a cost. It imposes at least three fundamental challenges on the design process. First, the use of data intrinsically raises data privacy concerns. These doubts become even deeper regarding the feature of datasets being shared. Second, Data-driven AI aims at identifying unknown relationships within the information. However, when using typical privacy enforcing mechanisms such anonymisation techniques or restriction in data collection, then it can't be excluded that inherent relationships within the data sets are not captured or deleted. As a result, such data sets are becoming of no value. While such typical privacy concepts are of high value for specific applications, they might impact the usability of AI objects in general. Hence, a dilemma for the general concept of collaboration based on reusability arises. Third, the reuse of AI objects in collaborative design requires trust among the developers and users of these objects. This trust ranges from obeying licences between developers to permitting governance on AI objects as required by societies and individuals, e.g. enabling GDPR or GDPR-like concepts on the use of data and AI objects in Europe. This paper aims to address these fundamental challenges by giving the insight to the privacy requirements in collaborative AI development. It will provide an initial taxonomy of privacy and Digital Rights Management (DRM) and the threats against objects in the AI pipeline. The paper summarises the GDPR act and its potential implications for Bonseyes-like AI marketplaces and describes potential ways of violating the DRM associated with the artefacts. It finally outlines a security architecture to circumvent the threats.

8.2 Collaborative AI Application Development

The purpose of data-driven AI is to analyse collected data in a smart way and come up with useful predictions about future data or provide new insights into existing data. However, in order to achieve good results, it is necessary to carefully prepare the data (*e. g.*, remove noise) and trim the algorithm parameters. The typical workflow model for data-driven AI, shown in Fig. 9.2 consists of five phases [7]:

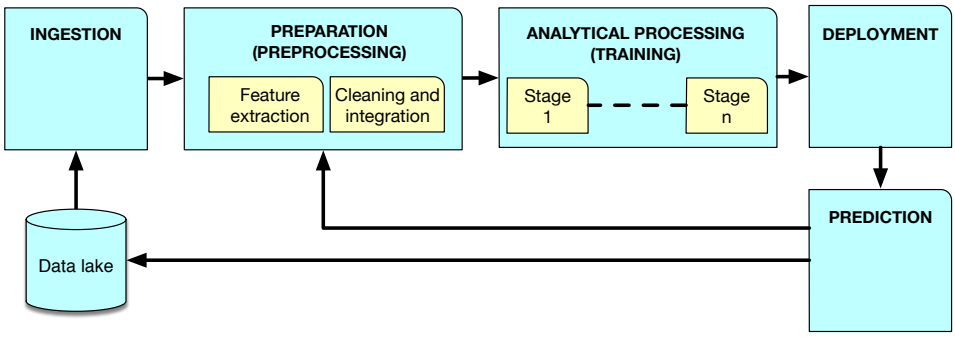

Figure 8.1: AI workflow model

i) ingestion,

ii) preparation (preprocessing)

iii) training (analytical processing),

iv) deployment

v) prediction

In the *ingestion* phase data is made available to the AI system either in the form of an existing dataset (*e. g.*, extracted from a data lake) or from live data (*e. g.*, collected from IoT devices).

The raw data must be converted to a format better suited for analysis. For example, data from heterogeneous sources that use different data type and syntaxes must be merged to a common format. This enables *feature extraction*, the retrieval of relevant attributes from data. Before data can be used for training it must be *cleansed* from errors resulting from the data collection process [8]. Additionally, scaling and normalisation may be required to enable uniform handling of features expressed with different scales of reference. Similarly, data reduction and transformation can be used to reduce the size of the input data for the training phase. All these operations are performed in the *preparation phase*.

The *training phase* is where the actual analytical processing of the data takes place. The output from this phase can be either new insights about the analysed data (*e. g.*, the correlation between specific data items) or a model used for predictions.

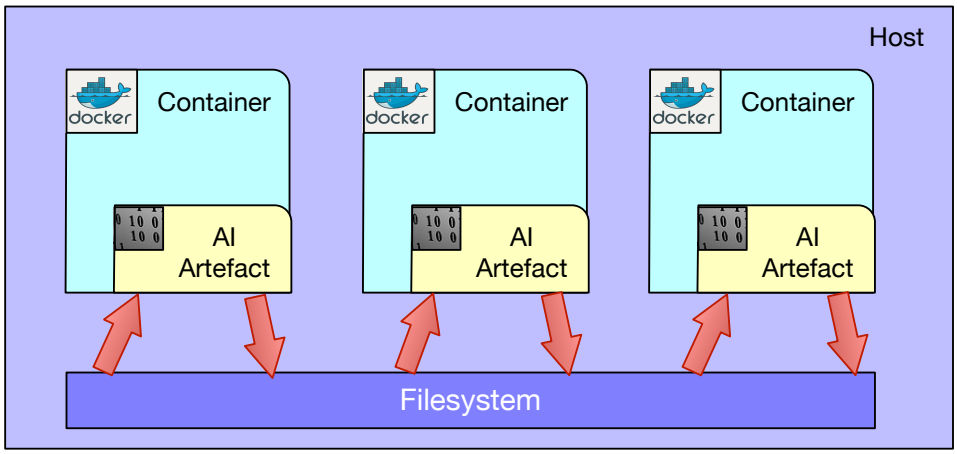

Figure 8.2: Original Bonseyes pipeline

A model needs to be *deployed* at a location where it can process input data. The location can vary between a powerful cloud computing environment to resource-constrained IoT devices, depending on the intended application.

The output of the models are predictions that can serve as input to business processes or decision-making systems. In some cases, a feedback loop is used, where the predictions are merged back into the original data lake or preparation phase to refine the existing model or train new models.

From a business perspective, we are witnessing the emergence of stakeholders that can provide access to high-quality data or algorithms. The co-dependency between data and algorithms in the AI workflow model suggests that collaboration between various stakeholders is required for developing complex, high-performant AI applications. To this end, the Bonseyes project is designing an AI marketplace that will enable such collaboration while maintaining privacy and enforcing DRM.

Bonseyes uses an Agile methodology for developing the marketplace. The current implementation of the AI workflow model is shown in Fig. 8.2 and is referred to as an *AI pipeline*. The light-blue rectangles in the figure are Docker containers. Inside the Docker container resides an *AI artefact*. The artefact can be pure data, an interface to a data source or an algorithm used in the AI workflow.

The developers retrieve the required AI artefacts from a Docker repository

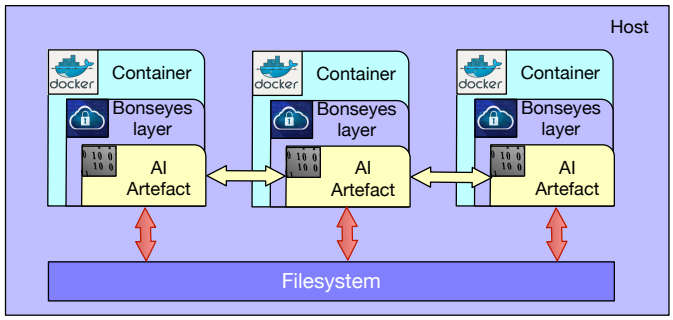

Figure 8.3: Improved Bonseyes pipeline

and assemble the pipeline on their local system. The red arrows represent data transfers to and from storage (currently a regular filesystem). The transfers are needed to bring data in the pipeline and to save the output from intermediate stages and the final result. The data may include confidential information, hence the red color.

Fig. 8.3 shows the next implementation of the AI pipeline, where the AI artefact is wrapped inside a *Bonseyes Layer (BL)*. The purpose of the *BL* is to enable secure direct artefact communication without the need to store intermediate data. It also provides DRM mechanisms to control access to artefacts and enforce license management policies. The need for DRM mechanisms is driven both by business motives as well as legal requirements.

The new implementation of the AI pipeline supports also a distributed model, as shown in Fig. 8.4, which allows various components of the pipeline to be executed on different hosts. This model requires that the *BL* is extended to the hosts in order to facilitate workflow distribution and is the model considered for the remainder of the paper.

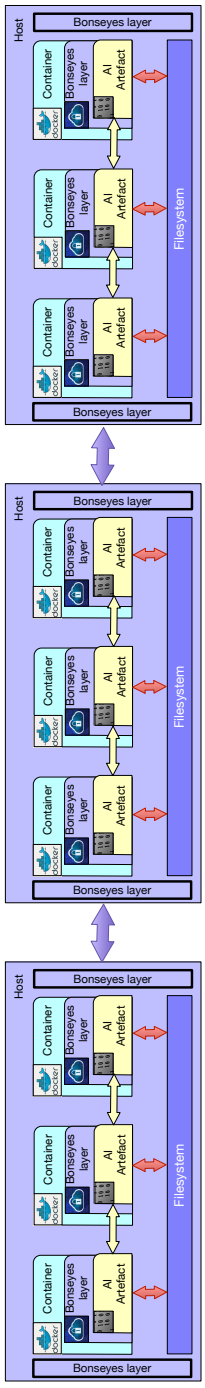

Figure 8.4: Distributed Bonseyes pipeline

8.3 DRM and Privacy Requirements

This section aims to highlight the privacy requirements stemming from personal data regulations and describes the high level relationship between privacy and DRM. The impact of the regulation on the AI pipeline and the challenges for collaborative AI development is presented in Section 8.3.4.

8.3.1 GDPR privacy requirements

General Data Protection Regulation (GDPR) is the uniform approach towards all EU countries to provide the protection of a natural person while processing that individual's personal data. It will come in force in May 2018 as a replacement of European data protection Directive (EU Directive 95/46/EC) to give the EU citizens better control over their personal information. GDPR expands the notion of personal data to photos and audios, financial transactions, posts on social medias, device identifiers, location data, users login credentials, and browsing history, as well as genetic information. The new regulation applies in a wide territorial scope and it includes all countries (EU or non-EU) that process the personal data of EU residents.

GDPR relies on six main principles for processing personal data namely a) lawfulness, fairness, and transparency; b) purpose limitation; c) data minimisation; d) accuracy; e) storage limitation; f) confidentiality and integrity [9].

GDPR limits the collection and storing of identification information of data subjects up to the minimum necessary required in order to safeguard data subjects' rights and freedom.

