Alma Mater Studiorum – Università di Bologna

DOTTORATO DI RICERCA IN

Ingegneria elettronica, telecomunicazioni e tecnologie dell'informazione

Ciclo XXXII

Settore Concorsuale: 09/F2

Settore Scientifico Disciplinare: ING-INF/03

CYBERSECURITY ISSUES IN SOFTWARE ARCHITECTURES FOR INNOVATIVE SERVICES

Presentata da: Andrea Melis

Coordinatore Dottorato

Prof. Eng. Alessandra Costanzo

Supervisore

Prof. Franco Callegati

Co-Supervisore

Prof. Marco Prandini

Esame finale anno 2020

Licence Notice

…ad Anacleto, possa la tua risata essere

la mia colonna sonora.

"Today, a young man realized that all matter is merely energy condensed to a slow vibration – that we are all one consciousness experiencing itself subjectively. There's no such thing as death, life is only a dream, and we're the imagination of ourselves. Here's Tom with the weather."

Bill Hicks

Abstract

The recent advances in data center development have been at the basis of the widespread success of the cloud computing paradigm, which is at the basis of models for software based applications and services, which is the "Everything as a Service" (XaaS) model. According to the XaaS model, service of any kind are deployed on demand as cloud based applications, with a great degree of flexibility and a limited need of investments in dedicated hardware and or software components. This approach opens up a lot of opportunities, for instance providing access to complex and widely distributed applications, whose cost and complexity represented in the past a significant entry barrier, also to small or emerging businesses. Unfortunately networking is now embedded in every service and application, raising several cybersecurity issues related to corruption and leakage of data, unauthorized access etc. However, new service-oriented architectures are emerging in this context, the so-called services enabler architecture. The aim of these architectures is not only to expose and give the resources to these types of services, but it is also to validate them. The validation includes numerous aspects, from the legal to the infrastructural ones e.g., but above all the cybersecurity threats. A solid threat analysis of the aforementioned architecture is therefore necessary, and this is the main goal of this thesis. This work investigate the security threats of the emerging service enabler architectures, providing proof of concepts for these issues and the solutions too, based on several use-cases implemented in real world scenarios.

Acknowledgements

My PhD has been a wonderful experience that made me grow (I hope) academically but above all as a person.
It would have been impossible to face this journey alone, but I was lucky enough to have so many wonderful people next to me who made this trip a wonderful adventure, and in this small paragraph I would like to thank them. It is impossible not to start with my mentors, my tutors, my supervisors. I hope to become one day even half of the fantastic academics they are, or at least to have stolen some of their curiosity and desire to improve themselves every day; thanks to prof. Prandini and prof. Callegati.
It would have been impossible for me to get the results I obtained without my colleagues, which is why I warmly thank Saverio Giallorenzo and Davide Berardi, fantastic colleagues.
I also thank my laboratory and departmental mates; you guys are not just roommates but much more; friends and confidants; therefore thanks to Federico, Francesco, Gianluca, Bahare, Elisabetta, Edu, Nour, Davide, Chiara and prof. Walter.
A very big thank you also to all the members and students who have had the patience to follow me over these years, especially the fantastic guys of the Ulisse Lab, whom I now consider almost a family. Finally, professionally, thanks to those who made my 7-month experience in Australia, in Brisbane, possible, a unique and unforgettable experience, thanks to prof. Portmann my colleague Siamak and the fantastic Anees and Khan.
In addition to all the colleagues and academic experiences, I have to thank those who have supported me for well over 3 years.
My family: my mom, my dad, uncles, cousins and my sister Camilla; that no matter where, how or when, they will always be by my side, supporting me for any choice of my life. To Carla, who more than any other has seen from the "seat passenger" my

guide on this journey, my favorite travel-mate. To the old friends, the "from south friends", who always manage to make look 1000 km away like a street block. Finally, but obviously no less important, the everyday friends, those known in Bologna, or around the world, who have contributed every day to improve this journey

Contents

List of Figures

List of Tables

List of Abbreviations

SDN	**S**oftware **D**efined **N**etworking
SMAll	**S**mart **M**obility (for) **A**ll
CoT	**C**loud (of) **T**hings
IoT	**I**nternet (of) **T**hings
SOA	**S**ervice **O**riented **A**rchitectures
MS	**M**icro **S**ervice
XaaS	**"X"**Everything (as) (a)**S**ervice
PaaS	**P**latform (as) (a)**S**ervice
IaaS	**I**nfrastructures (as) (a)**S**ervice
SaaS	**S**oftware (as) (a)**S**ervice
MaaS	**M**obility (as) (a)**S**ervice
PbD	**P**rivacy (by)**D**esign
NOS	**N**etwork **O**perating **S**ystem
NFV	**N**etwork **F**unction **V**irtualization
PDP	**P**rogrammable **D**ata **P**lane
AP	**A**tomic **P**redicate
ACL	**A**ccess **C**ontrol **L**ist
BDD	**B**inary **D**ecision **D**iagram
FDD	**F**irewall **D**ecision **D**iagram
FEC	**F**orwarding **E**quivalence **C**lass
API	**A**pplication **P**rogramming **I**nterface
ONOS	**O**pen **N**etwork **O**perating **S**ystem
MDD	**M**odel **D**riven **D**evelopment
OSINT	**O**pen **S**ource **INT**telligence
LAN	**L**ocal **A**rea **N**etwork
PoC	**P**roof (of)**C**oncept

P4	**P**rogramming **P**rotocol-Independent **P**acket **P**rocessors
BGP	**B**order **G**ateway **P**rotocol

Part I

Background

Chapter 1

Introduction

In recent years we have seen a clear trend on the digitization of services, which has literally exploded [21].

The last technological advances in data center development have been at the basis of the widespread success of the cloud computing paradigm, which represented the foundation for software based applications and services, which is the "Everything as a Service" (XaaS) model.

According to the XaaS model, services from every kind of scenarios (such as, smart mobility, health, PA, ecc) are deployed on demand as cloud based applications, with a great degree of flexibility and a limited need of investments in dedicated hardware and/or software components.

This approach opens up a lot of opportunities, for instance providing access to complex and widely distributed applications, whose cost and complexity represented in the past a significant entry barrier, also to small or emerging businesses.

It also allowed public administrations and private company to move to the digital domain their services and business models.

However, new service-oriented architectures are emerging in this context, the so-called services enabler architecture [14].

The aim of these architectures is not only to expose and give the resources to these types of services, but it is also to validate them. The validation includes numerous aspects, from the legal to the infrastructural one e.g., but above all the cybersecurity threats.

Unfortunately networking is now embedded in every service and application, raising a number of cybersecurity issues related to corruption and leakage of data,

unauthorized access etc.

A solid threat analysis of the aforementioned architecture is therefore necessary, and this is the main goal of this thesis.

In particular the structure of this thesis is organized as follows:

Part 1 "Background" will be dedicated to investigate the current literature and to argue the research questions which are motivating this thesis.
In chapter 2 I will review the state-of-the-art of the current new service oriented software architecture, with a particular focus on the development paradigm used.
Based on this analysis i will define an analysis path for such architectures. This path will be a top-down approach based on the orchestration level of the enabled services.
Part 2 "Contributions" otherwise will be dedicated to showcase the results obtained, explaining in details the steps we conducted.
Firs of all I will describe the real-world use case that I developed to support our security analysis, that is:

- a Clearing System in chapter 4.1.
- a Service Enabler Platform called SMAll in 5.
- a Software Defined Full-Stack Architecture in 6.2 called techNETium.

For each of this use case I will provided a detailed threat analysis, that it will include all levels of the architectures: network, control and application.

For each threat analysis then, I will propose a set of solutions that can be included as an architectural component.
Of course, this topic is vast and many future extensions are foreseen. For every chapter I will propose a set of future works to enhance the State-of-the-Art and foster scenarios in which contribution and threats prevention are a central enabler; eventually in Chapter 7 I wrap up the dissertation summarizing the main contributions.

Chapter 2

Security on Emerging Architectures: State of the art

2.1 From monoliths to microservice architectures to software defined architectures

Historically, architectures have always been a big block of codes and components, dedicated to a specific task, exposing a specific service. Since the last decade, we have seen a first evolution of such architectures, which became platforms for creating and managing services and resources, rather than simply being a service.

As main alternatives to these architectures, we consider two architectural styles for the development of distributed applications: monolithic and Service-Oriented architectures. We consider these two architectural styles in the left side of Figure 2.1.

The monolithic architecture represents the simplest example of a distributed application composed of a client and a server. Following the monolithic approach, the server consists in a single program deployed into a host. The server program, depicted in the top-left corner of Figure 2.1, contains the whole logic of the server. As reference to compare with the other architectural styles, we highlight the main logic components in the server program. In doing so, we follow the well-known three-tier approach [43], where the functionality offered by the server program results from the interaction of three software layers: presentation, application, and data handling/storage.

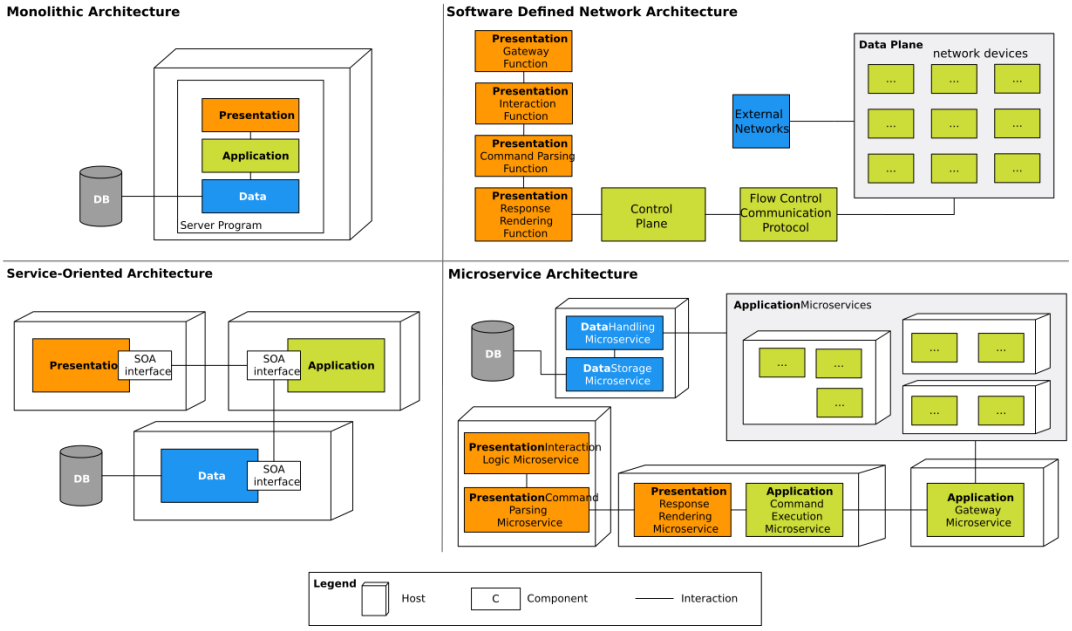


FIGURE 2.1: Evolution of XaaS architectures (top-left, counterclockwise).

From the point of view of the hosting platform, we consider the most recent scenario of Cloud deployment [113] and choose the solution that requires the least management effort for the developer: a Platform-as-a-Service (PaaS) deployment where the management of host hardware, operating system, data storage (represented by the DB component in Figure 2.1), and runtime environment are taken care by the PaaS provider.

Although for small-size applications the monolithic approach is the simplest and most effective, this architectural model shows its limitations when considering two important factors: scaling and code-base growth [54, 186]. Indeed, scaling a monolith, both vertically and horizontally, implies the allocation of resources for the whole server program. Such allocation is usually disproportionate. Inbound requests are rarely distributed in a uniform way over the components of the server program. Instead they tend to stress only a subset of said components modules, making the allocation of the new resources for the less-used components inconvenient [191]. Regarding code-base growth, large-size monoliths are difficult to maintain and evolve due to their complexity: tracking down bugs requires long perusals through their code base; monoliths suffer from the, so called, "dependency hell" [127] for which adding or updating libraries to the program results in an inconsistent code-base that

do not compile/run or, worse, misbehave. In addition, any change in one module of a monolith requires rebooting the whole application. For large-size projects, restarting usually entails considerable downtimes, hindering development, testing, and the maintenance of the project.

With the increasing maturity of distributed systems infrastructure, solutions shifted towards an architectural style that fostered separation and reuse of components. The intention was to reduce the dependency among the components of a distributed application in such a way that they could be deployed in distinct hosts and linked to each other through standardised protocols and interfaces. These design principles took the name of Service-oriented Architecture (SOA) [59]. In our depiction of Service-Oriented Architectures, in the bottom-left corner of Figure 2.1, we divided the three components of the "Server Program" of the monolithic architecture into three services, deployed in different hosts, communicating via SOA interfaces. In Figure 2.1 we marked the possibility to deploy SOA solution over a Platform-as-a-Service system.

Through standardisation and loose-coupling among components, any resource that behaves according to the specification of a component of the system can be employed as such. If said resource fails or becomes busy, it can be replaced with a new, compliant one by simply redirecting its messages. Modularisation also appeals to scalability, both in functions and size. Once deployed, a distributed program can also become a component employed by another program, enhancing its functionalities with minimal effort. In the same way, since components are modular, if the workload on a component can be shared, it is possible to add a new instance of the same component in the system and separate the workload between the two.

Although proposing a sound approach to the development of distributed systems, the first generation of service-oriented architectures (SOA) defined daunting and nebulous requirements for services (e.g., discoverability and service contracts [58]), which eventually hindered the adoption of the proposed model. Microservices, represented on bottom-right corner [96] are a recent reinterpretation of SOAs,

which abandon their unnecessary levels of complexity, focussing on the programming of cohesive [149] components, each implementing only strongly related functionalities. From a technical point of view, microservices are independent components conceptually deployed in isolation and equipped with dedicated memory persistence tools (e.g., databases).[29].

The architectural style of the microservice neither favours nor prohibits any particular programming paradigm. It provides a guideline to partitioning the components of a distributed application into independent entities where everyone faces its own concerns. That is to say that, provided that a microservice offers its functionalities through the passage of messages, it can be implemented internally with any of the main languages mentioned at the beginning of this section. The principle of microservices architectures helps project managers and developers, providing a guide-line for the design and implementation of distributed applications. Following this principle, the developers focus on the implementation and testing of some consistent features. This also applies to those higher level microservices, which deal with the coordination of the functionality of other microservices.