The new regulation expands the rights of data subjects and lists detailed obligations and responsibilities for two key entities: controllers and processors. The *controller*, who "determines the purposes and means of the processing of personal data", is responsible for implementing technical and organisational measurement to ensure the processing of data is performed as described in the regulation. The organisational measurement includes assigning a data protection officer, doing data protection impact assessment and risk mitigation plan. The technical measurement includes activities such as implementing accountability, pseudonymisation and data minimisation. A controller can delegate the processing of personal data to a *processor*,

who "processes personal data on controller's behalf". It is therefore the responsibility of the controller to select processors who guarantee the implementation of appropriate technical and organisational measures that meet the regulation requirements and ensure the protection of data subjects [9].

8.3.2 GDPR & DRM

Digital rights management enable content owners to control the distribution and usage of their content. DRM systems conduct policies to manage the access to the digital contents. In general, DRM schemes for content protection are built on the assumption that the content owner has exclusive rights under the copyright law for the usage and distribution of the owned content. Consumer access to the content is regulated by a, typically non-transferable, license that is enforced through the DRM mechanism. GDPR does not attempt to challenge copyright law, but there are scenarios when the two can come into conflict. For example, if the protected content contains personal data collected from data subjects, then that content is subject to requirements implied by the GDPR. Another example consists of DRM mechanisms that use personal data to track the activity of individual users. The tracking data becomes also subject to the requirements of GDPR.

Typical DRM schemes can only allow or deny access to the entire content and not to portions of it. Furthermore, these schemes do not support fine-grain access control over user activities after the access is granted. Therefore, typical DRM schemes do not address the GDPR requirements (*e. g.*, recording processing activities). However, if DRM technologies are complemented with auditing and accounting capabilities, they can potentially help the GDPR controller to implement a technical measurement.

8.3.3 Privacy & DRM

Privacy is a fundamental human right [9] that can be defined in different ways. S. Warren and L. Brandeis proposed privacy as a "right to be left alone" [10]. At the same time, philosophers define the privacy as a right to limit access to individuals' information about lives, thoughts, and bodies [11]. These definitions show the broad concepts of privacy that apply to processing individuals' information and observing human behaviours and relations. The philosophical aspects of privacy is handled through information privacy law and data protection policies that regulate the process, storage, usage, and

exchange of personal data. Some technologies, like DRM, seem to be aligned with privacy rights due to their basic natures. The main features of DRM, such as access control to protected content, restriction of unauthorised usage, and verification of the authenticity of identification data, are useful for implementing the legal requirements for privacy in a technical solution.

In [12], the authors proposed adaptations to DRM systems to address Privacy Rights Management (PRM). They define PRM as management of personal information according to the requirements of the EU Directive 95/46/EC [13]. To do that, the authors mapped the DRM entities to entities defined by the Directive. For instance, they mapped content owner to data subject, content distributor to controller, and user to processor. Unfortunately, their conclusion is that their method is not scalable in terms of issuing processing reports when the number of data subjects is very large. This remains a challenge in building DRM systems that support PRM-like features.

8.3.4 GDPR impact on AI pipelines

From a business point of view, the AI marketplace enables *artefact consumers* to obtain *licensed* access to AI artefacts owned by *artefact providers*. An entity can play both roles: it can be a provider for artefacts it owns, and it can consume artefacts from others. An artefact is coupled together with a license that specifies the terms of use for the artefact. Since the license is viewed as a legal document it must be anchored into existing laws and regulations (*e. g.*, GDPR) in order to be effective. A license can be encoded into a digital license to enable computer-based license management systems to monitor, detect and prevent license violations.

As described in Section 8.2, AI pipelines are setup in order to achieve specific goals such as to make predictions or obtain new insights from existing data. Since the purpose of the pipeline is determined by the entity setting up, it is reasonable to conclude that the entity is acting as a controller in the GDPR sense and thus is responsible for complying with the GDPR. When viewed in the context of an AI marketplace, individual AI artefacts in the pipeline act as generic building blocks and should not be aware of the pipeline's higher purpose. It is thus reasonable to assume that artefact providers act as GDPR processors. However, some providers may implement the functionality of an artefact by chaining together several other artefacts,

where each element of the chain consume the output of the previous element and provides input to the next. The determination of whether the provider of a chained artefact acts as controller or processor does not seem so clear cut in these cases.

The distinction between controllers and processors becomes even blurrier when the AI artefact contains a data source or pure data. The artefact provider could have collected and processed private data from its data subjects for specific purposes, thus being itself a controller according to GDPR. The provider may view the pipeline owner as a processor or they may establish a joint controller relationship [9].

This discussion highlights some of the immediate difficulties in attributing legal responsibilities to providers and consumers. It is not quite clear at the moment how an AI marketplace could *automatically* assign correctly (in a legal sense) the controller/processor roles to various entities participating in a pipeline. We believe that a tractable initial approach is to determine a baseline of requirements (legal and technical) for processors. All consumers and providers in the marketplace would be required to fulfill these constrains. This would enable controllers to delegate data processing to any processor.

8.4 DRM and Privacy Requirements for AI Development

In general, DRM generates the license for each authorised entity in an AI pipeline. This section identifies general artefact usage constraints that can be embedded in a license:

i) **Validity** is defined by the allowed access duration to the artefact and is referred to as the license *validity period*. Additionally, the validity may also be constrained by the number of times the artefact can be executed, a so-called n-times use price model. Also, a license should be revokable if users breach the license agreement or to allow users to interrupt an automatic subscription renewal.

ii) **Beneficiary** constraint specifies the identity of the *license user*. An end-user license has a single user as beneficiary, whereas a group license allows an organisation to allocate and revoke licenses to its members according to own policies.

iii) **Purpose** constraint defines the *license scope* (*e. g. ,* commercial, educational, or personal). This can even be tied to specific types of licenses (*e. g. ,* GPL) that govern how derivative work (*e. g. ,* AI applications) may itself be licensed. In addition, legislation, such as the GDPR, may stipulate how data must be collected, stored, shared and processed. The usage purpose constraint must be able to incorporate this type of law-derived policies.

iv) **Location** constraints define where the artefact can be utilised. This constraint can be either topological or geographic. The topological location defines the networked hosts that are allowed to use the artefact. For the Bonseyes project, the simplest topological constraint restrict artefact usage to the set of Bonseyes authorised hosts. However, artefacts can also be constrained to *virtual premises*, which are smaller subsets of Bonseyes authorised hosts[14]. The geographical location defines where in the world the artefact can be used or is prohibited from being used. This allows hosts and artefact to comply with local laws and regulations, such as EU Data Protection Directive and its successor, GDPR.

v) **Peering** constraint regulates which entities the artefact is allowed to interact with. The peers can be the successor or predecessor artefact in a pipeline or entities that monitor and control the operation of the artefact.

A license management system must protect the interests of both artefact consumers and providers, according to license constraints. This means that consumers are not prevented from using the artefact while the license is valid, but also that they cannot continue to use it if the license expires, is revoked for a legitimate reason, or if the artefact is used for a purpose or at a location prohibited by the license. Similarly, the providers must be prevented from blocking access to an artefact for consumers with a valid license, while at the same time retain the possibility to revoke a license when its terms are breached. The collaborative nature of the AI marketplace raises some interesting challenges in satisfying these requirements simultaneously.

To ease the threat analysis, we will consider two classes of misbehaving users: regular users and malicious users. Regular users can be consumers that try using the artefact in conflict to the license constraints, or providers that attempt to prevent consumers from using an artefact although a valid

license exists. Common for this case is that the users are either unaware that they are breaching the license agreement, or that they are aware, but do not employ any advanced means (*e. g.*, reverse engineering or code injection) to achieve their goals. Therefore, we consider these potential attacks as *simple threats*. On the other hand, the malicious users are assumed to be skilled attackers that may insert exploits into the AI pipeline or instrument the artefact's hosts in order to bypass the license and obtain unfettered access to the artefact of interest. We consider these to be *advanced threats*. Provider attempts to fraudulently prevent consumer with valid licenses from using an artefact belong also to the advanced threats category.

8.4.1 Simple threats

We begin by considering two obvious threats:

T-1: A user can modify license contents to bypass usage constraints

T-2: The AI marketplace repudiates the issuance of a license

It is assumed that the license contents are stored inside the BL that encapsulates the AI artefact. An additional assumption is that the license contents as well as the AI artefact are digitally signed by the marketplace. The signature enables the BL to detected fraudulent changes to the license contents and asserts the origin of the data thus preventing the marketplace from repudiating an issued license. We assume the signature algorithm itself (*e. g.*, RCA-PSS, DSA or ECDSA) when used with a reasonable key length is infeasible to break. This counteracts threat T-1 and T-2.

A *license validity period* begins on a specific start date and stops at an expiration date. The expiration date can be left undefined by the licensor if perpetual validity is desired. Additionally, if an n-times use pricing model is used, the validity can be further restricted by the number of times the artefact is executed. A license can be revoked before the expiration date for reasons mentioned previously in this paper. To determine if a license is valid, the BL is dependent on having access to a time source, such as a Network Time Protocol (NTP) server, and to a license revocation database. We consider threats against the integrity or availability of the time source and the license revocation database. More specifically, we require security mechanisms are put in place to protect against:

T-3: Blocking communication with the time source

T-4: Blocking communication with the license revocation server

T-5: Spoofing a time source

T-6: Rolling back the time on the artefact host to use the artefact past expiration date

Threat T-3 and T-4 are essentially DoS attacks on network services required by the license management system. To counteract them, the BL should refuse to execute the artefact unless communication with the servers is possible. The downside of this approach is that AI pipelines become unavailable during periods of none-malicious network outage (*e. g.*, on mobile units performing handoffs). This situation can be improved by introducing *grace time*, which is an acceptable duration for communication outage. Threat T-5 can be handled by using NTP Autokey [15] or a time server authentication mechanism similar to it. To defend against threat T-6, we assume that after an initial connection to the time source, the artefact is able to detect deviations from monotonically increasing time.

The *license beneficiary* uniquely identifies the licensed consumer. This information is determined at the time the license is acquired. Its presence in the license enables accounting and usage tracking. Since the license and AI artefact are protected by a digital signature, it is assumed that no simple threats exists against the integrity of this information. However, it is important to protect against the execution of an artefact by an unlicensed user (*e. g.*, in the case a pirate copy is made).

T-7: Artefact execution by unlicensed user

To counteract T-7, it is necessary to require users to authenticate themselves to the BL with a digital user certificate signed by a Bonseyes certification authority (CA). First, the BL would have to establish a chain of trust to the root CA and then verify the validity of the certificate by consulting certificate revocation lists (CRLs) or through invoking an Online Certificate Status Protocol (OCSP) responder. If the certificate is valid, the BL can verify if the license beneficiary matches the distinguished name (DN) in the user certificate. Finally, the BL would present a challenge to the user agent

to ascertain the user in possession of the private key associated with the certificate. It is important to realise that this defence will not work against attackers that are able to obtain the licensed artefact, the signed certificate and the private key.

License purpose is metadata encoding the permitted use scope for the artefact. Hosts in a virtual premise have similar metadata configured for their BL. The artefact BL and the host BL each expose their use scope. Thus, the artefact can decide if a specific host meets its purpose constraint and the host can decide if it can allow artefact execution. For example, an educational virtual premise may refuse to execute commercial artefacts, or an artefact can avoid running on hosts that does provide adequate privacy protection (*e. g.*, according to GDPR). This constraint faces the following threats:

T-8: Host breach against license purpose

T-9: Artefact breach against license purpose

In the case of T-8, the host is configured to present a false use scope that entices the artefact to execute its payload. This threat can be addressed by requiring that before an organisation can deploy a Bonsyes host it must obtain a digitally signed license for that host from the Bonseyes CA. The license carries the approved use scope and is tied to the host's hardware through a hardware ID [16]. The BL in the host can recompute the hardware ID and verify that it matches the one covered by the license, before presenting the host license to the artefact BL. Threat T-9 considers the situation where the artefact is attempting to illegitimately obtain execution privilege on the host. In this case, the host BL must verify the artefact license to ensure its use case is allowed.

We foresee the following topological location threats:

T-10: Artefact execution outside a virtual premise (*i. e.*, host lacking a valid BL)

T-11: Artefact execution on wrong virtual premise

These threats are similar to the T-8, but they are more generic in scope. Virtual premises are host pools for AI pipelines with different constraints.

The license purpose is an example of such a constraint. Performance can be another constraint. For example, one may define a development premise as a pool of workstations used by data scientists for research and development work. At the same time, a production premise can consist of powerful servers used for high-performance AI pipelines. Threat T-10 describes the trivial case where a host lacks the BL or contains an invalid BL, thus being outside any virtual premise. If the host BL is missing, the artefact BL cannot establish a connection to it and must abort execution immediately. An invalid BL is either old deprecated code (detectable through a revoked license) or contains an invalid license (*e. g.*, expired or missing a valid trust chain). Also in this case the artefact BL must abort execution.

Threat T-11 considers the situation where an artefact is restricted to a specific virtual premise and by mistake or intentionally is executed on a different premise. A scenario illustrating this threat is that of an artefact under development (*e. g.*, using an untested algorithm or containing low-quality data) being mistakenly deployed on a production premise, thus degrading the quality of the results. To counteract this threat we propose that host and artefact licenses contain a set of virtual premise identifiers that must match in order to allow artefact execution.

T-12: Artefact execution outside allowable geographic region.

Threat T-12 is a special version of T-11. In this case, we consider mobile virtual premises, a host or a set of hosts that can move or be transported geographically. The issue we try to capture is that when geographic boundaries are being crossed (like country or state borders) different jurisdiction may apply to computation and data. For example, some countries may have laws that severely curtail data privacy and where certain types of computation (*e. g.*, cryptography) can be considered a criminal offence. Theoretically, this threat can be dealt with in a similar manner as with T-10 and T-11, by encoding allowed geographic regions in the license. Unfortunately, enforcing this type of constraint is quite difficult. The reasons are firstly, that there is no single method that can do accurate Internet geolocation without assistance from GPS or cellular networks and, secondly, evasion methods through proxies and anonymization networks are quite successfull [17]. Therefore, more research is required to determine efficient countermeasures for this threat.

T-13: Unauthorized peering.

In an AI pipeline there are multiple artefacts chained together. Threat T-13 is about scenarios where an artefact in a chain must determine if the predecessor and the successor artefact are authentic and part of the same virtual premise. It applies also to the communication between control entities and artefacts. This threat can be addressed by having peering entities exchange licenses with each other. The neighbour licenses are verified and each peer is required to demonstrate that it has possession of the corresponding private key.

8.4.2 Advanced threats

In this section we consider threats from skilled malicious users. The assumption is that this type of users are able to modify the code for both the artefact BL as well as the host BM. Also, they are legitimate AI marketplace users and can, for example, obtain valid host licenses. Thus, these attackers have the capability to circumvent any authorization and authentication mechanisms running on systems that are under their control. Consequently, the defence mechanism for T-1 to T-13 are defeated on all virtual premises under the control of the attacker. More specifically, it means that license management mechanisms can be bypassed and AI artefacts obtained from the marketplace cannot impose any limitation on how they are used. The attacker can also extract the actual algorithms or data encapsulated by the artefact and make it available outside the marketplace. Even if artefacts employ encryption as a copy-protection scheme, they cannot be protected against this type of attacks. The crux of the problem is that for an artefact to be useful, it needs to be decrypted at some point. The attackers can instrument the hosts under their control using a tool such as Intel PIN[1] and locate the decrypted artefact payload. This type of approach was successfully used to defeat movie DRM [18].

Although the outlook for defending against advanced threats looks quite gloom, there are two complementary approaches that can raise the difficulty of mounting a successful attack. The first approach aims at providing a trusted computing environment for license management mechanisms to execute, in particular one in which the integrity of host and artefact BL are

[1] `https://software.intel.com/en-us/articles/pin-a-dynamic-binary-instrumentation-tool`

protected. The ideas about trusted computing have been vastly explored in the past and have lead to the design of the Trusted Platform Module (TPM), which is an international standard for a secure crypto-processor. A discussion about TPM is outside the scope of this paper and the interested reader is referred to [19] and [20]. The TPM is a passive chip located on the computer mainboard and can perform measurements (*i. e.,* compute cryptographic hash values) on software and its dependencies, before the software is executed. In addition, it enables a task called *remote attestation* where the measurements of a host are communicated over a secure cryptographic channel to a remote attestation server. The attestation server can compare the measurements with a set of pristine values collected in advance and, if they match, authorize the host to engage in activities with the rest of the system.

The TPM has been plagued from the start by a number of deficiencies and major manufacturers such as Intel and AMD have worked over the years to address them. Intel has recently released the Software Guard Extensions (SGX) for their CPUs, bringing support for secure *enclaves*. Applications and data running inside enclaves are protected from other software running on the same platform. Most recently, this mechanism has been used to protect Docker containers[21]. The downside of TPM- or SGX-based solutions is that the actual hardware module (or processor extension in case of SGX) is not available on all platforms. It should be noted that although SGX, if correctly used, can provide considerable protection against tampering with the computing environment, there are already attacks against it that are considered practical [22].

Our second proposed approach for providing a trusted environment is to remove the hosts from the control of a potentially malicious users. In particular, we envision a scheme, similar to [23], where the actual implementation of the AI pipeline is done by a third party, such as a cloud provider. The assumption is that the cloud provider's main business interest is in selling computation and storage and thus the provider has no incentive to engage in malicious activities agains the AI marketplace and its users. Artefact consumers interface with the artefact through well-defined interfaces, without having direct access to the artefacts or the hosts executing artefacts. Although, this approach can provide a high level of security against malicious users, it requires a major readjustment to current AI development practices and is thus regarded as a major inconvenience by the developers.

8.5 AI Marketplace Security Architecture

We propose three architecture models based on the requirements described in the sections 8.3 and 8.4 . The single and multi-host architectures are suitable to address the simple threats and the 3rd party cloud architecture aims at mitigating the advanced threats. The main entities in our protection schemes are as follows:

i) **MarketPlace (MP)**: it is the interaction point between all parties. The MP enables collaboration between the main entities listed below.

ii) **Artefact Provider (AP)**: makes the AI artefacts available by registering them in the MP. AP is the owner of the artefact having rights to describe the artefacts' access policy, license type, and privacy restrictions.

iii) **Security Manager (SM)**: is the heart of the license management system. It generates the access policy per artefact dynamically based on the AP description. SM has the multiple functionalities such as the DRM enforcement system, remote attestation, AAA, and certificate management. It is responsible for initialising the artefact per request. It manages the system log, license, privacy policies and the execution of each artefact.

iv) **Data Scientist (DS)**: is the entity that needs the AI artefact for testing or developing new applications. It is regarded as the artefact consumer.

v) **Computation Center (CC)**: is the datacenter which provides the infrastructure for executing the AI artefact. It can be located on the DS premise or with a 3rd party cloud provider and is required to establish trust to SM *e. g.* , through trusted computing. Therefore, the CC hosts should be equipped with Trusted Platform Modules (TPMs) or equivalent.

vi) **Virtual Premise (VP)**: spans the computational capabilities of the hosts involved and provides the trusted execution environment to protect the AI artefact. It prohibits the communication between the artefact and outside world. The MP and SM are part of every VP.

We assume the SM is under the control of the MP and helps the secure collaboration between parties. The payment transaction is beyond the scope of this paper. The BL contains the privacy rights and access policies and it is customised by the SM for each artefact instance and host.

8.5.1 Artefact retrieval and license enforcement

All entities must be authenticated to the MP before requesting access to the AI artefact. After successful payment procedure, the SM generates the license for the user based on the service term agreements between the DS, AP, and the MP. The SM provides customised privacy rights and inserts their description inside the BL. The DS could download his own copy of the artefact to execute it in his own premise if the requirements are met. The BL establishes a secure connection to the SM to check the validity of the license in order to permit artefact execution inside the DS premise. Blocking the communication channel to the SM is considered a policy violation and the BL will halt in that case the execution of the artefact. If the communication channel is unblocked and the artefact and the DS have valid licenses, the artefact will begin execution and a log entry will be submitted to the SM. If a policy violation occurs during runtime, the BL will log to the SM and exit artefact from execution mode.