The management of these services, which have become real components on independent (real or virtual) hardware has led to new methods of orchestration emerging from these architectures.

The need to control flows and services has therefore become increasingly necessary, both to create and manage the infrastructure needed to create new dynamic and software-defined technologies, to manage complexity even at network level.

We think trivially about the previous example of a microservice architecture, where a service is the result of the composition of two microservices, deployed with the same infrastructure. The possibility that the two microservices (let us imagine two containers) are totally disconnected, even at hardware level (real or virtual), would allow an extremely greater reuse of the same microservices. For example, it would be possible to perform a second deployment on another subnet, to duplicate it without having to configure each service based on the underlying network. To do so, however, it is necessary to be able to "define" or "design" the network infrastructure.

A non-trivial task if we consider that historically, the first two projects based on

the SDN concept were GeoPlex and WebSprocket. GeoPlex was born from the AT & T laboratories and was based on the network API and the Java language for creating a network middleware that mapped the IPs where activities of interest were performed in one or more services. GeoPlex was, therefore, in general, a platform that managed the association between online networks and services. It was not interested therefore in the operating systems present on the switches or on routers, since ATandT only required a "soft switch" that would allow them to reconfigure physical switches to adapt them to new services on an OSS (Operations Support System, system for operations support). GeoPlex, however, could not radically change the configurations of these network devices, automatically and independently of the systems operational, which thus became a limit, if we consider the current functioning of a network based on SDN.

WebSprocket, on the other hand, was composed of two entities: a NOS (Network Operating System) and a new structured runtime model object-oriented (based on Java) that could be modified in real time by a compiler and a network class loader.

Further efforts, after 2001 with the advent of virtual LANs [95], were carried out by several engineers from various companies, but only with CableLabs, via CableCard (a PC card that allowed users to see and record cable television channels) in 2007, it came to the definition of what we today mean by SDN. Further progress was then made by Berkeley e from Stanford University around 2008. The Open Networking Foundation (ONF) finally came founded in 2011 to promote SDN networks and OpenFlow standardization, a software approach to network design.

Depicted in top-right corner of Figure 2.1 SDN architecture became an emerging architecture able to merge the need of distributed and heterogeneous application with a dynamic network infrastructure. The SDN is an architecture design that is dynamic, manageable and adaptable. The fundamental concept that allows to realize all of this is the decoupling between the actual control of the network by the controller, and the forwarding functions used by the entities that implement traffic routing. This enables direct programming of network control and abstraction of the underlying levels through network applications and services.

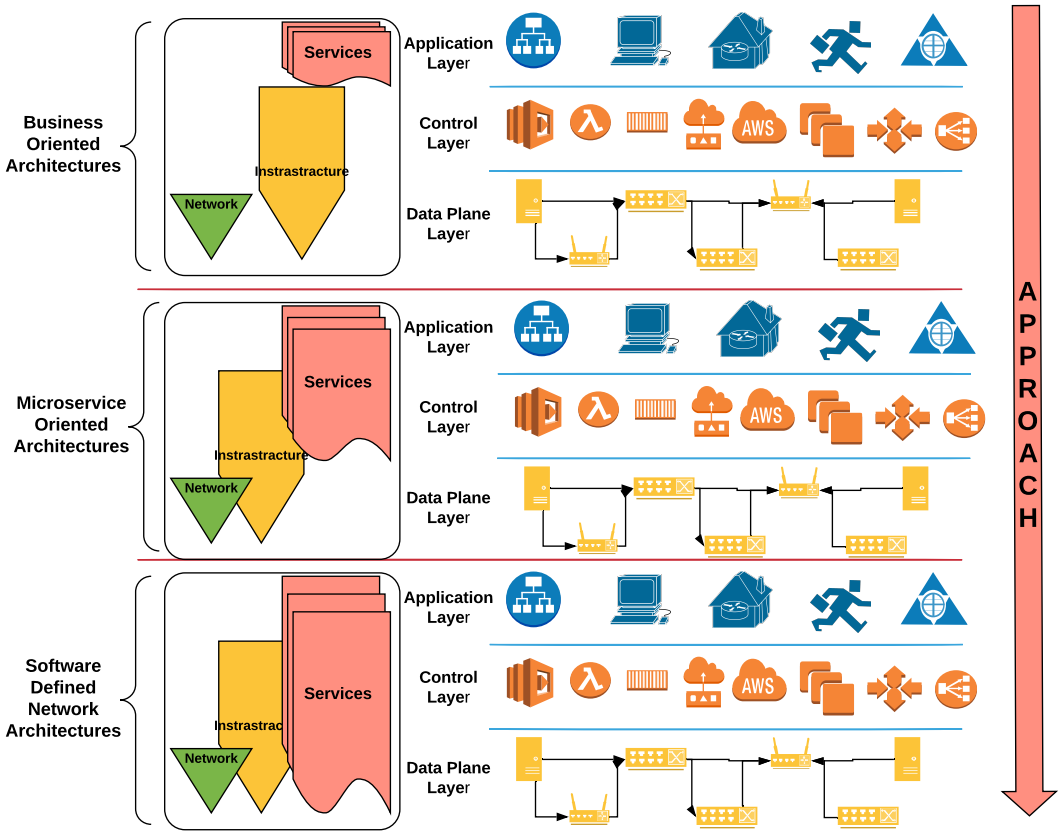


FIGURE 2.2: Top-Down Approach for Emerging Architecture Analysis

2.2 Emerging Architectures a Top-Down Approach

The most difficult task in this context is probably to decide a path analysis. Numerous emerging architectures are increasingly influencing and invading the services market, for this reason therefore it was necessary for our architectural analysis to determine which approach to use.

It was also important to try to start from a very specific context, a specific architecture and generalize as much as possible. For this reason, this work followed a top-down approach. The meaning consists in the fact that we started from a business-oriented architecture, strongly dedicated to the vertical service, descending more and more into the application stack. From the vertical level we have in fact passed to architectures with orchestrated services and therefore more horizontal, up to those at the network level.

This architectural style stack, well depicted in figure 2.2, has been created following the terms on which we usually classify the Architectural Styles:

- Components
- The way in which components are connected to each other
- The data flowing through the components
- The way in which all the above items are configured altogether to build the architecture
- Most important one: the component used to create services

The last differentiation criterion is the most representative one. Descending the stack we follow at which components level the architecture is able to model services. The Business-Oriented ones scratch only the surface of the available infrastructure, using only supports such wrappers, open-data, open-api etc in order to see where there is a possibility of a service, and then create it.

The microservice-oriented are otherwise able to work both on the top level, where the services are singularly exposed, but also on the middle level, which means where the service interacts. The service exposure, the combination of them, and the communications flow become a service opportunity in such level.

As a consequence the bottom layer, the network one, thanks to the SDN paradigm which allow to design and connect dynamically network infrastructures, the service creation opportunity reached the network layer, and in this case we are able to provide solution for all the stack.

According to [173] Software as a Service (SaaS) is a software distribution model in which applications are hosted by a vendor or service provider and made available to customers over a network, typically the Internet. Platform as a Service (PaaS) is a paradigm for delivering operating systems and associated services over the Internet without downloads or installation. Infrastructure as a Service (IaaS) involves outsourcing the equipment used to support operations, including storage, hardware, servers and networking components. The proposed architectures, are therefore a particular customized scenario of PaaS, because as for microservices architectures PaaS provides an infrastructure composed by API, ALS etc. that enable the services once developed, all customized for a specific context.

We believe that this is the true revolution of emerging scenarios, customized and contextualized versions of PaaS that enable the creations of services. From the vertical, business-oriented Architectures, to the more horizontal microservices-based, to the software defined as for the network level.

Such architectures must be seen even as a SaaS architecture because, the platform components could be interpreted as stand-alone services that could be invoked singularly. This is an important consideration if we want to present this as a special case of SaaS and PaaS.

2.2.1 Business Oriented Architectures

The advent of cloud computing and to a lesser extent of the so-called internet of things world has made new business development and business models possible. Previously the architectures were ad-hoc monoliths built for a specific service, that, in order to be created, had to satisfy important economic constraints (due to their high cost). The possibility of sharing resources, data and standards, given by the cloud, made new business model possible.

Modern companies had to respond quickly and effectively to opportunities in today's increasingly competitive and global markets. To promote business agility, companies should simplify (existing) business processes while exposing various packages. Home-oriented applications have been spread everywhere by the companies in a highly standardized way.

The emergence of Web services developments and standards to support automated business integration has led important technological advances in the area of integration software, in particular for those architectures identified as business-oriented architectures. The purpose of this architecture is to satisfy the requirements of free coupling, based on standards, then distributed independently of the protocol computer science and the mapping of company information systems in a manner adequate to the overall flow of the business process.

The peculiarity of these software architectures is that they are created with a precise and well-structured business model. The architecture, therefore, allows the creation of a single type of service, which can be generalized horizontally only to certain contexts but still vertically in the business model.

In these domains, the components that take part on a business process are a subset of the participants in the service-oriented domain. To model a business-oriented architecture we consider roles for process participants, not specific cases, then we talk about process roles, which will be typed by the participating classes.
A business process in a service-oriented domain is essentially a workflow of calls to service messages (between pairs of process roles), and of the internal actions of the roles of the participants (defined activities). The workflow is modelled by an activity diagram and, as we will see in the next section, it is not part of the service definition. In such architectures the service components are managed by simple message passing protocol, and system calls; but this workflow is not part of the service definition.

2.2.2 Microservices/Federated Oriented Architecture

The advent of cloud computing, however, has not only brought the possibility of new business models, given by the possibility of sharing resources and data. Together with the cloud, new software infrastructure paradigms have also been introduced. Hardware and services can, in turn, be shared and modulated in such a way as to create new ones. This composition of services is notoriously known as microservice architecture.

A microservice architecture is a distributed application where all its modules are microservices. A microservice is a cohesive, independent process interacting via messages.

For instance, consider a service intended to compute calculations. To call it a microservice, it should provide arithmetic operations requestable via messages, but it should not provide other (possibly loosely related) functionalities like plotting and displaying of functions.

From a technical point of view, microservices should be independent components, conceptually deployed in isolation, and equipped with dedicated memory persistence tools (e.g., databases). Since all the components of a microservice architecture are microservices, its distinguishing behaviour derives from the composition and coordination of its components via messages.

The advent of the world of virtualization and containerization has accelerated the development of this type of architecture. From the cloud seen as a simple computational space and resources, we have reached the virtualization of entire hardware systems, devices or production environments such as virtual machine dockers. Creating and deploying services through this infrastructure has therefore become much simpler and more automatic, making the management of microservices easier.

2.2.3 Software Defined Network Architectures

The evolution of the infrastructural cloud has also given a strong boost to the network level. The need to be able to orchestrate not only services at the provisioning level, or based on what they exhibit (a datum, an aggregate, an api, etc.), but also at the network level. How they are orchestrated has also become very important. The need arises to have a more dynamic network level on which it is possible to manage and orchestrate complex, heterogeneous and volatile infrastructures. Following these needs and motivations, new architectures were created, this time at the network layer, which fit into this context, the Software-defined networking.

Software-defined networking (SDN) is an agile network architecture designed to allow organizations to deal with the dynamic nature of today's applications. SDN separates network management from the underlying infrastructure, allowing administrators to dynamically manage network traffic to meet the requirements needed.

SDN is therefore a network paradigm in which the forwarding hardware is separated from the logic control, which, in software modules, are called instead controllers (or even Network Operating System, given their similarity to actual operating systems). The goal is to simplify the management of networks, making them more flexible and available to the introduction of new technologies. The controller has a global view of the network, provides the basic services for network management applications, and communicates with the devices by forwarding through well-defined protocols. Forwarding devices do not have instead any logic inside them, but they memorise the tables filled by controllers, which indicate the actions to be performed on specific packages.

Briefly, when a packet reaches a switch, if any rules are indicated in the internal

table for that particular package, the switch will perform the indicated actions, otherwise default actions will be performed, such as communication to the controller of the arrival of a package not yet managed. The controller, the mind of the network, will decide what to do with the package and communicate the decision to the switch.

The main advantages of using SDN are [107]:

- Agility: the abstraction of control from routing allows administrators to dynamically manage traffic.
- Centralized control: the controller has a global view of the network, allowing the configuration of the individual forwarding devices from a single point.
- Third-party services: SDN allows integration into third-party service networks parts (Depending on the controller these services may also be added during execution as external modules. In other cases where the controller is monolithic the controller must be restarted).
- Flexibility, scalability, and efficiency.
- Capital savings: existing hardware can be reconfigured to follow the instructions of an SDN controller, and more efficient devices from the point of view cost-performance can be used with great success, given the simplicity of the forwarding devices used in SDN.
- Global network vision: applications can take advantage of having the same global view of the network, to impose consistent policies.

2.3 Security

Literature started to address the main security issues related to IaaS, PaaS, and SaaS since 2008, with the release of OpenNebula, the first true implementation of a Cloud service. According to [183] an essential list of most important issues encompasses:

- network security (spoofing, sniffing, DoS);
- data security (locality, integrity, segregation, authenticity, confidentiality, privacy, access control);

- authentication, identity management, sign-on process, and authorization;
- web application security;
- virtualization vulnerability;
- availability (high availability, disaster recovery).

The change introduced by these emerging architectures has caused new security vectors to be automatically mitigated, but at the same time new possible attack scenarios have arisen. The software architecture is no longer analysed and tested as a single monolith, but the iterations between the various orchestrated modules, and the management of the infrastructure become the protagonist of safety analysis.

Let's discuss about the evolution of security threats among PaaS-like over such emerging architectures. Considering a smart mobility scenario, to contextualise our example, we argue that the main differences from most PaaS solutions came from its intrinsic openness and flexibility. In this context customers can access available data and services to build and deploy their own services, possibly making them available to themselves or other customers for the same purpose.

A simple example to clarify this process: a one-stop ticketing application orchestrates: *a*) a dynamic planner service that provides the routing options; *b*) a user profile manager to sort them according to preferences; *c*) a real-time availability checking/seat reservation service for each operator; *d*) a set of gateway services for payment.

Every one of these services is available (and useful) as a standalone application. It can be the result of a vertical business model for a specific business service, or a platform for service composition. Moreover, the dynamic planner is not a simple service: it is the result of orchestrating a static planner with real-time information about delays, planned extraordinary events, and disruptions. The branches of this tree can be followed through many levels, until raw data is reached (yielded in a standard form by a wrapper service).