The DS does not have direct access to the artefact, but instead must communicate with artefact through predefined interfaces denoted by blue arrows in Fig. 8.5- 8.7. The interfaces are typically managed through an integrated development environment (IDE) such as Eclipse, denoted by DS Workbench (staging area) in the figures.

The general activities to obtain an artefact from the MP are the same for different architectures and the exact artefact retrieval protocol is under development.

8.5.2 Single Host Architecture

This model is adapted to current AI development practices, where the DS has access to the entire pipeline in his own workstation. It provides minimum security compared to the other two architectures and it is targeted towards a convenient work mode for developers. In addition, it provides convenient artefact mobility. The artefact could move either together with the host (*e. g.* , laptop) where it is installed, or the artefact itself can be transferred

to other BL-enabled host (*e. g.*, being carried on a USB memory stick). This architecture prefers simplicity over security and thus does not offer any protection against advanced threats.

The architecture depicted in Fig. 8.5 shows the pipeline in a single authorised host. The yellow arrows represent the communication through BL interfaces between peering artefacts, whereas the blue arrows denote the communication between the interfaces in the DS workbench and the actual artefacts. The green arrow represents the communication channel between BL entities and the SM that was described in Section 8.5.1. The DS follows the general approach in Section 8.5.1 to obtain the artefacts from the MP, assemble and execute the pipeline in the hosting workstation. The DS can move artefacts to a different topological and/or geographical location if this is not prohibited by the artefact policies.

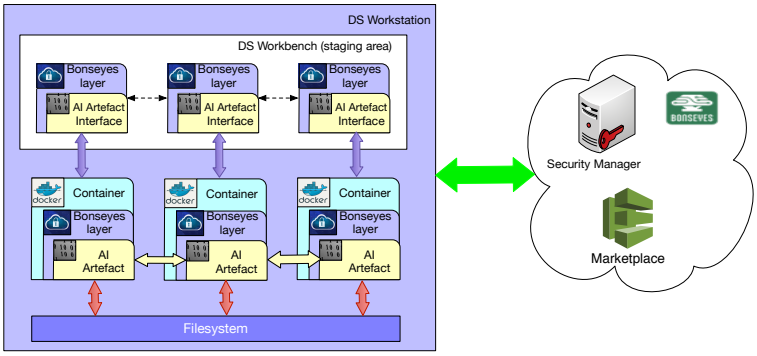

Figure 8.5: Isolated pipeline host

8.5.3 Multi-Host Architecture

The multi-host architecture depicted in Fig. 8.6 addresses the situation when the computing requirements of the pipeline exceed those available on a single host. In this model, the DS can extend its pipeline to multiple hosts. The containers shown in grey colour represent artefacts belonging to a different pipeline. The green arrow is a communication channel to the SM. Each host belonging to a pipeline must establish its own secure channel to the SM. A SM must issue licenses for multiple authorised hosts under the control of the DS. This architecture is designed to improve the performance of the single host architecture by facilitating the implementation of the pipeline across several hosts.

The general approach described in Section 8.5.1 is used to receive the artefact. However, the hosts are under the control of the DS and thus, when the DS is malicious, this architecture is still vulnerable to the advanced threats described in Section 8.4.2.

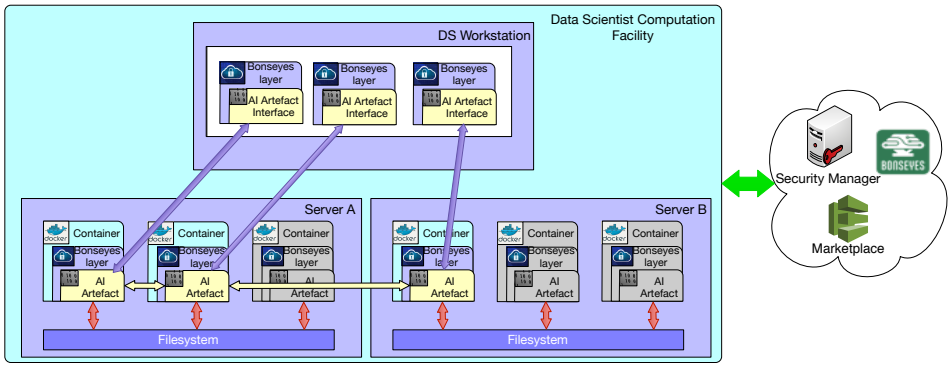

Figure 8.6: AI pipeline execution at a CC hosted by the DS

8.5.4 3rd Party Cloud Architecture

Fig. 8.7 shows the 3rd party cloud architecture aiming to address the advanced threats. Following the approach from Section 8.5.1, the DS authenticates to the MP and requests for the AI artefact to be executed in a 3rd party CC. The 3rd party CC must be trusted by the MP and be authenticated against the SM. In addition, the provider must install in advance the BL on hosts available for BL pipelines. The CC must be out of the control of the DS and moreover have no interest in the artefacts. More general, we assume there is no collusion between the CC and the DS. The procedure from Section 8.5.1 will follow and the provisioned artefacts will instantiate for execution in the 3rd party CC. The BL in the artefact must be able to communicate with the SM through secure channel to enable license management. The BL permits the artefact execution if the communication channel is alive and the license requirements are matched.

The green arrows in Fig. 8.7 represent the communication channel between the SM and all other participating entities. To prevent compromised artefacts belonging to malicious DSs from exfiltrating entire datasets, the total volume of data retrieved as output of the pipeline is restricted . The

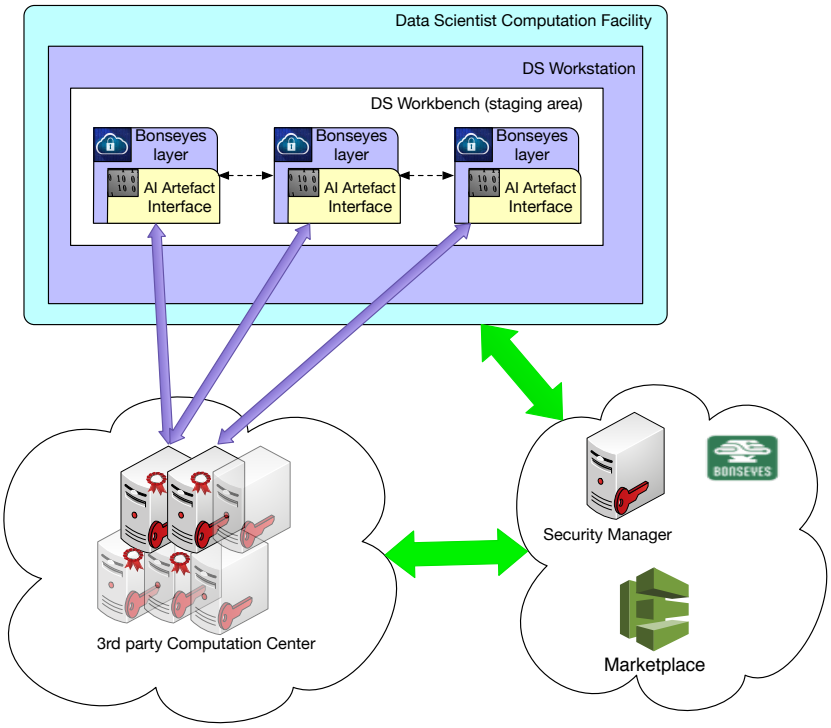

Figure 8.7: AI pipeline execution at a CC hosted by the 3rd party cloud provider

limitation is both in terms of request rates for pipeline execution as well as on the size of the output for individual requests. We believe this architecture is more secure against the advanced threats because artefact consumers are forced to use dedicated interfaces instead of having direct access to raw artefacts or the executing hosts. This measure raises the difficulty to circumvent DRM mechanisms by compromising pipelines with instrumented artefacts and hosts.

8.6 Conclusion

The paper describes the privacy requirements for enabling collaborative concepts in AI application development. It investigates the ordinary DRM constraints to address privacy requirements by regulations and proposes a DRM scheme to achieve fine-grain access to the artefacts. This work describes

the possible threats in collaborative AI and suggests three architecture models to meet the requirements. The paper aims at a practical solution for the privacy challenge in AI application development and at facilitating collaboration between different stakeholders.
In the future work, we will implement the proposed architecture models and investigate the vulnerability of the models based on the identified threats.

Acknowledgment: This project has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 732204 (Bonseyes). This work is supported by the Swiss State Secretariat for Education Research and Innovation (SERI) under contract number 16.0159. The opinions expressed and arguments employed herein do not necessarily reflect the official views of these funding bodies.

References

[1] I. Stoica, D. Song, R. A. Popa, D. Patterson, M. W. Mahoney, R. Katz, A. D. Joseph, M. Jordan, J. M. Hellerstein, J. E. Gonzalez, et al. "A Berkeley View of Systems Challenges for AI". In: *arXiv preprint arXiv:1712.05855* (2017).

[2] T. Llewellynn, M. Fernández-Carrobles, O. Deniz, S. Fricker, A. Storkey, N. Pazos, G. Velikic, K. Leufgen, R. Dahyot, S. Koller, et al. "BONSEYES: platform for open development of systems of artificial intelligence". In: *Proceedings of the Computing Frontiers Conference*. ACM. 2017.

[3] *PipelineAI - Home*. available at `https://pipeline.ai/`. (Visited on 04/06/2018).

[4] *Orange Data Mining Fruitful & Fun*. de-DE. available at `https://orange.biolab.si/`. (Visited on 04/06/2018).

[5] *Data Market Austria*. available at `https://datamarket.at/`. (Visited on 04/06/2018).

[6] *Industrial Data Space e.V.* de-DE. available at `http://www.industrialdataspace.org/`. (Visited on 04/06/2018).

[7] S. Yegulalp. "Data in, intelligence out: Machine learning pipelines demystified". In: InfoWorld, May 2017.

[8] C. C. Aggarwal. *Data Mining: The Textbook*. ISBN: 973-3-319-38116-9. Springer International Publishing Switzerland, 2015. ISBN: 973-3-319-38116-9.