With these considerations we argue that the security issues known for these architectures starts from the well-known typical cloud ones [36]. So hereinafter, we

outline the specific problems that are most relevant for our context of mobility, with the mitigation strategy trends.

2.3.1 Emerging threats

As service enabler architectures, they should assume responsibility for the trustworthiness and reliability of the services; this is unusual for "classic" [118]. While it is unrealistic to expect that they can guarantee complete correctness of data sources and deployed services, especially under the assumption of being substantially open to any customer, they should guarantee at least that security is an essential (and value-adding) service. In particular, it is necessary to define indicators for data quality and service behaviour, and to devise a way to compute their values for complex data sources and services resulting from the aggregation and orchestration of existing ones [61, 55].

As it is made clear in the following discussion, these functions are an important component of a defence strategy against insider threats, which are likely and dangerous in this studied architecture.

As with the choice of architectural analysis, we needed an approach, a path to follow, even with regards to the security threat categories, it was necessary to have one.

As seen the safety challenges are innumerable and of different categories, it was therefore difficult and counter-productive to think of being able to carry out an exhaustive analysis on the whole spectrum.

For this reason we have decided to focus mainly on the so-called insider threat attacks. Statistically, insider threats are one of the most expensive security issues for business companies [142]. Cert Research Center [1] states that almost 30% of known attacks in last years have been made by insiders. One prominent reason of these expensive outcomes is that companies did not foresee all possible malicious insider activities [182]. Indeed, the problem is not the lack of proper countermeasures as much as the difficulty of identifying a malicious insider in the first place. Literature abounds with guidelines and principles aimed at providing general descriptions of the context and the identity of the insiders [64, 40]. However, experts agree that

[1] http://www.cert.org/insider-threat/

the strong contextual variance of threats [171] makes providing a general yet precise identification of all possible insiders difficult.

Data Provenance

One of the most studied issues about data sources security is data provenance [178, 24]. Ascertaining provenance means ensuring that the source of data is verifiable, i.e., that it corresponds to the one declared in the process of creation. In our scenario, provenance protection is a defence against malicious operators claiming to expose data of a competitor, forcing them to gain unfair advantage.

Data Trustworthiness

Data trustworthiness, intended as the possibility to ascertain the correctness of the information provided by a data source, is loosely related to provenance [47]. Ideally, but infrequently, data samples can be independently measured by different users, thus allowing cross-checking and error correction. For original data, i.e., provided by its creator, the trustworthiness score is usually derived from the reputation of the creator. In this case, it is very difficult to block attacks in which, for example, the creator advertises a data source of given quality, but then exposes a degraded version, to keep the advantage of more precise/timely information for itself.

Service Maliciousness

The trustworthiness of a service is an easy concept to intuitively grasp, but difficult to formalize and to verify in an open environment. In the described service context trustworthiness can be associated with its compliance to a declared function, and shall be evaluated before the application is admitted to the platform and, ideally, again at every usage. If a service creates aggregated data, processing various sources, it is necessary to ensure that the computation is correct, that no useful results are hidden (completeness), and input data is not tampered with (soundness). These are all likely opportunities for a malicious insider that succeeds in registering a rogue service. For example, a tampered travel planner could slightly deflect routes

to favor or damage certain businesses; a modified delay-checking application could hide or amplify violations of agreed service levels.

Service vulnerability to external attacks

There are applications that exhibit wrong behaviors not because of their maliciousness, but because of unexpected vulnerabilities to maliciously crafted invocations. In this case, there is an external threat in addition to the twofold insider threat: one in the loose meaning of an insider being so careless as to deploy a vulnerable application, the other in the case that another insider is in the best position to exploit it. For example, a service with extensive access privileges to private data could be tricked to leak it to a service with much more restrictive access rights. Another example, mixing service vulnerability with data provenance issues, is that of a service that provides crowd-sourced information about the status of the road transport network. Failure to implement an effective fraud-prevention mechanism could allow an attacker to inject fake reports to influence the behavior of users.

SDN Threats

Generally, the security of an SDN architecture is built around an automated detection of violations of network invariants on the data plane, possibly in real-time.
The recent work by Beckett et al. [15] shows several tools for finding network misconfigurations. These tools have been classified in two different categories: control-plane oriented, i.e., able to discover buggy configurations pro-actively, and data plane oriented, i.e., able to discover mis-configurations reactively, i.e., observing events while traffic is flowing. Proactive approaches are particularly useful to predict potential network misconfigurations that might lead to security issues (e.g., BGP prefix hijacking). Security attacks, however, are hard to predict. In addition SDN introduced new security threats but has also been able to mitigate many of them [172, 211, 46]. A few comprehensive surveys on the state-of-the-art of today's SDN security solutions is provided by [2, 50, 158, 200]. From the literature it is clear that many solutions have been proposed but, so far, none of them has been standardized. In particular Akhunzada et al. [3] provides a detailed taxonomy of SDN threats. In

this thesis we focus on those classes of threats that compromise the forwarding and monitoring activity. Examples include:

- multiple compromised network boxes as a result of bugged injected flow-rules, that may prevent nodes communicating or may divert traffic flows for eavesdropping;
- inner loops and black-holes (usually difficult to detect via normal network scans);
- flow-rules replacement or removal with the aim of causing unexpected network behavior, Denial of Service (DoS) and Man-in-the-Middle attacks;
- any kind of SDN controller exploitation that results in a compromised forwarding activity.

We argue that the SDN paradigm provides the perfect platform to pursue this goal, given the interleaving between control and data planes: in a SDN network state changes in the data plane are the outcome of what is produced in the control plane and, at the same time, the control plane has a view of the data plane.

2.3.2 Mitigations

Many of the described problems are intrinsic to the concepts of cloud, SaaS, and data sharing. For this reason, before being able to develop a solution for data quality and data provenance, there is a need for a preliminary analysis of all the metrics of these types of solutions.

Data Provenance

The first thing that we must consider when dealing with services that expose or elaborate data is to differentiate between data and information.

According to [41], information systems provide data in a certain business context. When data is used by human beings, it turns into information, and information finally turns into knowledge by being interpreted and linked for a given purpose. Regardless of such clear theoretical differentiation between data and information, practitioners use the term "data" in a broader sense.

As already mentioned, in the context of mobility, verified information is of paramount importance. There is a need to supports the provision of different sources of data along with their associated metadata (e.g., used to verify their provenance). However, it should also provide techniques, embodied by helper services, to transform those data into verified information.

Data Provenance verification is a known problem in literature. Different approaches can be taken to support a solution for the problem of recognizing the source of a data stream. Literature agrees with [79] that the requirements for a provenance management system are:

- Verifiability: a provenance system should be able to verify a process in terms of the actors (or services) involved, their actions, and their relationship with one another.
- Accountability: an actor (or service) should be accountable for its actions in a process. Thus, a provenance system should record in a non-repudiable manner any provenance generated by a service.
- Reproducibility: a provenance system should be able to repeat a process and possibly reproduce a process from the provenance stored.
- Preservation: a provenance system should have the ability to maintain provenance information for an extended period of time. This is essential for applications run in an enterprise system.
- Scalability: given the large amounts of data that an enterprise system handles, a provenance system needs to be scalable.
- Generality: a provenance system should be able to record provenance from a variety of applications.
- Customizability: a provenance system should allow users to customize it by setting metadata such as time, events of recording, and the granularity of provenance.

In a microservice architecture, an application is essentially a collection of workflows. These workflows can compose many levels of services, each processing and

modifying the data before its final destination. What we need is a way to certify the metadata related to a data stream and manage its validity during time and re-elaboration [187]. Literature generally considers four different sets of techniques [179]:

- Subject of Provenance, in which provenance can either be available explicitly or be deduced indirectly.
- Representation of Provenance, in which the way provenance is represented follows from a tradeoff between the cost of recording it and its richness; it is typically implemented either with annotations or with inversion [17].
- Provenance Storage in which the design of metadata is also important to enable scalable storage.
- Provenance Dissemination where, in order to use provenance, a system should allow rich and diverse means to access it.

According to works like [180], this problem could be solved only with a creation of private and public key system for data stream certification. A good reference is the system developed in [198], describing a cryptographic provenance verification approach for ensuring data properties and integrity for single hosts. Specifically, the authors designed and implemented an efficient cryptographic protocol that enforces keystroke integrity. However, public-key schemes are known for their significant computational load, thus existing techniques may not be suitable for high-rate, high-volume data sources. Moreover, there could be the need for an algorithm for the provenance of composed data.

In some cases, data originated from the composition of raw (or otherwise "lower ranked") sources should be accompanied by suitable metadata that allows to verify the provenance of the input values, in a cryptographically strong way. In our context it could be important and useful to capture and understand the propagation of data.

In [209], the authors exploit the propagation of the metadata on the various levels to create a general metadata storage and management layer for parallel file systems, in which metadata includes both file operations and provenance metadata.

Also Merkle hash trees could be a good candidate to build proofs for composed data pieces [128]. The combination of metadata propagation with key distribution

propagation management can guarantee a good level of trust in provenance management systems. Works similar to [83] discuss how to support provenance awareness in spatial data infrastructure and investigates key issues including provenance modeling, capturing, and sharing that can be easily used to implement a key propagation system.

Data Trustworthiness

Rating systems are one possible implementation for trustworthiness evaluation mechanisms. Auditing services, attribute scores to the data-source services based on different criteria. For example, feedback from users of the data source, or a combination of reputation scores when the same data can be fed by many sources and cross-checked. Of course, reputation systems in turn introduce their fair share of problems [100, 122, 159, 112]. Alternative or complementary solutions are anomaly detection services based on machine learning and pattern recognition [51]. Then, as cited for provenance, the emerging architectural context should automatically compute the trustworthiness score of data originated from the composition of other sources, based on their scores.

Service Maliciousness

In principle, the service deployment interface can verify the correctness of an application before accepting it. In practice, this operation is very hard to perform. One indicator of correctness is the compliance to a template of acceptable interfaces for the kind of service the application provides. However, it is very difficult to define templates strict enough to allow sensible compliance checks, but general enough to avoid hindering the deployment of legitimate services. Another way to check correctness is to look at the actual behavior of the application, as it is common in anti-malware checks, through static analysis, verifying the code to discover possible malicious behaviors. These techniques are far from infallible, and their scope falls much shorter than what is required in our context. Indeed, in this context a malicious behavior can be a subtle deviation from the correct calculation [140], which is far more difficult than the detection of traditional malicious behaviors (e.g., damaging or self-replicating ones).

A more promising technique is that of dynamic analysis of malware [60]. A way to discover a malicious behaviour of a service is to implement a mechanism that could guarantee, in every moment, a reproducibility of the results. Taking advantage of the data provenance, certifications of raw data, and of its propagation to results, it is possible to implement a reference monitor to verify the compliance of results to the expected values. In case of conflict between the declared results and the actual ones, it could be discovered what has been tampered with: the source data, or the service logic. In the first case, this detection can also feed the data trustworthiness rating system.

Service vulnerability to external attacks

Main prevention to implement is provide input sanitization, or in general, Intrusion Prevention System / Intrusion Detection System as a service, to protect applications from most of the malicious invocations. Tracing cross-callings between services can thwart insider attacks aimed at privilege escalation. This can be done by taking into account the whole set of access control rules before allowing unauthorized data to leak in or out through a careless application.

Chapter 3

Motivations and Research Question

In this chapter we will outline the major contributions of this thesis dissertation, with respect to what we argued has been the state-of-the-art and its evolution on Chapter 2 and as a consequence of the considerations that we introduced in Chapter 1.

With this short chapter we aim to bind the two parts of the dissertation together – Part I is about how the current research evolved in the last 10 years wrt to SOA and Part II is about what is being presented as a contribution to the State-of-the-Art. In particular, we will localize where our research work fits into the big picture, to extract the challenges and develop specific use cases according to the related scenario on which we focused and, most importantly, why we think it is important and worth it.

First of all we will briefly describe the context in which our architecture study was done. Despite our analysis can be considered as general as possible, to support our arguments it was necessary to create several real use cases. These use cases have therefore fallen into real, functioning and currently in use application contexts. They are obviously derived from projects activated and concluded between our research team, companies in our area of influence, government agencies and other research institutes.

The architectures we have developed have therefore fallen into two main contexts: smart mobility and MaaS, and industry 4.0, which we will describe in the next section.

3.1 Context: Smart Mobility and Industry 4.0

3.1.1 Smart Mobility and MaaS

In recent decades, public transport services steadily profited from the introduction of new technologies. The means of transport became faster, less polluting, more comfortable and accessible. The interaction of passengers with transport services became easier, as ticketing and payment systems went from paper to electronic and on-line planning and real-time information systems became available. Through more efficient exchange of information, transport operators began to see the interoperability between competitors as an added value (for example: coordination at transfer points can lead to better service, attracting more customers than it would happen by aggressively competing for the same route).

Mobility as a Service [164] is an innovative approach to the integration of public and private transport, made viable by the integration in a coordinated infrastructure of the technologies illustrated in the previous section. Born in the city of Helsinki, this paradigm is starting to spread throughout Europe and beyond, aiming to establish standards for the interoperability between different (even in terms of country) operators, and to encourage the creation of alternative solutions to the standard "mass transit / private car" duality, as both new technologies and social trends emerge [124].

Very briefly, the principle of MaaS is that as long as every detail of the demand and supply for transportation services is known in real-time, there is no need for passengers to commit on specific means. Instead, they will enjoy a broad spectrum of alternatives from which to choose, taking into account the needs of the moment. For example, one could specify a very strict set of constraints in terms of comfort and timing, likely to result in a choice of premium means, while another could simply express the need for reaching a destination at the best price, getting a virtual ticket, and receiving real-time instructions about which means to use to complete the trip. Many business models are possible. In the simplest form, a MaaS operator could simply be a smart broker for planning and paying trips on existing networks.
A more innovative way would be selling mobility packages allowing travelers to use pre-configured amounts of usage of different means. From the transport operators

viewpoint, a MaaS platform could be a great opportunity to leverage integration and to exploit unused capacity. For example, a taxi company exposing vehicle availability and position in real-time could offer lower prices during off-peak times, thus appearing as a good alternative to mass transit; data-mining could allow operators to foresee correlations between various conditions (events, weather, accidents) and transportation needs, to allocate materials in the best possible way.
Ideally, the ICT infrastructure enables these models by tracking timing, position, and availability of trains, buses, subways, shared bikes, shared cars, taxis, Uber cars, Lyft cars, etc. in an overall effort of opening data and standardizing the interfaces to access them [193].
In short, in a mobility context, both the users and the operators can benefit from the smart definition of trips, provided enough availability of data and efficiency of processing is available. This is exactly the kind of challenge IoT architectures are up to [165].

Thus, the role of public administration can undergo a significant change. Some administrations could choose to play the role of a central MaaS operator, exerting a stronger control on the local mobility agenda. Others could leave the field to private companies, hoping to benefit from market-driven optimization of citizens' patterns of mobility. They could also accurately monitor citizens, using collected data to plan investments and direct incentives towards specific goals. The first scenario allows for a more respectful and regulated approach to citizens' privacy while the second leaves room for malevolent or misleading collection and use of data.
One example of the latter is Google legal advisor David Drummond's defensive reply to the question about future uses of data collected by their driverless car [154]. According to authors, it is too early to regulate over the driverless-car about data collection and uses, because it is not yet foreseeable what is worth (implicitly 'for the company'). In any case, governments will need to face the challenges MaaS provide, and to think about needed regulatory changes to make it viable for innovative cities. The adoption of MaaS will sustain a transition from a public transport system traditionally coordinated by the government to a multi-faceted system where exert coordination through the help of other actors. For example, determining who is in charge of setting the standards will affect business and consumers in parallel with

data protection policies.

It is worth noticing that, in many places, this change could introduce significant trust issues. The organization of transport infrastructures by public bodies guarantees (at least in principle) that travelers' interests are safeguarded. In a fuzzier scenario, it could be very difficult to verify how sensitive data are processed and by whom, as detailed in Sec. 4.4.

3.1.2 Industry 4.0

The Industry 4.0 (I4.0) paradigm is an initiative started in Germany that is gaining worldwide momentum in smart factories[110].

The level of complexity of communications between sensors, actuators, and control centers required in I4.0 scenarios calls for a new generation of Industrial Networks, with demanding levels of performance, reliability and security.
Such revolution comes in hand with Software Defined Networking (SDN), that also brought a significant paradigm shift: network programmability has introduced several advantages, and not only for programmability of the forwarding plane. At the same time, SDN have significantly expanded the attack surface and the space of reachable and exploitable vulnerabilities, within the (physical or virtual) network as well as within each connected system.

SDN can be used to design industrial networks that meet all requirements derived by current standards, yet its growing complexity calls for new generations of management architectures. Such architecture should be abstract enough to subsume any possible application scenarios within the realms of I4.0 and IoT.

3.2 Security Analysis

3.2.1 The Insider Threat Problem

Having defined the context in which we developed our architectures we performed the security analysis of the enabled and exposed services. Following the state of the art of chapter 2 we can see that the problems are found on different levels of the architectural stack; include numerous agents and for different purposes.

In fact, we immediately noticed how an in-depth security analysis would have been very extensive, and not possible in the timing of our use cases.

Furthermore, through interviews and discussions with our business partners, we have defined that despite the wide spectrum of possible attacks within this type of architecture there was a particularly widespread and sensitive one, namely the insider threat.

As well explained in chapter 5.4, statistically, insider threats are one of the most expensive security issues for business companies [142]. One prominent reason of these expensive outcomes is that companies did not foresee all possible malicious insider activities [182].

For these reasons, our security analysis, although centered on all levels of architecture, will mainly focus on identifying and solving insider threat problems, going superficially, or not considering, the other common agent attackers.

3.2.2 Policy Enforcement oriented Solutions

The proposed solutions as a consequence will be based on preventing the previously described attackers. The study of the solutions in particular will be structured as follows. Through the study of possible threats a categorization of services and vulnerabilities will be developed. A definition of the security policies to be applied to prevent certain attacks will therefore be created on this categorization.

For each policy and architecture a solution will therefore be proposed that is developed, tested and deployed showing as how our study can be applied in a real and practical use-case.

3.3 Use Cases and motivations

Finally, let us try to summarize in this last paragraph of the following chapter how the motivations and the research questions have structured the work of this thesis. The main purpose of this thesis is to study the security issues of some of the emerging software architectures resulting from the evolution of cloud computing, the XaaS paradigm, and more generally the new business models on software architectures. For these studies, however, despite an intense phase of literature and state-of-the-art

study it was necessary to create a real use case on which to carry out tests and evaluations.

Obviously the use case could not be NOT contextualized, for this reason, also thanks to our collaborations, we have created different use cases in the context of smart mobility and industry 4.0. These use cases have been applied in several other parallel and/or complementary projects, resulting in further scientific publications that have just validate more the usefulness of our use case.

We therefore go backwards in the path.

Starting from a specific use case we studied the security issues related to the insider threat. From this specific use case we proposed and used a process of categorizing the elements of the architecture so that it was as general as possible and therefore also applicable to other contexts. From this categorization we therefore produced an analysis of the insider threat security problems, which wanted to be as general as possible.

On this analysis we therefore produced security policies aimed at mitigate these threats. Finally, to validate our research, we proposed a practical implementation of these policies, in a way of a development component of the software architectures created as use cases.

Part II

Contributions

Chapter 4

Privacy-Preserving Design for a Clearing System Architecture

The current trend in management of public transport systems is to outsource services to multiple private operators, requiring them to integrate their ticketing and fare system with one another. When this concept is introduced in regions that used to have single local entities managing every aspect of ticketing, or that conversely left operators free to adopt incompatible, separate ticketing systems, a new layer of coordination must be put in place.

There are many examples of this kind of approach around the world, such as the Oyster card system in London, the Octopus card system in Hong Kong, or the Istanbulkart in Istanbul just to name a few. As a case study, this thesis considers the Emilia-Romagna region of Italy, where the Regional Government has been running for several years a project to integrate the control processes of the various transport companies operating in the region. These companies typically operate over disjoint territories, and they used to manage independent and localized ticketing systems.

The trend with the new regional system is to go more and more towards integration of tariffs, routes and ticketing, so that the citizen may buy a ticket in city by a given operator and use it in another city with another operator. While providing an improved service to citizen this approach also brings some additional burden, since a clearing system is needed to share the revenue of tickets sales according to the actual service each operator has provided (and thus, supposedly, to the real costs it has incurred). Data detailing every trip, collected by public transport operators, previously confined to internal use only, now must be shared and can potentially

harm passengers.

The system that manages it does not merely need to control data disclosure, but has to be designed to manage potential risks during the collection and processing of data. This is a challenging task, which must manage privacy risks appropriately on the one hand, and preserve data utility to a level that guarantee usefulness for clearing purposes on the other hand.

This first contributions chapter of this thesis illustrates the work done and need to design and test the clearing system in a way that safeguards the protection of personal information, not as a result of some policy superimposed to the existing functions, but rather taking into account this requirement from the start, by applying the principles of Privacy by Design.

Contributions Contributions on this Chapter are organized as follows. In Section 4.1 the local context and the general ideas behind the clearing system are briefly presented and reviewed, and research questions are stated. Section 4.2 gives an overview of the general principles of data sanitization for the purpose of safe release of sensitive information. Section 4.3 illustrates the risks connected with the release of sensitive information, focusing on the desanitization attacks that exploit public data sources, and Section 4.4 gives two examples of how these attacks can affect the clearing datasets. Section 4.5 describes the general principles of Privacy by Design, and outlines the direction of current and future work to apply them to the scenario of the clearing system, before conclusions in Section 4.6.

4.1 Context Scenario

4.1.1 The local context

The Emilia-Romagna Region has approximately a population of 4.5 Million with an area of 22500 square Kilometres. About half of the population leaves in the 13 main cities that are lined along the ancient Roman road called "Via Emilia" which gives its name to the Region. Emilia-Romagna is highly industrialized with a number of companies typically spread along the Via Emilia around the main urban centres. For

this reason the mobility infrastructures are a key part of the logistics supporting the economy of the region.

The estimate of daily trips made by Emilia-Romagna citizens for work or leisure is about 9 Millions of which 25% walk or bike and 8% by public transport means. The public transportation system is built around a rail backbone basically parallel to the Via Emilia which links the main urban centres. Local transportation systems in the cities mostly use buses. The local systems are run by four large operators and a few small operators on specific routes.

The policy maker is the regional Mobility and Transport Councillorship which is competent, among many subjects, for the planning of the infrastructural network, regional and local mobility systems. Over the last decade the Councillorship pursued service integration and multi-modality of public mobility systems, promoting, in particular, the deployment of regional integrated fares with an investment of about 20M euro in supporting hardware (central control systems, ticketing machines, vehicle monitoring systems etc.).

The issue of the MiMuovo (I move) chip card was the flagship project of the fare integration process, supporting multi-modal tickets valid over a given path spanning several operators and transport means. For instance a user holding a MiMuovo card with an integrated travel contract is allowed to use the bus (run by operator A) in his home town to reach the railway station, the train (run by operator B) to his/her working town and the bus (run by operator C) to his/her working place. To date about 300,000 MiMuovo cards have been deployed and are used daily.

Trend is also fostering fare integration for single trip tickets that can be bought in any town and used in any other within the Region. This requires the operator selling the ticket to share the revenue with other operators, if they are involved in its use. This is called clearing process, and has to be implemented in a way accepted fairly by the whole set of operators involved, to guarantee the integrated system sustainability.

4.1.2 The clearing Service System

The clearing system is based on a vertical smart mobility service architecture in which each operator is responsible for the management and maintenance of its own

data. The data needed for the computation of the clearing function is collected in a clearing database located in a central processing centre, operated by a regional in-house company, in order to guarantee neutrality and to avoid disturbing the production systems of the operators.

The creation of the clearing database requires the sharing of the operators dataset in a standard, machine readable format, thus creating a possible threat as a consequence of secondary uses. Moreover the regional Councillorship aims at using the data for in-depth analysis of the transport system performance. Eventually, part of the datasets could be released to the public as open data.

Operators and public bodies do not have any effective control over future uses of their dataset once it is publicly available. Unfortunately the data about sales and usage may reveal issues the operators consider part of their industrial secrets and/or sensitive information in terms of personal privacy.

This problem can be (partly) mitigated by applying full anonymization safeguards, which is very difficult when the utility of the database has to be maintained. Moreover, it is possible to adequately inform the involved subjects of the intention to disseminate the dataset in an open data format, alerting them to potential risks, but this action can limit the degree of user acceptance, especially if the policy intentionally leaves open what kind of secondary uses of their data will be done. Therefore a trivial solution to the issue does not exist.

4.1.3 Research questions

From the point of view of the users, in the widest sense that encompasses passengers, operators, and regulators, the most pressing questions to be answered are: with the current data storage and utilization processes, is it possible to breach data privacy and re-identify the data subjects? Which kind of processing and links to people and business-related issues are possible, for example by matching the clearing database data with other external databases? From the point of view of the researchers, these questions can be answered by analysing the underlying scientific and technical tools: what features do the current data sanitization algorithms exhibit? Is it possible to measure their effectiveness in any given scenario? Symmetrically, can the experience gathered from similar projects in other cities/regions point

our research in the right direction, or are our requisites too specific?

Once the background analysis is complete, if it highlights deficiencies either in the basic technologies or in their application to specific scenarios, the research activity will be directed towards the definition of a more effective framework for public transport data sanitization.

4.2 Sanitization: a critical overview

The architecture outlined in Section 4.1.2 introduces two possible security attack vectors. The first one is the intrusion of an unauthorized party, in which data are subtracted from the primary database; this is a classical issue of information security and access control, and this work does not deal with its direct form; yet, it takes into account the similar situation of purposely releasing data for public use, considering that it could be enriched though correlation with external data sources, to the extent of disclosing details that should not be made public.

The second one is called an insider attack, also referred to as an insider threat; this type of attack arises due to a malicious threat from somehow authorized actors, from inside the organizations that are legitimately involved in data collection and processing; the next chapter illustrates ways to perform this kind of attacks and corresponding effects.

A data sanitization phase is commonly proposed in the literature as the necessary step to prevent these issues; this phase as defined by [44], is "the process of altering [a dataset] so that it remains usable for beneficial purposes, while minimizing its use for harmful purposes". To properly define this process, the key issue to deal with is to understand what "keeping the beneficial purposes" and "minimizing the harmful purposes" mean. Ideally, the process should be able to manipulate the data in a way that prevents privacy attacks but at the same time preserves the possibility of performing many kinds of economic computations.

To progress towards this goal, the existing literature is analysed to find (a) whether convincing measures of utility and vulnerability of the dataset exist and (b) whether existing algorithms result in positive trade-offs when applied to our context. The literature was analysed as follows. Starting from the basic requisite of having to

anonymize the data, the generally-applicable techniques of data anonymization were reviewed, highlighting their limitations.

Next, the application of these techniques to the clearing scenario was attempted, taking into account the literature on the evaluation of these techniques, namely verifying the impact of their known limitations, and introducing metrics that allow to determine whether a satisfying level of privacy is attained.

Subsequently, the symmetrical path was followed, starting from similar cases for which documentation of the process of transport data privacy protection exists, such as those of Montreal and Amsterdam. However, the analysis of the various aspects highlighted that, even though there are important points of contact, these experiences did not need to take into account some requirements that turn out to be of critical importance for the clearing scenario. As a consequence, the techniques devised for those systems would leave ours subject to numerous types of attack, both of kinds already known in the literature, and of other kinds described in this dissertation as proofs of concept.

4.2.1 General-purpose sanitization approaches in the literature

As a preliminary consideration, to understand the way algorithms manipulate datasets to achieve the aforementioned results, it is useful to note that every approach is based on the classification of data elements according to the potentially sensitive information in three categories [206]:

Identifier attributes (or identifiers) can individually distinguish the data subject more or less directly. Typical identifiers include: name, address, social security numbers, mobile phone number, IMEI number.

Quasi identifier (or key) attributes can be used to identify a data subject using auxiliary sources of information, by linking to databases that contain identifying information. They are indirect identifiers of a data subject, which make an individual more distinctive in a population. Typical key attributes include: age, race, gender, date of birth, and place of residence.

Sensitive (or secondary) attributes cannot individually identify a data subject directly and may require significant amounts of auxiliary data to be useful for re-identification purposes. A data subject may then be identified individually through

more sophisticated methods such as fingerprinting, rather than mere linking of databases. Examples include settings in an application, battery level measured over time, or location patterns.