[9] European Parliament and Council of the European Union. "Regulation (EU) 2016/679 of the European Parliament and of the council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)". In: *Official Journal of the European Union*. Vol. 59. L 119. ISSN: 1977-0677. May 2016.

[10] S. Warren and L. Brandeis. "The Right to Privacy". In: *Harvard Law Review* 4.5 (Dec. 1890).

[11] F. D. Schoeman, ed. *Philosophical Dimensions of Privacy: An Anthology*. English. 1st. Cambridge Cambridgeshire ; New York: Cambridge University Press, Nov. 1984. ISBN: 978-0-521-27554-5.

[12] S. Kenny and L. Korba. "Applying digital rights management systems to privacy rights management". In: *Computers & Security* 21.7 (Nov. 2002). ISSN: 0167-4048. DOI: 10.1016/S0167-4048(02)01117-3.

[13] *Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data*. en. Nov. 1995. URL: http://data.europa.eu/eli/dir/1995/46/oj/eng (visited on 05/03/2018).

[14] V. Mehri and K. Tutschku. "Flexible Privacy and High Trust in the Next Generation Internet: The Use Case of a Cloud-based Marketplace for AI". In: *Proceedings of 13th Swedish National Computer Networking Workshop (SNCNW)*. Halmstad University, Sweden, 2017.

[15] B. Haberman and D. L. a. Mills. *RFC 5906: Network Time Protocol Version 4: Autokey Specification*. IETF. June 2010. URL: https://tools.ietf.org/html/rfc5906.

[16] Microsoft. *Specifying Hardware IDs for a Computer*. Available at https://docs.microsoft.com/en-us/windows-hardware/drivers/install/specifying-hardware-ids-for-a-computer. 2017.

[17] J. A. Muir and P. C. Van Oorschot. "Internet Geolocation: Evasion and Counterevasion". In: *ACM Computing Surveys* 42.1 (Dec. 2009).

[18] R. Wang, Y. Shoshitaishvili, C. Kruegel, and G. Vigna. “Steal This Movie: Automatically Bypassing DRM Protection in Streaming Media Services”. In: *Proceedings of 22nd USENIX Security Symposium*. Washington DC, USA, 2013. URL: %5Curl%7Bhttps://www.usenix.org/conference/usenixsecurity13/technical-sessions/paper/%7D (visited on 03/20/2018).

[19] K. Hall, T. Lendacky, E. Ratliff, and K. Yoder. “Trusted Computing and Linux”. In: *Proceedings of the Linux Symposium*. Ottawa, Ontario, Canada, 2005.

[20] B. Parno, J. M. McCune, and A. Perrig. “Bootstrapping Trust in Commodity Computers”. In: *Proceedings of IEEE Symposium on Security and Privacy*. Oakland, CA, USA, 2010.

[21] S. Arnautov, B. Trach, F. Gregor, T. Knauth, A. Martin, C. Priebe, J. Lind, D. Muthukumaran, D. O’Keeffe, M. Stillwell, G. D., D. Eyers, R. Kapitza, P. Pietzuch, and C. Fetzer. “SCONE: Secure Linux Containers with Intel SGX”. In: *Proceedings of USENIX OSDI*. Savannah, GA, USA, 2016.

[22] F. Brasser, U. Müller, A. Dmitrienko, K. Kostiainen, S. Capkun, and A.-R. Sadeghi. “Software Grand Exposure: SGX Cache Attacks Are Practical”. In: *arXiv:1702.07521 [cs]* (Feb. 2017). arXiv: 1702.07521. URL: http://arxiv.org/abs/1702.07521 (visited on 03/20/2018).

[23] R. Petrlic. “Privacy-Preserving Digital Rights Management in a Trusted Cloud Environment”. In: *Proceedings of TrustCom*. Liverpool, England, UK, 2012.

9

Designing a Secure IoT System Architecture from a Virtual Premise for a Collaborative AI Lab

Vida Ahmadi Mehri, Dragos Ilie, Kurt Tutschku

Abstract

IoT systems are increasingly composed out of flexible, programmable, virtualised, and arbitrarily chained IoT elements and services using portable code. Moreover, they might be sliced, i.e. allowing multiple logical IoT systems (network + application) to run on top of a shared physical network and compute infrastructure. However, implementing and designing particularly security mechanisms for such IoT systems is challenging since a) promising technologies are still maturing, and b) the relationships among the many requirements, technologies and components are difficult to model a-priori.

The aim of the paper is to define design cues for the security architecture and mechanisms of future, virtualised, arbitrarily chained, and eventually sliced IoT systems. Hereby, our focus is laid on the authorisation and authentication of user and host, as well as on code integrity in these virtualised systems. The design cues are derived from the design and implementation of a secure virtual environment for distributed and collaborative AI system engineering using so called AI pipelines. The pipelines apply chained virtual elements and services and facilitate the slicing of the system. The virtual environment is denoted for short as the *virtual premise (VP)*. The use-case of the *VP* for AI design provides insight into the complex interactions in the architecture, leading us to believe that the *VP* concept can be generalised to the IoT systems mentioned above. In addition, the use-case permits to derive, implement, and test solutions. This paper describes the flexible architecture of the *VP* and the design and implementation of access and execution control in virtual and containerised environments.

9.1 Introduction

The Internet of Things (IoT) is evolving. Originally, it aimed at wirelessly interconnecting cyber-physical devices (sensors, actuators, wearables, vehicles, home appliances) with control applications [1]. However, complex enhancements of the IoT have been recently developed. IoT is increasingly viewed as a service- and Cloud-based system [2]. Such systems leverage, for example, the intermittent nature of wireless connectivity or provide for stateless information access using RESTful services. The concepts of Fog computing [3] and Multi-access Edge Computing (MEC) in public cellular networks[4] also complement the original IoT idea. Here, edge devices are located close to sensors and perform substantial amounts of computation, storage, communication (*e. g.,* high bandwidth or multi-homing for reliability). Furthermore, slicing concepts for IoT systems have been proposed [5]. They typically focus only on slicing techniques for the network segment and use SDN (Software Defined Networking) concepts to control the data forwarding.

Our hypothesis is that IoT systems accelerate even stronger towards flexible, virtualised, programmable, sliceable, and arbitrarily chained IoT elements and services using portable code. We substantiate this hypothesis by highlighting three technology trends: a) small but powerful single-board computers became popular, such as Rasberry PI [6] with 14 million units sold by 2017 [7]; b) various hypervisors and container environments became available for this class of devices [8], and c) component-based application development [9] and orchestration of containers [10] have gained traction in application and service provisioning.

While it becomes clear that future IoT systems pick up concepts and mechanisms from virtualisation, slicing and code portability, limited research has been conducted on the impact of those technologies on IoT security architectures and mechanisms. A recent analysis [11] shows that network security, data encryption, API security and authentication and PKI infrastructures are among the most required technologies.

Our approach is to focus on authentication and authorisation mechanisms in IoT systems that allow for virtualisation, slicing and code portability. However, many virtualisation and slicing technologies are still evolving. For example, Docker container security is increasingly addressed [12, 13], however not yet solved. This status impacts the engineering of ICT systems

in general and of IoT systems in particular.

To overcome this dilemma, we applied a prototype-based approach for identifying the design cues to for security architectures and mechanisms in virtualised IoT systems. We start our investigation by analysing and comparing today's IoT architectures and draw parallels from the use-case of AI system engineering using a virtualised AI lab, denoted as secure *VP*. The lab permits the usage of a marketplace for AI artefacts and pipelines for component-based development. The use case is of particular challenge since it includes code mobility, multi-party collaboration including security, DRM, and privacy requirements for the aforementioned parts and edge devices. The *VP* aims at simplifying and orchestrating the security and security policy management. That means, the *VP* reliefs the system designer from addressing the security mechanisms and management, including security policy enforcement (*e. g.*, for DRM and data privacy). The system designer can focus on the AI application.

The paper is organised as follows: Sec. 9.2 describes our design concept, analyses and compares the various IoT architectures. Sec. 9.3 describes a component-based AI development concept, its requirements and the suggested architecture for the *VP*. Sec. 9.4 focuses on the authentication and authorisation of users, hosts, and code in virtual labs using code mobility. Sec. 9.5 discusses the lesson leaned and generalisation of the mechanism, incl. their vulnerabilities. Sec. 9.6 summarises our finding and provides a brief outlook.

9.2 Towards Future Secure, distributed and flexible IoT system Architecture

9.2.1 Design Concept

Engineering security is a multidisciplinary endeavour [14]. Designing ICT architectures, and IoT systems, in particular, is typically a rigorous and disciplined system engineering task [15]. However, it is also acknowledged in [15] that there is an increasing disagreement between theory and practice in system design. This gap is mainly accounted to the huge push for fast system integration, *e. g.*, by market needs. We additionally attribute it to the lack of means for modelling and validating techniques for large-scale systems using pre-maturing technologies. To overcome this dilemma, [15] establishes the

vision of autonomous IoT systems that can cope with the complexity through tighter integration of computing elements and compensating the reduced human intervention by enabling self-adaptation. While we believe that self-management mechanisms are of future importance, fully autonomous security control is hard to establish currently. Therefore, we focus on adaptivity in security, *c. f.,* Sec. 9.2.2 for a definition, as a first approach.

Requirements versus Architecture

Our scientific approach is not to design a fully flexible IoT architecture from scratch. We derive our design cues for security architecture from the interaction of security requirements and system architecture. In order to have a valid body of requirements and possible solutions, we consider the challenging use-case of collaborative AI system design.

From a distance, requirements specify what a system must be capable and the architecture describes how a system will be organised and behave such that it will fulfil these requirements. Hence, requirement engineering precede typically architecture design. However, when using immature technologies, architectural design choices easily influence these techniques. Therefore, we start our approach to security design by considering the IoT architectures first, *c. f.,* Sec. 9.2.2 and then define requirements for security mechanisms, in particular, authentication, described in Sec. 9.3.

9.2.2 Comparison of IoT Architectures

Fig. 9.1 outlines the three major IoT architectures introduced in Sec. 9.1. In general, IoT architectures consist of three different layers, namely: a) the *Things Layer*: sensors and actuators, b) the *Network Layer*: forwarding data or service routing (including access points and gateways), and c) the *Application Layer*: implementing data processing and control.