In summary, the literature describes four main techniques of data anonymization.

k-anonymity [163, 129] is the most well-known technique for generalization. The basic principle here is to replace exact values with ranges, wide enough to guarantee that every attribute in a database appears with identical values in a given number of other rows, forming a group of k rows indistinguishable from each other. This approach may take the form, for example, of grouping subjects' locations into sufficiently large areas such that no set of locations is unique to any individual.

The enforcement of k-anonymity requires the preliminary identification of the quasi-identifier. The quasi-identifier, as previously defined, depends on the external information available to the recipient, as this determines her ability to make correlations (not all possible external data sources are available to every possible data recipient); different quasi-identifiers can potentially exist for a given table. Many variations and improvements exist, yet k-anonymity techniques cannot hide whether an individual is in the dataset, and it performs poorly in protecting sensitive attributes against attacks based on background knowledge or on the knowledge of the details of its application [188].

l-diversity [116] is an improvement of k-anonymity that require the sensitive attribute associated with each quasi-identifier to appear at least with l different values. Other refinements have been proposed, but as described also in [195] processing a large dataset to achieve l-diversity is time-consuming and vulnerable to inference attacks in presence of a series of updated publications of the same dataset, if it is simply re-anonymized with the same approach every time. Finally the added requirements proved to be neither necessary nor sufficient to prevent sensitive attribute disclosure (Li et al., 2007) [111].

t-closeness [111] To improve robustness of k-anonymity, the same authors of l-diversity also proposed a privacy notion called t-closeness, which requires that the

distribution of a sensitive attribute in any equivalence class is close to the distribution of the attribute in the overall table (i.e., the distance between the two distributions should not be greater than a threshold t). In other words, an equivalence class is said to have t-closeness if the distance between the distribution of a sensitive attribute in this class and the distribution of the attribute in the whole table is no more than a threshold t. A table is said to have t-closeness if all equivalence classes have t-closeness.

Differential Privacy is a process derived from cryptography. As defined in [56], it "aims to provide means to maximize the accuracy of queries from statistical databases while minimizing the chances of identifying its records." Unlike other methods, differential privacy operates off a solid mathematical foundation, making it possible to provide strong theoretical guarantees on the privacy and utility of released data. The most used technique is called ε differential privacy and it is modelled via a randomized algorithm; a theoretical definition is given in [196] and summarized as follows.

"A randomized function K gives ε-differential privacy if for all data sets D and D'differing on at most one row, and all S $\subset$ Range(K),"

Pr[K(D) $\subset$ S] >= exp(ε) x Pr[K(D') $\subset$ S]"

This formula can be interpreted as stating that the risk to one's privacy should not substantially (as bounded by ε) increase as a result of participating in a statistical database. Namely, that an attacker should not be able to learn any information about any participant that they could not learn if the participant had opted out of the database. This goal is pursued by adding some noise to the result of a query on the dataset. There exist many different mathematic mechanisms to do that; the most commonly seen in this context is the Laplace mechanism, which adds noise derived from the Laplace distribution. It has only one parameter, defining the standard deviation, or noisiness. This parameter should have some dependence on the privacy parameter, ε; it should also depend on the nature of the query itself, and more specifically, the risk to the most different individual of having their private

information teased out of the data.

Differential privacy comes in many different forms and variations which we will not cover in detail, but they all have several limitations, due in particular to the high computational complexity that the cryptographic techniques could introduce in a big dataset. The main advantage of this approach is that its mathematical foundation makes it possible to actually measure the strength or the weakness of the results. The concept of differential privacy holds much potential, and is still the topic of active research.

4.2.2 Sanitization in the Clearing Scenario

To ascertain the suitability of the illustrated techniques to the clearing scenario, the first step is to define the correct evaluation criteria, which could depend from:

- The context of collection and usage of the data;
- The structure of the data;
- The objective of data processing.

As an example, the work of [23] measures the trade-off between privacy (i.e., how much the adversary can still learn from the sanitized records) and utility (i.e., the residual accuracy of data-mining algorithms executed on the sanitized records with respect to what could be found for legitimate purposes from the original data set). Their paper showed that k-anonymity provides no privacy improvement on the tested dataset; furthermore, l-diversity is no better than trivial anonymization.

Another interesting work [42] tries to quantify the effectiveness of sanitization in terms of privacy impact of a data release. To this end, the study introduces the idea of incorporating a metric over 'privacy breaches' based on a notion of empirical privacy, and evaluating the corresponding empirical utility of the released data.

The measure of a privacy breach is defined as the increase in correct a-posteriori inferences obtained by an adversary about sensitive values in the data, using a Bayesian classifier with previous knowledge. The cited paper applies this metric to the main four techniques, and concludes that differential privacy often provides the best empirical privacy for a baseline utility level, but that for increasing utility

levels it can be preferable to adopt methods like t-closeness or l-diversity. There are other works that pursue the same kind of investigation, that mainly derive as a conclusion the weakness of k-anonymity and l-diversity algorithms.

The main limitation of the reviewed literature is that only few papers interact with large amounts of data derived from public transport system. An exception is the work in [70], which aims at preventing privacy attacks in a general sense, especially those damaging from a user's perspective. The proposed solution is an algorithm based on the LK-privacy model, using the approach of "identifying the LK-privacy requirement, and then eliminating the violating sequences by a sequences of suppressions with the goal of minimizing the impact on the structure of the user tracking." What the authors claim is that their anonymization algorithm thwarts identity record linkages, while effectively preserving the information quality in terms of its suitability for the generation of a probabilistic flow-graph. It is a very interesting result, yet insufficient in the scenario of a clearing system, where the user's privacy perspective is not the only one that must be protected; in fact, the insider threat is not taken into account.

Eventually, with the exception of differential privacy (which cannot be easily applied to huge amounts of data), it would seem that not a single sanitization solution is really effective. Actually, these studies demonstrate only how these techniques are not effective enough for the particular context taken into consideration. To properly evaluate their potential in our scenario, a more precise definition is needed for various characteristics, namely:

- the privacy requirements;
- the expected level of utility of datasets;
- how to measure the effectiveness of algorithms at preserving these properties.

As regards privacy requirements and utility levels, it is possible to reason in our case study, on the structure of sensitive values and quasi-identifier. The sensitive values are the user's identity and location data; these values are the one to hide and protect. The means of transportation and the user's contract data, otherwise, can be classified as quasi-identifier values, since they could become sensitive if crossed

with other information; at the same time, these are the data needed to calculate the clearing functions, so they cannot be depleted because of the precise information they carry. The goal of the anonymization process is to break the link between user identity and location, and to mask the QI values in a way that preserves the values needed for the clearing system.

As regards the metric used to quantify privacy, the best work done in this path is [144] were authors created a metric called δ-presence to evaluate the risk of identifying an individual in a table based on generalization of publicly known data. This work shows that existing anonymization techniques are inappropriate for situations where δ-presence is a good metric (specifically, where knowing that an individual is in the database, as it very often happens to travellers of public transportation networks).

So, despite its quality in general terms, this metric cannot be used in our context. Otherwise, the metric discussed in [67] is defined as the amount of "useful" data mining queries still existing after the sanitization phase. As shown in the following section, an insider threat attack in our context is likely to take the form of a search pattern analysis; for this reason this measure of privacy seems to be more interesting.

4.3 Security Analysis

In our scenario, the clearing datasets could be exploited by any of attacker categories wishing to infer information regarding the operators' business and/or the passengers of the public transportation system. Moreover, there is a specific insider threat: the participating companies may try to use data analysis to gain competitive advantages, both of the kind usually associated with market analysis (e.g. uncommonly profitable routes) and of the kind usually regarded as a trade secret (e.g. the optimization of the allocation of resources such as buses, trains, and personnel on board).

The main issue is that when pursuing their goals, adversaries are not limited to the analysis of the clearing data alone. Conversely, they can reap great benefits through correlation with many existing public databases. The first widely known case of identification through correlation of different public datasets dates back to

2006, when AOL released anonymous data about search queries and New York Times reporters were able to find the real user linked to a specific hash [13].

As noted by [19] this case also shows a peculiar effect of the failure of the privacy protection: since the user acted as a proxy for friends with no Internet access, her name was associated to many queries unrelated to her condition and habits. The same could happen with public transportation data. As an example, if zones of boarding and alighting are kept wide enough to conceal the exact location of a passenger, they could end up enclosing points of interest (hospitals, schools, recreational facilities, shopping districts, etc.) which could lead an attacker to draw wrong conclusions, possibly even more damaging to the victim than the correct ones.

A final example of the risks associated with location data is carried by the study of possible privacy breaches as a consequence of the traces left when renting bikes in London. This is a very relevant case for this work, since the kind of data used in the studies described hereinafter is strikingly similar to what could be found in our datasets. Here [176] authors analysed a publicly available Transport for London dataset that contained records of bike journeys for London's bicycle hire scheme over a period of six months between 2012 and 2013, reaching the conclusion that "with a little effort, it's possible to find the actual people who have made the journeys".

4.3.1 Specific attack scenarios in our context and countermeasures

The kind of correlation with external databases exemplified in the previous section is feasible in our context too. Not only it is possible to leverage online social networks in the same way, but also to browse many free-access databases of sensitive data, strongly related to the regional context. Just to give two examples, it is very easy to extract from the corresponding web sites all the professional data information (such as office address, office telephone number etc.) of the regional health organization staff, as well as of the university staff in all the cities that have an academic institution. These data are not sensitive if taken alone; in fact, the transparency laws of the public administrations mandate their availability; as shown in the next section, it is their combination with the public transport dataset that could allow privacy breaches.

If this were not enough, as explained previously the same sanitization algorithm are not free from attacks. In particular when there is the need to keep a good level of utility, algorithms as k-anonymity have been proven to be weak against attack where the adversary have a previous ("a-priori") knowledge. The work of [116] and the l-diversity algorithm have been created to overcome these de-anonymization issues of k-anonymity. As well explained in [70], anonymizing public transport data structured over a space with a high number of dimensions has been studied widely, but in general none of the proposed solutions takes into account the clearing scenario with its peculiar requisites about utility. In [70] the differences between the different methods are clearly detailed.

[71] propose a permutation method that groups transactions with close proximity and then associates each group to a set of mixed sensitive values. [184] propose an algorithm to k-anonymize transactions by generalization based on some given taxonomy trees. [84] extend this method by introducing local generalization, which awards better quality. [199] extend the k-anonymity model by assuming that an adversary knows at most a certain number of transaction items of a target victim, which is similar to our assumption of limited background knowledge of an adversary.

This is a very interesting model because it is definitely related to our scenario. It deals with attempts to gain a basic knowledge of some transaction rows, which is equivalent to get a certain number of possible travel records of a user: a valuable outcome for an attacker in our context.

Yet, although their method addresses the high-dimensionality concern, it considers a transaction as a set of items rather than a sequence; this makes it useful to prevent attacks against the privacy of single users, but not to prevent attempts at general pattern discovery, which is typical of insider threat attacks. Therefore, it is not fully applicable to our problem, which needs to take into consideration the sequential ordering of travel data. Furthermore, [199] achieve their privacy model by merely global suppression, which significantly decrease information quality on transport data.

The last reviewed model was developed by [38]. It studies the releasing of transport dataset while satisfying differential privacy techniques. Although they claim that their approach maintains high quality and scalability in the context of set-valued

data and is applicable to the relational data, their method is limited to preserving information for supporting count queries and frequent item-sets, as opposed to [199], and not passenger tracking. The combination of these two pieces of research is a very promising research direction towards a complete solution for our scenario.

4.4 Case Studies

This section presents some simple case studies built along the lines of the illustrated threats, showing how such concepts may easily be applied to the case under consideration. Two different threats are considered:

- an attacker who tracks the movement of a specific person (one of the authors) on the public transport network, exemplifying a threat to the privacy of individuals;
- an attacker who is interested in understanding what are the more profitable areas in terms of regional tickets sold, to challenge the business of an operator, exemplifying a threat to trade secrets of operators.

A summary of the data items collected by operators for each ticket validation is described in Table 4.1. The definition of the minimal subset needed for clearing, and the anonymization of the selected fields, are the goal of the work in progress described in this section. However, it is immediately possible to notice that the most sensitive attribute and the only direct identifier (in case of personal passes), i.e. the serial number of the ticket, cannot be omitted.

Following its usage through the dataset (possibly over a period of time that cannot be known a priori) is the main function of the clearing system, which has to compute the share of revenue (generated when the ticket was bought) to distribute to each carrier which provided service to the ticket holder. In a broader sense, it is possible to define the required utility level of the dataset as being very similar to the goal of a potential attacker: that is, allowing to reconstruct a traveller's itinerary. It is worth detailing how this reconstruction happens, to understand also how an attacker could try the same process and follow a traveller.

TABLE 4.1: Dataset Item Summary

Field name	Content
CONTRACT SUPPORT	Type of physical token
CONTRACT TYPE	Type of contract (single trip, pass, etc.)
VALIDATION TSP	Timestamp of ticket usage
VALIDATOR LOCATION	Placement of the validating equipment
CONTRACT RESELLER	Company which sold the ticket
VALIDATION LINE	Bus/tram/train line number
VALIDATION NR	Number of parallel validations of the same ticket (e.g. many passengers on a single pay-as-you-go ticket)
VALIDATOR SERIAL	Serial number of validating equipment
CONTRACT SN	Serial number of the ticket
VALIDATOR MODEL	Model of the validating equipment
VALIDATION ZONE	Fare zone where the ticket was validated
CONTRACT VALIDITY	Geographical extension of the contract (regional, urban, etc.)
CONTRACT ZONES	Number of fare zones the contract allows to traverse

The itinerary is "blurred" within the dataset, because in the studied system passengers validate tickets only when boarding a bus/train, but not when leaving it. Consequently, raw data does not show a sequential structure; each row represents a leg of a trip for a contract, but there is no direct connection with the next legs of the same contract. Each leg can be represented by a structure like (A)T1 $\rightarrow$ (B..Z),T2 meaning that on time-stamp T1 a user (Contract_SN) goes from A in a known direction (inferred from Validation_Line) leading to a set of possible stops, one of which is reached at T2. This introduces uncertainty in the computation of the number of traversed zones, which is needed by the clearing system when a vehicle of a different operator is used to continue the trip: in this case, the end of the first leg must be inferred from the validation that happens at the start of the second one.