Table 9.1 shows a feature-based comparison of the architectures displayed in Fig. 9.1. The following features were used in the comparison: compute and storage location, service chaining, slicing, enabling collaboration, flexibility and adaptivity towards application workflows, scalability, and finally, security and trust.

The original IoT concept demonstrated impressively the capabilities of the applications and the scalability of the Internet as a transport platform.

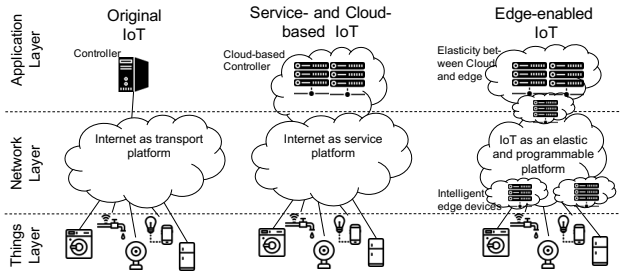

Figure 9.1: Types of IoT architectures

Feature	**Original IoT**	**Service- and Cloud-based IoT**	**Edge-enabled IoT**	**Virtual Preminse**
Compute location	at controller	in Cloud	elastically at Cloud / edge	elastically and programable at Cloud and edge
Data location	at controller	in Cloud and service proxies	elastically at Cloud / edge	elastically at Cloud / edge
Service chaining	not indented	in Cloud	in Cloud and edge	per pipeline in Cloud and edge
Slicing	not indented	in network (when using SDN)	in Cloud and network (when using SDN)	application overlay per pipeline
Collaboration	single application	in Cloud	in Cloud (not across edge)	per pipeline
Flex. / adaptive to app. workflow	static by design	in Cloud	elastic and programable	through programmability and orchestration
Scalability	by transport and processing capacity	by Cloud and service elasticity	by Cloud and edge elasticity	per pipeline through Cloud / edge elasticity
Security and trust	static by secured transmission, nodes and PKI	static by secured transmission, nodes, service API and PKI	predefined by architecture [16]	**flexible per pipeline through programmability, policies and orchestration**

Table 9.1: Comparison of the different types of IoT implementation

Service- and Cloud-based IoT enabled a larger range of applications by augmenting the original IoT with the elasticity and flexibility of Cloud and service computing. Edge-enabled IoT permits fine-grained resource elasticity between Cloud and edge and also the slicing of the network segment. All in all, over the years the IoT concept became more flexible in terms of resource usage. However, Table 9.1 also reveals that these IoT concepts still lack in terms of flexibility of security. Flexible security, of course, aims at high security but also at specialising and adapting the security to the application needs, *e. g.*, specific data privacy is enforced at participating Cloud and edge nodes. In addition to security, trust is needed for collaboration. Hence, mutual trust must be adaptively established between those entities who would like to cooperate on data or application.

9.3 Pipeline- and Component-based Development of AI Systems

The concept of a pipeline- and component-based system development was recently successfully applied for engineering data-driven AI systems [17]. Its main objective is to reduce design costs and time by re-using knowledge manifested in so-called *AI artefacts*. These artefacts are objects, code and functions used in AI development. Multiple artifacts can be chained to form an *AI engineering pipeline*. Artefacts and pipelines enable AI specialists to focus on AI functions and to exchange the results with another specialists. This approach may improve the quality of the AI solution by use of specialisation. Google has recently published a concept for a Cloud-based pipeline [18] and the H2020 project Bonseyes suggests a distributed one [19].

9.3.1 AI Pipeline

An excerpt of a Bonseyes AI training pipeline for Keyword Spotting (KWS) is depicted in Fig. 9.2, *c. f.*, [20]. The KWS applies a Deep Neural Network (DNN) model for identifying keywords in spoken sentences. This pipeline includes four steps: Data Sourcing, Data Preparation, Training, and Target Benchmarking. The steps are needed to obtain and pre-process data and

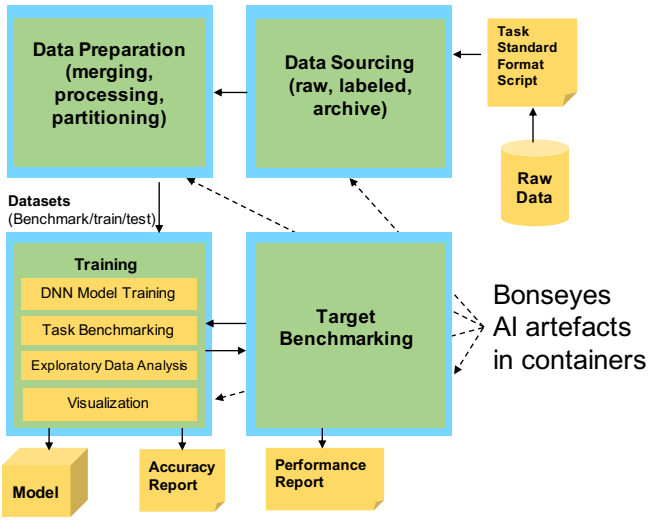

Figure 9.2: An AI data collection and training pipeline [20]

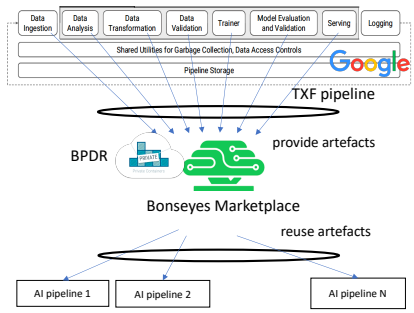

Figure 9.3: Bonseyes AI marketplace, using a pipeline [18]

to train and benchmark the DNN model. Each step is embedded in an artefact and instantiated as a Docker container. The output of an artefact is handed over as input to the succeeding one. Hence, the pipeline forms an *AI service function chain*. In Bonseyes, the pipeline is distributed, *i. e.*, Docker containers are executed on arbitrary hosts (including datacenter or local developer workstations). The outcome of the training is an AI model that can be converted into code, which can even be deployed on IoT devices with adequate computational capabilities.

9.3.2 AI Marketplace

A core element of the Bonseyes project is the *AI Marketplace (MP)*, *c. f.*, Fig. 9.3, which enables the exchange of AI artefacts and the component-based AI design using artefacts. Developers can develop artefacts and upload them into the MP as containers. The containers are stored by the MP in the *Bonseyes' Private Docker repository (BPDR)*. In turn, AI specialists can search the MP for specific artefacts, download them from the BPDR, and place them into pipelines. In this way, the marketplace enables *collaborative development*, *i. e.*, multiparty cooperation in AI design.

9.3.3 Component-based Development Challenges

Bonseyes use of MP for AI artefact exchange imposes these specific *high-level challenges on security related functions*:

1. Protection of DRM (digital rights management): code and data are of value and thus it is reasonable that their owners want to protect and

manage their digital rights.

2. Protection of privacy and data ownership: individuals may be willing to share private data for the purpose of AI training but may not be willing to waive data privacy and data ownership.

3. Compliance in collaboration: digital rights, privacy and data ownership need to be obeyed across various physical locations and legal entities when collaborating.

4. Fair enumeration: individuals, developers or companies need to be reimbursed when their digital intellectual property or data is used.

These high-level security challenges need to be translated into *technical security solutions*: An AI marketplace architecture is required, which enables authorised access, integrity and compliance verification for artefacts in a distributed environment, *i. e. ,* across various physical locations. Moreover, the use of virtualisation technology for artefacts requires a) the compliance verification of the virtual environment at a host, as well as b) the validation of the artefact integrity, *i. e. ,* that the Docker image carrying the artefact was not changed. Hence, authentication, authorisation, and policy enforcement are the core elements of our solution.

9.3.4 The Virtual Premise (VP)

Our solution for solving the technical challenges is to enhance the MP by a *secure VP*, *c. f. ,* Fig. 9.4. The core functions of the VP are authentication, authorisation and policy enforcement. VP places the AI pipeline in the centre and builds a virtual boarder around it. The pipeline is instantiate on federated authorised resources. The pipeline elements/artefacts can be downloaded from the MP (or created and registered in MP using a developer's workbench). The VP uses network slicing and virtualisation techniques to provide isolation of pipelines, prevents unauthorised pipeline communication with the outside and unifies the security and access policies per pipeline. It allows workflows between authorised artefacts through secure programmable and polymorphic interfaces. The VP has similarities to an application overlay and builds a *virtual AI laboratory* per pipeline. The inside of the VP is a trusted area where artefacts comply with each other on DRM, privacy and collaboration policies, while the outside is untrusted.

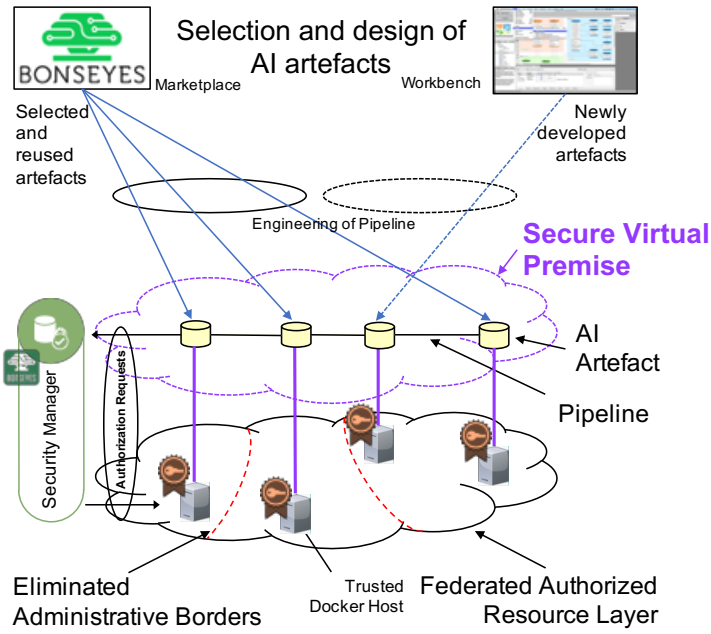

Figure 9.4: Architecture of the Virtual Premise

The artefacts, hosts and users of a pipeline must be authenticated by the *Security Manager (SM)* in order to join the VP. The policies of the VP are enforced locally on the Docker hosts participating in the VP. The split between security policy decision entities and enforcement points is discussed in *c. f.*, Sec. 9.4.

In terms of IoT architectures, the VP can be considered as an enhancement of the Edge-enabled IoT concept which focuses on slicing with respect to adaptive and flexible security policies, *c. f.*, Fig. 9.5. The IoT features of the VP are summarised and compared with other IoT architecture in the last column of Tab. 9.1. As can be observed, the VP enables flexible security and trust per AI pipeline or IoT service chain.

9.3.5 Bonseyes AI Artefact Architecture

The second major component of the VP concept is the architecture of AI artefact, which is shown in Fig. 9.6. The artefact is implemented as a Docker container which is enhanced by a specific *Bonseyes Layer (BL)*. The BL serves three main purposes: a) to control communication with the artefact; b) to

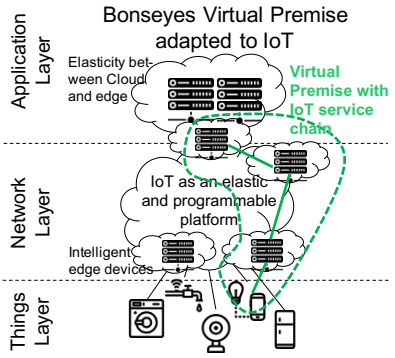

Figure 9.5: Virtual Premise as an IoT Architecture

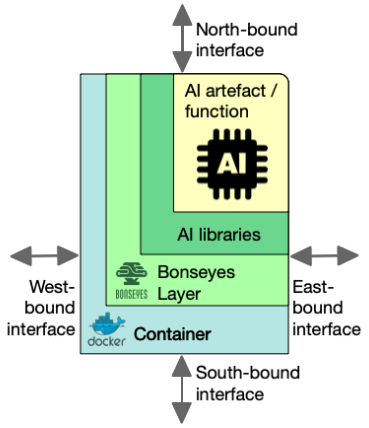

Figure 9.6: A Bonseyes' Docker container comprising an AI artefact.

enable secure APIs to access AI functions and content; c) to facilitate artefact authentication with entities from the Bonseyes eco-system; and d) to verify the functional integrity of the underlying host platform, for maintaining the confidentiality and integrity of data processing and data exchange (*e. g.*, enforcing access control and DRM services). The BL is stsrted as soon as the container is executed, thus ensuring that all communication into the container is intercepted and secured. The secure APIs for the workflow between artefacts are referred to as the *east-bound interface* and the *west-bound interface*, respectively. The API for the execution of the AI function and towards the workbench is denoted as the *north-bound interface*, while the one towards the host (including virtualisation environment and filesystem access) is denoted *south-bound interface*. The AI function together with associated executable code is stored in component called *AI payload* and the required libraries for this function can be implemented as a layer in the Docker image.

9.4 Authentication and Access Management

The MP offers an interface through which artefacts can be uploaded into or downloaded from the BPDR. In addition, the MP supports payment services for commercial artefacts. AI artefact usage can be conditioned by licensing

terms, privacy requirements or local legislation. Furthermore, hosts available for artefact execution may belong to various administrative domains, each free to enforce different policies in terms of which artefacts and users are allowed to execute on their systems as well as how resource allocation should be performed. Therefore, access to hosts and artefacts is coordinated through the SM to enable compliance with this type of constraints through a distributed license management system.

In addition, the SM is the fundamental component that enables mutual authentication and functional integrity checks for the components belonging to a VP. This involves both the AI artefacts (*i. e.*, the containers and their contents) as well as the supporting infrastructure (*i. e.*, the server hosts executing the containers). The main concern in this case is that hosts or containers are tampered with in order to circumvent the license management system and bypass the usage constraints associated with an artefact (*e. g.*, using the artefact without a valid license). However, as past experience has shown, being able to counter *any* attack on the license management system, including those where malicious code is injected into the system, is a very difficult issue. Therefore, we make a set of reasonable simplifications, shown below, to relax the difficulty of the problem.

9.4.1 Assumptions for the Security Model

We assume that the SM is not compromised and that administrative domains are not malicious, do not collude with attackers and do not instrument the hosts to bypass the license system. Also, we assume that Docker images uploaded to BPDR are searched for malicious software before becoming available for download. Furthermore, we assume that the SM is equipped with a Certificate Authority (CA) that enables it to issue certificates for other entities in the MP. Essentially, the SM is the trust anchor for the system. Finally, the SM is equipped with a HTTPS server capable of certificate-based authentication.

9.4.2 Authentication and Authorization Protocol

Given the aforementioned assumptions, we have defined a protocol that verifies if a host is allowed to execute an artefact, and similarly if the artefact is allowed to execute on the given host. The actors involved in the protocol

Table 9.2: Security entities in Bonseyes

Actor	Roles
Bonseyes layer (BL)	Located inside the Docker image. Manages the authenticity of the artefact.
Bonseyes module (BM)	Located on the Docker host. Manages the authenticity of the host.
Bonseyes' Private Docker repository (BPDR)	Remote-accessible storage for Docker images containing artefacts.
Marketplace (MP)	Virtual meeting place for Bonseyes users. Enables artefact, host and user registration as well as artefact exchange with the aid of the SM.
Security Manager (SM)	Manages identities and crypto material and facilitates authentication and authorization. Contains a certification authority (CA). Trust anchor for the entire system.
User	AI developer creating and executing pipelines.
Virtual Premise (VP)	A set of Bonseyes registered hosts (each having a valid BM). Used for executing one or more pipelines.

and their roles are listed in Table 9.2. The protocol consists of a set of discrete phases listed below and shown in Fig. 9.7:

1. Artefact, host, and user registration
2. Host authentication
3. Image integrity check
4. Host (BM) authorisation for execution of artefact
5. Image execution (BL start)
6. Artefact (BL) authorisation for execution on host

In general, phases 1 and 2 can be performed at any time. However, given a specific artefact-host-user triplet, the other phases cannot execute correctly before phase 1 and 2 are completed successfully. Artefact registration entails that an image is created and uploaded to the BPDR. Policies concerning the artefact execution can also be configured during this step. Then, the SM adds the BL layer to the image. It also inserts an artefact identifier (a human readable string) which is then signed with the SM private key yielding the *artefact ID*. This ID is shared by all instances of the same

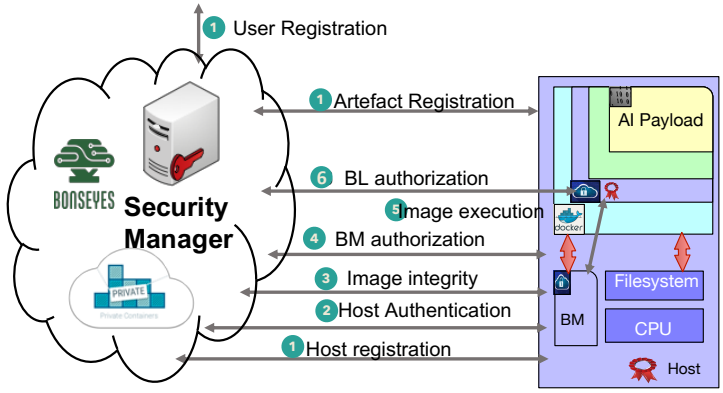

Figure 9.7: Mutual authentication and authorization

image and is used when the artefact states its type in phase 6. Finally, the SM's root certificate and any intermediate certificates required for signature verification are added to the trust database in the image. A message digest (*e. g.*, SHA-256) is computed over the final image yielding the *image ID*. The artefact ID and the image ID are stored as a pair in the SM database.

For hosts undergoing the registration process, the first step is to compute their *hardware ID* (*e. g.*, by reading specific entries in the SMBios). The hardware ID is signed with the host owner's private key, thus tying the identity of the owner to the host. The signed hardware ID is denoted as the *host ID*. If the host platform contains a trusted platform module (TPM), the hardware ID can be replaced with the Endorsement Key (EK) certificate for the TPM. The owner (typically the system administrator or the AI developer) is responsible for verifying that hosts are free from malware before taken in use. As part of the registration, the owner can also specify if there are restrictions in the use of the host (*e. g.*, if it is reserved for a specific developer, or restrained from executing artefacts from a specific vendor).

The hardware ID, the host ID and any usage policies are stored in the SM database. The SM generates a software component called Bonseyes Module (BM), which is installed on the host. The BM plays the same role for the host as the BL does for the artefact. Inside the BM there is an asymmetric key pair and the corresponding certificate, both generated by the SM specifically for this hardware ID. The SM's root certificate and any

intermediate certificates are also present inside the BM.

Developers and host owners need also to register and obtain user IDs and user certificates (*e. g.*, in PKCS#12/PFX) format). This is a one time process, and the ID is carried over to any pipeline or VP that the developer is working on. The certificates enable seamless client authentication (*e. g.*, from a web browser).

The hosts are configured to start the BM after the operating systems has booted up. This is where phase 2 begins. The BM will open a HTTPS connection to the SM and use the host ID (and EK certificate[1], if available) to authenticate itself with the SM. The SM will refuse to authenticate hosts that are missing from registration database or for whom the authentication fails for any other reason. If the authentication succeeds, the SM marks the host as available resource. Thanks to the cryptographic material installed in the previous steps, both client and server authentication can be performed when the HTTPS connection is established.

Phase 3 begins when a developer instructs the SM to assign the pipeline to a VP for execution. Each artefact can be pinned to a specific host in the VP or the SM can automatically assign them to hosts based on a predefined policy. The SM coordinates with the BMs in copying the artefacts to the hosts and in checking the integrity of the received Docker images using strong message digests (*e. g.*, SHA-256).

During phase 4, the BMs request from the SM policies surrounding the execution of the artefact under the controller of the developer. The requests are transported over HTTPS and each request contains the message digest of Docker image, the host ID, the user ID and a nonce. The message digest allows the SM to determine the identity of the artefact being executed, the host ID provides the identity of the host, and user ID denotes the developer executing the artefact. The purpose of the nonce is to help SM in constructing identifiers that can be used to differentiate between multiple instances of an artefact running on the same host. The requests are transported over HTTPS.