To this end a probabilistic flow-graph can be exploited. According to [70] a probabilistic flow-graph is a tree where each node represents a point in space-time, the edges corresponds to transitions between two places, leaving the origin at a given time to reach the destination at a different time, and each transition has an associated probability of being actually followed. For every node, there may also be a non-zero termination probability, which is the percentage of passengers who exit the transportation system at the node. By looking for validations of the same contract that are consecutive within a given time-frame, a possible itinerary can be identified.

For example, if a validation (A)T1 could take a passenger to (B..Z), and there is a validation (D)T2, with the value of (T2-T1) falling within a given threshold, a non-zero probability can be associated to the edge (A)T1 $\rightarrow$ (D)T2. The analysis can proceed seeking for destinations that can be reached from (D).

Table 4.2 and Figure 4.1 depict an example of the probabilistic flow-graph derived from one of our datasets for a few contracts. With enough samples, probabilities can be estimated with acceptable precision, and the graph becomes a faithful enough representation of the distribution of passenger over the network. At the same time, each set of itineraries for a given contract represents the habits of a passenger, enabling correlations with other data sources (places around the nodes, events close to the timestamps), and potentially leading to the association between the contract and a personal identity.

TABLE 4.2: Probabilistic flow path of our dataset

Serial	Content
70001112	(1, 245)T1->(2,249)T2->(3,248)T3->(1,245)T4
0004058	(1, 245)T3->(1, 245)T4-(1, 245)T6
50004077	(4,260)T1->(2,249)T2->(1, 245)T5
50004070	(1, 245)T1->(2,249)T2
70001386	(1, 245)T3->(2,249)T4
70001389	(1, 245)T1->(2,249)T2->(1, 245)T4

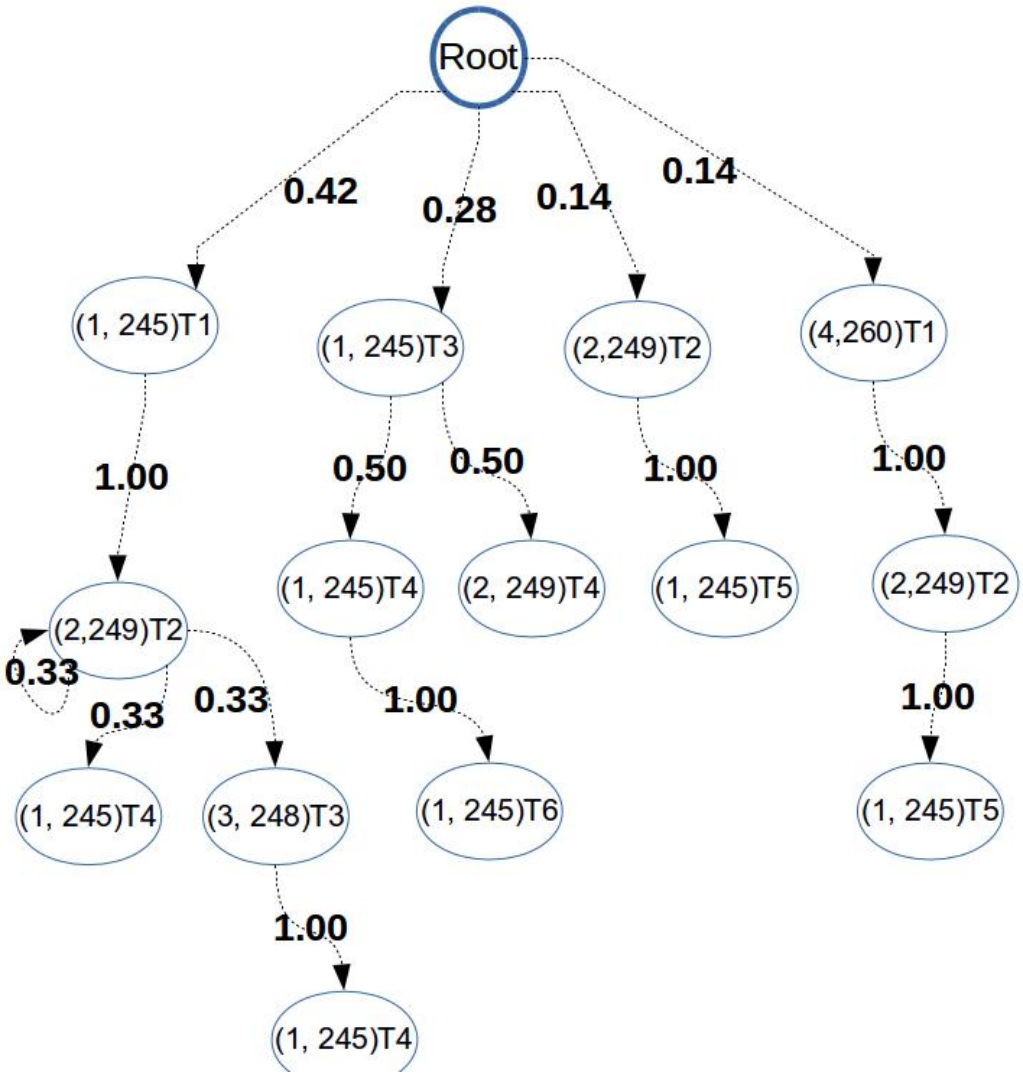


FIGURE 4.1: Travel data represented as a probabilistic flow-graph

4.4.1 Stalking Franco Callegati

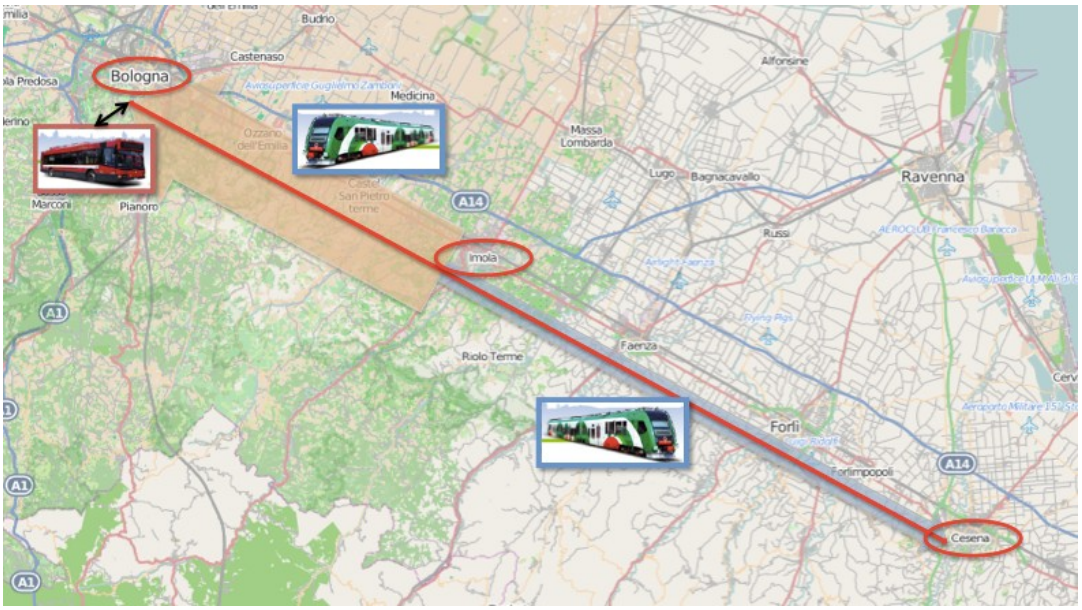


FIGURE 4.2: Commuting flows from Imola to Bologna and Cesena

As a proof of concept, let us present the case of an attacker who wants to target one of the authors, to track his movement on the public transport network. Franco Callegati is a professor at the University of Bologna. He has a public web page detailing the address of his office which is located in the off-site campus of Cesena, about 60 Km East of Bologna, and showing that he works at the Department of Electrical, Electronic and Information Engineering, which has its central offices in Bologna. The phone directory lists his home address in Imola, a smaller town about 30 Km east of Bologna.

Clearly from this basic data it can be inferred that Callegati will mostly travel to work from Imola to Cesena where he teaches and tutor students, but he will likely travel to Bologna too, for those sort of activities requiring physical presence related to the Department or to the University's central offices. Sometimes he will also travel from Cesena to Bologna (or the other way) when he has some commitment in both sites in the same day.

With this background, an attacker who has got a copy of the clearing dataset can associate the victim's identity with the serial number of his MiMuovo pass. Callegati is admittedly a very easy target, yet he serves us well for the purpose of giving

a concrete and real example of usage of clearing datasets. Given the almost non-existent effort that allows a potential attacker to reach his goal, there is little doubt that "harder" targets can be exploited with some more, but still reasonable effort; as already illustrated at the end of previous section, even a coarse localization of commuting start and end points, when correlated with some background information about the victim, can yield significant results.

The dataset shows about 2,000 passengers boarding trains that leave Imola in the morning (all figures are computed as averages on working days). Since the validation occurs only at the start of the trip, their destination is not explicit, but of course it can be inferred with a good approximation by coupling the onward trip of a given ticket with the return trip. This further analysis yields slightly more than 1,000 passengers getting back from Bologna in the afternoon (Fig. 4.2, pink arrow) and slightly less than 200 passengers getting back from Cesena (Fig. 4.2, grey arrow). The number of candidates drops dramatically when only passengers who travel alternatively to both Bologna and Cesena in different days of the week are considered. Only 14 MiMuovo users show a commuting pattern of this kind (Fig. 4.2, red arrow).

The possible inferences do not stop here. It is easy to check that the Callegati's office in Cesena is near enough to the train station (15 minutes on foot), and that it is not conveniently served by public transport (direct bus only once an hour). An attacker could make an educated guess that Callegati will not take a bus when he leaves Cesena's train station, and thus eliminate candidates who do it. Conversely, the site of Callegati's department in Bologna is twice as far from Bologna's train station, compared to the Cesena situation, and much better connected to it by bus (6 to 8 connections per hour).

In this case an educated guess would lead an attacker to consider the exclusion of candidates who do not board a bus in Bologna after reaching the train station. Note also that the clearing function can be computed without taking into account the line number, but in case the full database is leaked, or in case the line number is kept on record for secondary uses, it would be possible to further restrict the set of candidates to those boarding one of the two bus lines connecting the train station with the department, out of the 19 serving the train station.

In our tests, this is enough to pinpoint the victim. This result was reached without even taking into account another very valuable source of information, the timetable of the lectures in Cesena, which would allow establishing a precise spatio-temporal constraint to the victim's movements towards one of his usual destinations. In conclusion, by following these patterns, an attacker can identify Callegati's MiMuovo card ID and then follow his movements also outside his most common habits by accessing the clearing database.

4.4.2 Unfair competition

FIGURE 4.3: Results of a clustering analysis of the dataset with Weka

Ticket validation datasets contain potentially useful information for an operator wishing to uncover the business practices of its competitors or challenge their business practice. This kind of attack comes from the inside, and it is very difficult to deal with. Access control rules cannot be very strict against insiders, who enjoy not only the possibility of easier read-only access to datasets, but also the opportunity to inject carefully crafted data to stimulate the production of particularly useful outputs, like a cryptanalyst that is able to perform a chosen plaintext attack. One of the most valuable pieces of information would be the planning strategy in the usage of vehicles, which is a crucial issue for a transport provider and that can be inferred at

some extent by exploiting the information in VALIDATION ZONE, VALIDATION LOCATION, and VALIDATION LINE.

Here a simple and realistic inference is shown, built by looking at the correlation between the type of ticket and the zone of its usage. It is a piece of information that can give a very small margin of profit by pushing sales of multi-trip tickets where they are most appreciated, making profits on the rate of unclaimed trips for lost tickets (what is not claimed for clearing remains in the pockets of the seller). This should clearly be a small percentage of the whole ticket volume. Nonetheless in today's competitive markets every source of income may be vital; moreover the examples show that this sort of analysis may pave the road to similar analyses in "business areas" which are not considered today, because they are impossible to accurately explore in absence of large datasets.

We performed Over 30 data mining tests over the ticket validation datasets using the Weka software [82]. The correlation of interest was best highlighted by means of cluster analysis, i.e. a set of exploratory techniques that aim to group the unity of a population in statistics on the basis of their similarity in terms of values taken by the observed variables. As an example, Figure 4.3 shows the result of cluster analysis according to the Simple K-means_1 classifier. It is clear that the attributes of the sold tickets form well-separated clusters, whose significance can be useful from a business perspective. Once this hypothesis is verified, a Bayesian_2 classifier allows to infer more details over some attributes. The structure of the clearing system allows an attacker to feed the Bayesian classifier a large amount of past knowledge from the snapshots. The result is that the algorithm is able to correctly predict the belonging to a given cluster of over 90% of new instances.

This result needs to be interpreted in a specific context in order to show the power of this kind of attacks. The Bayesian test shows that theoretically, by knowing only the contract type and the validation zone, it is possible to infer the correct serial number of the contract support, which would reveal the pseudo-identity of the contract holder. Beneath the privacy risks for the contract holder, this discovery would allow a company to determine the history of a pseudo-identity. This history would be revealing the type of contract, along with its movements, leading to a kind of profiling and possible definition of targeted offers that is usually regarded as unfair

competition in our context.

A more general attack is also possible, again by using the results from cluster tests. Cluster analysis usually aims to group the elements of a population on the basis of their similarity, in terms of values found in the observed variables. However, if the focus is put on a particular attribute, for example the contract type, it becomes possible to trace the trend in terms of other variables, for example (see again Figure 4.3) how the contract is used in a group of specific zones. This could easily lead an operator to discover the contract distribution of a competitor. So by intercepting this market trend, once again, the opportunity may arise to engage strategies deemed as unfair competition.

4.5 Mitigation Policy Guidelines

In order to take the described factors into account when designing the clearing system, we propose to undertake an iterative approach described by the following steps:

- **Data minimization** - take into account the needs of the legitimate analysers to define the smallest set of attributes that allow performing the intended computations;
- **Sanitization techniques** - choose the perturbation and generalization algorithms that provide the most effective concealment of sensitive data without jeopardizing its utility for legitimate uses;
- **Verification** - evaluate the results to formally verify that the privacy policies are respected and that the resulting dataset actually preserves its intended utility. If some basic constraint is found violated, or margins for further improvement are visible, the cycle is reiterated to achieve the foreseen refinement, otherwise the process stops. Note that this step could highlight an intrinsic contradiction between some of the privacy policies and some of the analysis requirements, leading either to the decision to relax some requirement or to the conclusion that the desired scenario is unachievable.