If the request is accepted, the SM constructs an entry consisting of five items: image digest, user ID, host ID, nonce and host IP address. This combination, *runtime ID*, uniquely identifies the artefact instance about to

[1] In this case, formal remote attestation is possible.

be started. The reason for storing the IP address is to enable SM to open connections towards the host, if necessary. The SM looks up an existing *host policy specification*, based on the contents of entry. If one is found, it is associated with the entry and stored in an internal database. The policy specifies the constraints of the host in running the artefact under the control of user ID. If the policy allows artefact execution, the SM creates an asymmetric key pair and a certificate for the artefact instance. The runtime ID, the associated policy specification, and the key pair and certificate (if created) are together signed with the SM's private key. The signature and the signed items define the *host license*, which is sent to the host BM over the HTTPS connection established previously[2].

If the received license specifies that the host is allowed to execute the artefact, phase 5 can begin and the BL is started. Otherwise, an error message is sent to the developer and the artefact execution is aborted. When the BL is started it expects to find a copy of host license from the BM. This enables phase 6 to begin. The BL uses the SM public key to verify the signature from the license and determine if it is being executed on a genuine host. If that is the case, the BL constructs a request for SM consisting of the license from BM, its artefact ID, and a nonce. The BL is not aware of its own image digest. Therefore, the inclusion of artefact ID in the request lets SM detect if the license presented was generated for a different type of artefact. The request is sent to the SM over HTTPS. The SM uses the artefact ID to lookup the image digest stored during phase 1 and compares it with the one stored in the license. If there is no match, an error is returned to the artefact which stops execution. Otherwise, the SM looks up an existing *artefact policy specification*. If none is found, a default one is used. The looked up artefact policy is associated with the runtime ID maintained by SM. The SM prepares then a reply consisting of the runtime ID and policy specification. These items are signed by the SM and the resulting *artefact license* is transported over the established HTTPS connection to the BL. The BL learns through the license if it is allowed to execute the artefact code, or if it is supposed to cease execution. Also, the BL is expected to forward the license to the BM before a timeout occurs. In case of a timeout, the BM will stop BL's execution. Otherwise, the protocol completes successfully

[2] The BM will be able to observe the private key transmitted to the BL, but according to the assumptions in Sec. 9.4.1, the hosts are not instrumented to exploit this situation in an undesirable way.

having created mutual trust and the pipeline can begin execution.

9.5 Generalisations from the VP as Design Cues for IoT Systems with Flexible Security

The specifications of the high-level security and trust requirements and implementation of mechanisms for collaborative AI design using pipelines and artefacts gives detailed insights into the security architectures and mechanisms for IoT system using similar concepts. We will generalise our findings:

1. *Collaborative and distributed workflows in IoT service chains require security and compliance verification and enforcement along the workflow,* i. e. , *in an "horizontal way" (authentication for cooperation).* The policies, *e. g. ,* for DRM or privacy, need to be enforced on the involved IoT element, *e. g. ,* edge devices, along the east/west-bound interfaces of the artefacts.

2. *Code mobility in IoT,* i. e. , *use of portable container, requires security and compliance checking on the IoT host,* i. e. , *in a "vertical way" (authentication for resources)* .

3. *The use of a virtualisation concepts and of containers in IoT need the verification of integrity of containers and hosts.* Therefore, the container and the host need to implement a module/layer to verify each others integrity.

4. *Docker's container technology is a great tool for component-based system design for AI system and probably also for IoT service chains.*

5. *Docker's mechanisms to ensure that code in the container can not be by-passed are still limited.*

6. *A centralised Security Manager is typically a single-point-of-failure/attack and needs to be distributed in future.*

An initial analysis of the security threads against the VP is given in [21]. It lists 13 simple threats by inexperienced malicious users, ranging from by passing license constraints to artefact execution on wrong VPs, and

techniques to protect against them. While these techniques are not yet fully implemented in the Bonseyes MP and VP demonstrator, their availability suggests that VP-enabled architectures can be secured against them. More difficult to solve are advanced threats from skilled malicious users. A major approach is to raise the overall trust level by using TPMs.

9.6 Conclusion

The security architecture for the AI marketplace and the authentication protocol described in Sec. 9.4 are the main contributions of this paper. In addition, the investigation of the security, DRM, and trust requirements for pipeline-based collaborative AI system design and the specification of the VP enabled the investigation of similar concepts for future IoT architectures. These IoT systems will apply service chains where flexible and adaptive security per service is needed.

The suggested solution builds a *VP* which forms a virtual boarder around the pipeline or service chain. This chain facilitates also the slicing of the full infrastructure, *i. e.* , of the application infrastructure and of the network. The use of containers in such IoT systems requires the authentication of both, containers and execution environments.

Although only Docker containers were considered in this work, our solution is easily extensible towards other types of virtualized environments such as KVM, LXC, or VMware.

Container and virtualisation technologies are still rapidly evolving in their capabilities. Hence, their future capabilities might impact the features of the VP, *e. g.* , the VP might need to enforce that a container starts always the Bonseyes Layer either by Docker mechanisms or by other verification mechanisms. Another future feature for the VP might be the automatic policy compliance matching, enforcement and eventually policy orchestration, *e. g.* , using distributed ledgers [22]. Another future enhancement is the use of distributed TPMs [23].

Acknowledgments

The authors would like to thank the Bonseyes team for extensive discussions, in particular our gratitude goes to Tim Llewellyn, Samuel Fricker, Lorenzo Keller and

Yuliyan Maksimov. This project has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 732204 (Bonseyes). This work is supported by the Swiss State Secretariat for Education Research and Innovation (SERI) under contract number 16.0159. The opinions expressed and arguments employed herein do not necessarily reflect the official views of these funding bodies.

References

[1] F. Mattern and C. Floerkemeier. "From the Internet of Computers to the Internet of Things". In: *From active data management to event-based systems and more*. Springer, 2010.

[2] ITU-T. *Overview of the Internet of Things*. ITU-T Recommendation ITU-T Y.4000/Y.2060 (06/2012). June 2012.

[3] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. "Fog Computing and its Role in the Internet of Things". In: *Proceedings of the First MCC workshop on Mobile cloud computing*. 2012.

[4] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella. "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration". In: *IEEE Communications Surveys & Tutorials* 19.3 (2017).

[5] V. Sciancalepore, F. Cirillo, and X. Costa-Perez. "Slice as a Service (SlaaS) Optimal IoT Slice Resources Orchestration". In: *Proc. of GLOBECOM 2017*. 2017.

[6] Raspberry Pi Foundation. *Raspberry Pi*. www.raspberrypi.org.

[7] L. Tung. *Raspberry Pi: 14 million sold, 10 million made in the UK*. ZDnet Information available at:https://www.zdnet.com/article/14-million-raspberry-pis-sold-10-million-made-in-the-uk/. 2017.

[8] P. Bellavista and A. Zanni. "Feasibility of Fog Computing Deployment Based on Docker Containerization over RaspberryPi". In: *Proc. of the 18th Int. Conf. on Distributed Computing and Networking*. Hyderabad, India, 2017.

[9] P. Mohagheghi and R. Conradi. "An empirical investigation of software reuse benefits in a large telecom product". In: *ACM Transactions on Software Engineering and Methodology (TOSEM)* 17.3 (2008).

[10] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes. “Borg, Omega, and Kubernetes”. In: *Queue* 14.1 (Jan. 2016).

[11] G. Press. *TechRadar: 6 Hot Internet of Things (IoT) Security Technologies*. Forrester Research Information available at: https://www.forbes.com/sites/gilpress/2017/03/20/6-hot-internet-of-things-iot-security-technologies. 2017.

[12] H. Gantikow, C. Reich, M. Knahl, and N. Clarke. “Providing Security in Container-Based HPC Runtime Environments”. In: *International Conference on High Performance Computing*. Springer. 2016.

[13] R. Yasrab. “Mitigating Docker Security Issues”. In: *arXiv preprint arXiv:1804.05039* (2018).

[14] C. C. Wood. “Why information security is now multi-disciplinary, multi-departmental, and multi-organizational in nature”. In: *Computer Fraud & Security* 2004.1 (2004).

[15] J. Sifakis. “System Design in the Era of IoT? Meeting the Autonomy Challenge”. In: *Proc. of the 1st Int. Workshop on Methods and Tools for Rigorous System Design (MeTRiD 2018)*. 2018.

[16] R. Roman, J. Lopez, and M. Mambo. “Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges”. In: *Future Generation Computer Systems* 78 (2018).

[17] S. Yegulalp. “Data in, intelligence out: Machine learning pipelines demystified”. In: InfoWorld, May 2017.

[18] D. Baylor, E. Breck, H.-T. Cheng, N. Fiedel, C. Y. Foo, Z. Haque, S. Haykal, M. Ispir, V. Jain, L. Koc, et al. “Tfx: A tensorflow-based production-scale machine learning platform”. In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2017.

[19] T. Llewellynn, M. Fernández-Carrobles, O. Deniz, S. Fricker, A. Storkey, N. Pazos, G. Velikic, K. Leufgen, R. Dahyot, S. Koller, et al. “BONSEYES: platform for open development of systems of artificial intelligence”. In: *Proceedings of the Computing Frontiers Conference*. ACM. 2017.

[20] M. de Prado, J. Su, R. Dahyot, R. Saeed, L. Keller, and N. Vallez. “AI Pipeline-bringing AI to you. End-to-end integration of data, algorithms and deployment tools”. In: *arXiv preprint arXiv:1901.05049* (2019).

[21] V. A. Mehri, D. Ilie, and K. Tutschku. "Privacy and DRM Requirements for Collaborative Development of AI Applications". In: *Proceedings of the 13th International Conference on Availability, Reliability and Security*. ARES 2018. event-place: Hamburg, Germany. New York, NY, USA: ACM, 2018.

[22] T. Swanson. *Consensus-as-a-service: a brief report on the emergence of permissioned, distributed ledger systems*. Report available at: http://www.ofnumbers.com/wp-content/uploads/2015/04/Permissioned-distributed-ledgers.pdf. 2015.

[23] P. W. et al. "Distributed Usage Control Enforcement Through Trusted Platform Modules and SGX Enclaves". In: *Proc. of 23rd ACM on Symp. on Access Control Models and Tech.* ACM, 2018.