The last step of the iterative process calls for a formal metric to evaluate correctness and effectiveness. The literature already provides useful methodologies for this purpose. Here authors [67] studied the general problem of fusion resilient anonymization. They stated the problem as a search for the optimal value of an objective which is the weighted sum of protection and utility. Protection is measured in terms of how hard is for an adversary to gain information by correlating the anonymized dataset with external sources. More formally, if P is the original sensitive dataset, and P I is the sanitized release of P, an adversary can exploit information fusion techniques to derive a de-sanitized version P'from P I. The effectiveness of the applied sanitization process is measured as the dissimilarity between P'and P .

Brickell and Shmatikov [23] performed a similar analysis, again studying the trade-off between utility and protection, but on a more formal level. Their claim was quite thought-provoking: sophisticated anonymization techniques offer no real advantages over trivial ones, i.e. datasets sanitized with complex application of generalization and perturbation algorithms, depending on the algorithm parameters, either provide no additional utility vs. trivially-sanitized datasets, or leak much more information to adversaries than what is gained in terms of legitimate analysis. Besides posing interesting questions that researchers in this field could find useful to orientate their efforts, their paper also provides formal definitions for various metrics related to privacy of data tables and utility of sanitized databases. In particular, they study privacy both under a syntactic perspective (pure statistical correlation) and under a semantic perspective, measuring the gain in adversarial knowledge afforded by the sanitized table.

Bishop et al. [19] explore the topic by focusing on relationships that can be used to desanitize sensitive data. They model the problem of data sanitization as a double set of assertions, made of the constraints defining the privacy properties that must hold against adversarial attacks, and of the targets defining the information that the legitimate analysts want to extract from the sanitized dataset. They highlight the importance of defining a precise threat model as a requisite for drawing complete and concrete privacy policies, and a precise analysis policy that, in opposition to the privacy policy, puts limits to the sanitization process to avoid excessive loss of utility. They exploit ontologies to automate reasoning over these opposing requisites

and solve the constraint satisfaction problem they represent. A question they leave open is: what is the most appropriate language to express these requirements?

To define our specific privacy by design process, we plan to investigate and possibly apply the common concepts and techniques presented in all of these works, in addition to a specific and noteworthy suggestion that comes from the last one. In the words of the authors: “Perhaps the most constructive approach is to provide two sets of relationships. The first lists those relationships that are known to hold in the raw data, and must not hold if desanitization is to be prevented. The second is a set of relationships that, if they held, would enable desanitization. The sanitizer can deal with the first set as appropriate. The second enables the sanitizer to perform a simple risk analysis, centred on two questions: (1) What is the probability that the relationships in this set hold; (2) What is the probability that the adversary will be able to determine that the relationships hold, and use that to desanitize the data?”. This approach seems to be especially useful because it allows both designing sanitization by evaluating the effectiveness of the planned techniques, and to measure and understand the risks deriving from future evolution of the intended use of the datasets.

4.6 Conclusions

This chapter of the dissertation highlights the privacy threats that can emerge from sharing or publishing the data related to usage of public mobility tickets. In the context of an integrated mobility system run by a set of operators, sharing data about tickets usage become mandatory for revenue clearing purposes. Unfortunately this may also pave the road to privacy attacks to individuals or institutions, such as, but not limited to those exemplified in this thesis.

A discussion is presented of how the sanitization approaches could work in this scenario, and what their limitations are; it lays the ground for future research aimed both at improving the effectiveness of sanitization techniques in the specific scenario, and to derive generally-applicable principles.

As an alternative solution, the applicability of a set of principles derived from the "privacy by design" approach is examined. In the context of the design of the

data sharing system for clearing purposes, its aim is to minimize the likelihood of the emergence of privacy threats. The general-purpose definition of the approach is integrated with an implementation plan that takes into account a variety of literature sources to verify the effectiveness of the applied methodology, in order to iteratively converge towards the solution that strikes the most appropriate balance between data utility and privacy, possibly quantifying the effects of a breach.

The first set of research questions, regarding the suitability of existing techniques to protect privacy in the public transport scenario, has thus being answered in a substantially negative way.

The second wave of research questions, on the possible definition of a more effective framework to devise privacy-enhanced data management processes, has been partially addressed. The present study meets its main limitations here: the negative findings from the review phase, and the realization that similar initiatives actually deal with less demanding requisites, were useful in highlighting the deficiencies of current approaches and lead to devise suggestions on how the framework could be structured, yet its concrete development will be the subject of future work.

Chapter 5

SMAll: A Global Federated Market for MaaS Operators

Multi-modal travelling is a common phenomenon. Commuters, tourists, and travelling workers are used to compose trips out of legs covered with disparate means: bike, car (personal, rented, hailed, or shared), bus, train, boats, and planes. Usually, each ”hop” requires interaction with a different operator and, mostly, with a different information system, since mobility resources are administrated and owned by a scattered plethora of mobility operators. As noted in [151], this is due to regulatory and logistic constraints that favour site-specific solutions. Hence, the experience of multi-modal travelling results often into a discontinuous flow of interaction, scattered over many applications, having a negative effect on both mobility providers and customers. The former suffer *opportunity costs* due to the loss of potential clients that could not find or know about their services. On the other side, the customers undergo many inconveniences, culminating in a sensible waste of time. Reasons comprise:

- *uneven experience*: customers may have to plan their trips over separate systems and different media with inconsistent interfaces and flow of interaction (e.g., calling a taxi via phone and then continuing the trip on train, whose ticket was booked online);
- *access issues*: although multi-modal planning services have become freely available (e.g., mapping services provided by Google and Microsoft), customers still need to find out what provides information on their trips. They have to

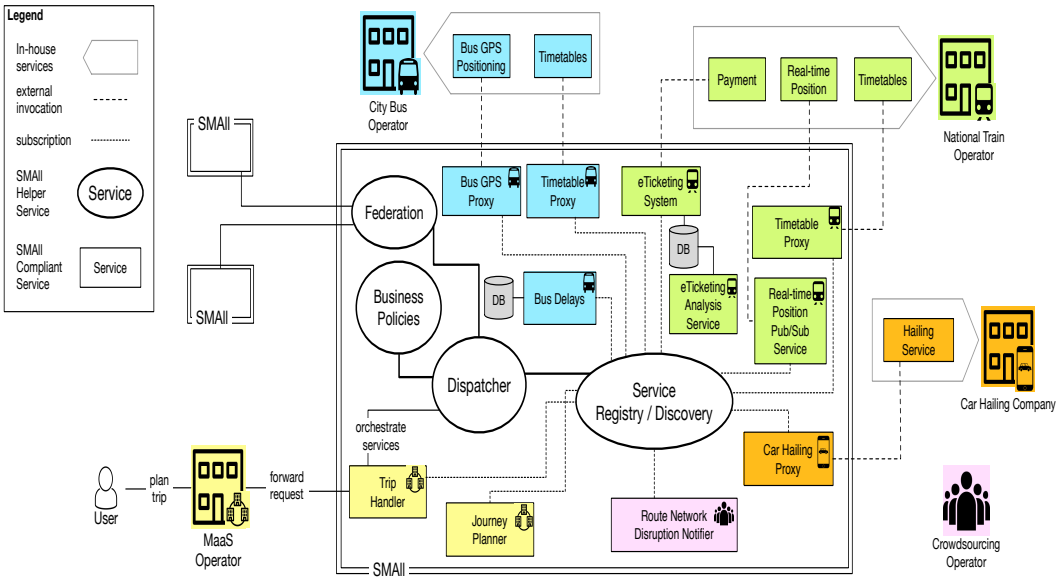


FIGURE 5.1: Representation of the SMALL architecture.

look for places, phone numbers, and dedicated applications or websites to retrieve information and book the trips.

- *interaction issues*: once customers found what means provide information on their trips, they have to negotiate how to get those information, dealing with multiple authentication systems, extracting data from different representations, and aggregating them to obtain a comprehensive plan of the whole travel. Each step increases the risk of introducing inconsistencies or missing key pieces of information.

In addition to the time lost by customers and missed opportunities of mobility providers, there are strong concerns regarding *data replication and security*. When users interact with multiple, separate systems within the same travel (i.e., to plan/book one of its legs), they are likely to replicate information that is already present in another system. However, since they have to manually replicate such data (e.g., their IDs, dates of the trip, or personal needs), they could introduce discrepancies among legs. As an example, consider the steps from a multi-modal travel-planning application to the actual booking of the travel. Although the travel-planner holds information on each leg of the travel, the customer has to manually replicate a subset of such data when she books each leg, managed by a different operator. Also the security of the data of the travellers plays an important role [29]. Customers have to provide their

personal data to systems that guarantee uneven security measures, making difficult or even impossible to assess the level of security of the whole process.

Mobility as a Service a possible solution to the issues illustrated above is the creation of a unifying framework for mobility that supports the coordination of different transportation systems. Such an idea has been envisioned and described in several works in the last decade [120, 121, 151, 73]. Recently, this idea has materialised in some practical applications inspired by the concept of Mobility as a Service (MaaS) [155]

Analogously to the case of Cloud Computing (conveyor of the everything-as-a-service paradigm), MaaS hides a dynamic infrastructure of different travel agencies into a consistent interface: this makes MaaS users experience travelling as provided by a single agency. Ideally, a MaaS provider, also called MaaS operator, shall provision its users with information and procedures for discovering, planning, booking, and guiding journeys, combining any variety of means of transportation. To the final user, the provisioning of *mobility resources* (i.e., information on transportation and the actual transits) is transparent wrt the actual provider of the service. Since mobility resources are administrated and owned by disparate mobility providers, we argue that the leading economic model of MaaS markets is that of federations of providers, each trading its mobility resources. In such a federated market, MaaS operators dynamically partner with each other whilst preserving their individual autonomy and without a centralised regulation authority which, in the case of transportation, would be practically impossible to appoint.

Contributions in this chapter we present several contributions. First, we introduce the *MaaS Stack* (section 5.2): this is the first tiered view that provides a structure for the, so far, informal concept of MaaS. The MaaS Stack originates from our discussions with companies interested in entering the market of Mobility as a Service, as well as from investigations conducted within the EU EIT Digital project SMAll[1]. We deem our view useful to isolate and clarify which elements must be in place for mobility operators to join the MaaS market, possibly becoming themselves MaaS operators.

[1]Project description: `https://goo.gl/WKnnSW`

The second contribution is the presentation of a microservice-oriented platform called *Smart Mobility for All* (SMAll)[2,3] (5.3), developed to support the creation of a federation-based MaaS market and which is structured according to the principles of the MaaS Stack. SMAll facilitates the publication, automatic retrieval, and orchestration of functionalities for mobility, provided by different mobility operators.

The platform builds on the concept of Federated Cloud Computing [160, 25] and maintains an open approach wrt the possible members of the federation: MaaS operators, traditional transport agencies, and other players that trade information linked to mobility, like weather forecasts or crowd-sourcing communities [90].

SMAll is developed following the microservices paradigm [54] and members are strongly supported in publishing their functionalities as microservices. As remarked in section 5.3.1, such architectural choice positively impacts on both SMAll and the resources deployed by members, which enjoy great flexibility, gradual deployment and continuous integration, eased software maintenance and, most important, efficient scalability according to dynamic demands.

Next, using the SMAll platform we will propose two real world use case scenario of SMAll applications. A Federation of Platooning Operators for smart truck mobility, and Cloud-of-Things interpretation of Smart Mobility.

Another important contribution will be an extensive security analysis of the main security issues in such architecture. We will explain why we mainly focused on insider threat attacks and in this part the contributions will be of two types. We will first produce a taxonomy concerning a general architecture approach, based on the previously predefined MaaS stack, and we will also show a more specific analysis on the two use cases implemented.

As a consequence, lastly we will show the proposed solutions for this scenarios, consisting in a overlay network for data management and a business policy enforcement.

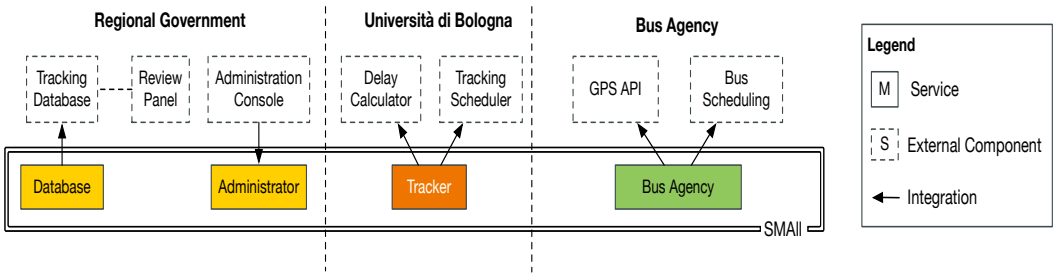


FIGURE 5.2: Representation of the BusCheck Pilot.

5.1 Context Scenario

In this section, we overview the concept of MaaS and illustrate the main features of SMAll with a representative instantiation of our platform, depicted in Figure 5.1. In the remainder of this dissertation, we use the term *service* to indicate an application deployed within or outside SMAll, while we use the term *microservice* to indicate an instance of a service within SMAll[4]. In Figure 5.1, the coloured entities outside of the boundaries of SMAll (bordered with double lines) are providers of mobility resources and MaaS operators. These are public transportation agencies, private companies, and online communities. By entering the platform, each member can be dynamically federated with the other agents already present. Once in SMAll, members can deploy their own functionalities as microservices, e.g., in Figure 5.1, the City Bus operator deploys three services: the first two are Bus GPS Proxy and Timetable Proxy, which function as wrappers for some pre-existing applications deployed in-house by the operator. The third service, Bus Delays, is completely contained in the platform and orchestrates the other two services of the Bus Operator to calculate the delays of buses by comparing the actual GPS position of the rides with the expected scheduling from the timetables.

SMAll provides helpers (e.g., Registry / Discovery and Dispatcher) to publish, discover, compose, and regulate the usage of the deployed microservices. Considering the example above, although all the microservices belong to the Bus Operator, all the invocations from the Bus Delays service are routed and managed by the Dispatcher. The Dispatcher also enforces the usage Business Policies defined by the

[2]Wiki: https://github.com/small-dev/SMAll.Wiki/wiki
[3]Deployable platform: https://hub.docker.com/u/smallproject/
[4]Hence, to a service correspond one or more copies of the same microservice that implements its functionalities.

owner of the invoked service, e.g., it can refuse to proxy the invocation to the addressee as well as to delay frequent requests if they exceed the rates established by the owner.

Business Policies are fundamental for marketing on-demand services. As an example, consider the case in which the City Bus Operator integrates the crowd-sourced data on Route Network Disruption in its Bus Delays service (e.g., to forecast day-long delays on the interested routes). With SMAll, using such crowd-sourced data does not require the presence of pre-existing contracts of usage between the Bus Operator and the Crowdsourcing Operator. By accepting the business policies formalised by the Crowdsourcing Operator, the Bus Operator can dynamically access (and pay for) the Disruption information.

Finally, different SMAll installations can be federated as well, so that region-wide instances can constitute a federation of international- and world-wide platforms. Consider, for example, a MaaS operator that wants to provide transportation solutions to its users travelling abroad. It would be unthinkable for the MaaS operator to foresee and stipulate contracts of usage in advance with all the possible foreign transport agencies. On the contrary, with SMAll a MaaS operator can automatise the dynamic aggregation of foreign federated services for its users, letting them access transport solutions of other operators. Moreover, like the other members of the platform, also MaaS operators can deploy services.

For example, in Figure 5.1 the MaaS Operator deploys a Journey Planner and a Trip Handler. The latter, in particular, is the service that orchestrates the dynamic multi-modal trips for the users of the MaaS Operator. To do that, Trip Handler interacts with the Dispatcher to reach and orchestrate the other federated services: it uses information on scheduling, availability, disruptions, and the position of buses, trains, and on-demand cars to dynamically plan a multi-modal trip, booking and paying the rides for the user.

5.1.1 SMAll: Smart Mobility for All

In this section, we illustrate the proposed approach with a real-world use case developed within the recent EU EIT Digital project SMAll. As part of the project, we

investigated the suitability — in terms of development, interoperability, and scalability — of SMAll wrt the creation of new smart mobility applications, possibly integrating pre-existing services.

As a real-world example, we report one of such applications, which we implemented as a pilot, called BusCheck. The pilot, commissioned by the Department of Transportation of the government of the Emilia-Romagna (ER) region (Italy), aims at recoding and displaying the quality of service of the buses in the Bologna province. Figure 5.2 represents the architectural view of our solution, composed of interacting services (continuous boxes) and external functionalities (dotted boxes) owned and provided by three organisations: the ER regional government, the University of Bologna, and TPER, the bus agency of the Bologna area. In Figure 5.2, we omit Registry, Discovery, and Dispatcher helpers that enable interaction among the deployed services (explained in section 5.3.1).

As shown in Figure 5.2, BusCheck emerges from the composition of four services deployed within SMAll (double-line in Figure 5.2): *i*) *Administrator* is a service owned by the ER government and used by operators to schedule and issue the tracking of buses. Operators interact with the service through a in-house client GUI outside SMAll. *ii*) *Tracker* is a service developed and maintained by the University of Bologna. It implements the actual logic to track buses. The service relies on two sub-services: a Tracking Scheduler that, based on the timetables of the tracked bus, triggers the retrieval of its real-time position; a Delay Calculator that computes the divergence between the expected and actual positions of the tracked bus. *iii*) *BusAgency* is a service maintained by the Bus Agency. It exposes to the Tracker the static and real-time data on all the vehicles of the bus agency. *iv*) *Database* is a service owned by the ER government that interacts with the Tracker, receiving data on delays and serving them internally to regional operators for both real-time and static inspection.

5.2 The MaaS Stack

For the clarification of the concept of MaaS, we deem useful to define it as a tiered structure, called the MaaS Stack. We use such partition in section 5.3 to analyse the

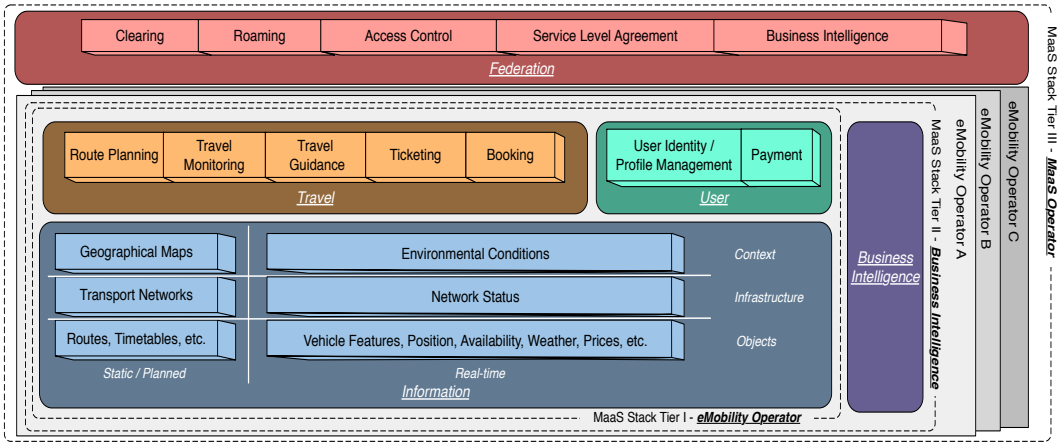


FIGURE 5.3: The MaaS Stack

elements of MaaS markets and the solutions we integrated in SMAll.

Figure 5.3 represents the MaaS Stack, comprising 3 tiers (dashed lines): *eMobility Operator*, *Business Intelligence*, and *MaaS Operator*. Inside the tiers, we identify 4 macro-layers of services (rounded rectangles), each building on top of the others: the *Information* layer contains basic services like timetables and real-time positions; *Travel* and *User* services build on Information ones to offer more advanced features to users; *Business Intelligence* services analyse data on performance and usages of the aforementioned services, providing insight to their owners; *Federation* services let operators trade their solutions and form dynamic partnerships.

In the next sections we analyse in depth the tiers of the MaaS Stack and the layers that characterise them.

5.2.1 Towards Mobility as a Service

Recently, traditional transport agencies have been publishing the data of their transportation solutions. Airline companies [139], train operators [143], and city-to-region-wide bus operators have been compelled to open the data regarding their transport systems to integrate with (de-facto) standards [76]. Beside traditional transport agencies, new companies entered the transportation market offering functionalities for mobility such as mapping, travel guidance, multi-modal journey planning and booking. The latests novelty on the transportation market scene are hailing companies like Uber [189] and Lyft [115]: "virtual" agencies that offer software functionalities to enable transport solutions. This scenario characterises the first tier of the

MaaS Stack where isolated entities, called *eMobility operators*, provide functionalities for mobility called *eMobility services*..

5.2.2 MaaS Stack Tier I | eMobility Operators

Definition 1 (eMobility Service)**.** *A software functionality for mobility, provided in a machine-readable form.*

For example, an eMobility service could give access to airline/train/bus schedules in machine-readable formats [76, 77, 35]. Support for machine-readable formats is key to enable the dynamic composition of services, so that information can be automatically processed, enriched, aggregated, etc., and exposed to other services in the same fashion. Figure 5.3 shows the layered taxonomy of eMobility services in the MaaS Stack. The mobility-specific services fall into two macro-categories: *Information* and *Travel* ones; then, there are what we define *User* services, such as user identity management, user preferences, payment circuits, etc., which are not specific to mobility. A listing of the fundamental services in these categories includes the following.

Information services allow users to access basic data needed for transportation purposes.

Static / Planned data is stable or seldom updated, regarding elements like maps, infrastructures (e.g., bus stops, parking, rails, docks), and timetables of transport services.

Real-time data report the status of the system. This ranges from unexpected infrastructural unavailability to current weather conditions, GPS position of vehicles, available seats/spots, traffic, delays, strikes, etc..

As depicted in Figure 5.3, Information services can be stacked as well. We illustrate the concept considering the *Static / Planned* information stack: the basic data on vehicles regarding e.g., *Routes* and *Timetables* provide a base for the static *Transport Network*. This holds also for *Geographical Maps* with respect to the transport network.

User services provide functionalities loosely related to transportation but necessary to interact with users.

- **User Identity / Profile Management** services allow to authenticate and authorise users, and also include functions to manage their preferences and historical records.
- **Payment** services handle transactions to pay transportation solutions and (possibly) the access to eMobility services.

Travel services mainly build upon Information ones.

- **Journey Planning** services find journeys, possibly multi-modal, between two points. They work on static data, possibly integrating real-time one for dynamic results.
- **Travel Guidance** assist the user with real-time travelling information wrt her position (e.g., turn-by-turn guidance).
- **Travel Monitoring** services track the position of the user and notify other subscribed applications of check-ins/-outs and other events related to her movements.
- **Booking** services use Information and Payment services to place a reservation for the user (e.g., seats, spots, etc.).
- **Ticketing** integrates Booking services to create and deliver transportation tickets (i.e., access tokens) to users.

In the first tier of the MaaS Stack, we consider only single eMobility operators (i.e., operators that do not use and integrate the services of other operators).

Definition 2 (eMobility Operator)**.** *An entity that owns, administrates, and exposes eMobility services.*

Intuitively, an eMobility operator is any entity (company, association, etc.) that publishes and orchestrates a set of eMobility services directly administrated by itself. An example of tier I eMobility operator is the National Train Operator in Figure 5.1: it exposes services to buy tickets online and to publish timetables and the real-time

position of vehicles in machine-readable standards. Our definition of eMobility operators comprises also providers of services not directly linked to transportation solutions, for example weather forecasts: indeed, they provide an important information on mobility and can enter the MaaS market as well.

5.2.3 MaaS Stack Tier II | Business Intelligence

The second tier of the MaaS Stack still focuses on single eMobility operators but it enriches the taxonomy of eMobility services with the category of *Business Intelligence* [37]. This category of services is separated from first-tier ones for two reasons: first they are not meant for users but rather for eMobility operators, and second, they span over all first-tier services by monitoring and analysing their usages. The aim of Business Intelligence services is to provide insight on the performances of eMobility services.

In Figure 5.1, an example of second-tier Business Intelligence service is the eTicketing Analysis Service. The service can access the data of the eTicketing System and, e.g., can suggest new pricing policies to the National Train Operator as well as reporting rarely used routes that could be merged/discarded. More generally, other examples of Business Intelligence services comprise reporting on the usage of the published eMobility services, analysis of the quality of transit systems (e.g., relating the discrepancies between scheduled trips and real-time delays), monitoring the profitability, sustainability, and reliability of the provided services and determining trends and making predictions on future usage, for capacity planning and policy definition.

5.2.4 MaaS Stack Tier III | MaaS Operators

As mentioned in section 5, the market of eMobilty operators is a very scattered one. The concept of Mobility as a Service originates from the economic opportunity of bridging the gaps between operators, both cutting down overhead for users and enabling synergistic strategies among transport providers. For the creation and success of such a MaaS market, it is imperative that eMobility operators can trade and

use said services on-demand. Such high degree of flexibility (and trust) is typical of *federations* [160, 25].

Definition 3 (MaaS Operator)**.** *An eMobility operator federated with other eMobility operators. A MaaS operator provides to its users eMobility and transit services of other operators as its own. The usage of such foreign services undergoes formal business policies.*

In the context of MaaS, when eMobility operators federate, they accept to adopt common technologies and formal *business policies*. Such technology standards and regulations are critical for a marketplace where eMobility and transportation services are traded like stocks, i.e., dynamically (not regulated by long-term, static contracts) and on-demand. Federated operators establish business policies to mechanise the trading of their services. The aim is to let users integrate eMobility services and transportation solutions of "foreign" operators into their travelling experience. The principle, already envisioned in [151], resembles that of *roaming* of GSM phone networks [141], where users connect through the services of another phone company when travelling outside the geographical coverage area of the home network.

Definition 4 (MaaS Roaming)**.** *Users of a MaaS operator can transparently use eMobility and transit services of other, federated operators.*

As an example, consider the MaaS Operator in Figure 5.1. Being federated with the National Train Operator and The City Bus Operator, it can offer multi-modal journeys that span different means of transportation (rail and road) and have wide-to-narrow scopes (inter-city and intra-city). For example, the MaaS Operator can leverage the available business policies so that its users can plan and purchase a trip (through its journey planner) associated with an eTicket bought from the National Train Operator which also comprises 5 trips of the City Bus Operator. The synergy benefits all partners: the MaaS Operator provides (and it is paid for) a comprehensive service to its users; the National Train Operator acquires users and can charge for the access to its eTicketing system; the City Bus Operator acquires new users that (probably) would otherwise have taken a taxi due to the overhead of looking for the right route and where to buy the needed tickets. Our example introduces the last

fundamental element of the third tier of the MaaS Stack: *Clearing* services, i.e., eMobility services that account for roaming usages and compensate operators according to the established business policies. This is of course the glue between the SMAll architecture and the previous use case scenario described in 4.1), a clearing system result of a vertical service enabled with the SMAll architecture.

5.3 The SMAll Architecture

5.3.1 A Market of Microservices

Following the lesson of Cloud Computing [26] (and the related SaaS/PaaS/IaaS stack), we argue that MaaS providers will require tools and infrastructures to harness the heterogeneous landscape of eMobility operators. We choose microservices [54] as the enabling technology for an on-demand marketplace after the observation that operators (e.g., the Bus Operator in Figure 5.1) already have a collection of legacy software systems that address some specific issues (e.g., the Bus GPS Positioning and the Timetables services) and that other operators (e.g., the National Train Operator in Figure 5.1) are willing to pay to access them. In practice, operators would like to include specific external functionalities (e.g., a bus tracking service) in their own services rather than use (and pay for) a bundle of unneeded functionalities (e.g., a real-time planner that includes the mentioned bus tracking capability). Microservice architectures achieve such degree of granularity.

In chapters 2 we already motivated the choice of microservice philosophy. Microservice architectures bring fundamental features for on-demand provisioning, among which: independent development cycles, per-usage resource allocation (limiting the allocation for unneeded bundled functionalities), and freedom to use task-specific technologies.

As expected, microservices come with some trade-off: loosely-coupled microservices communicate via message passing, which requires proper routing and may suffer latencies and failures; many requests can overload a service, hence it should prevent outages by limiting them and/or scaling accordingly; microservices are heterogeneous but their data-formats and Application Programming Interfaces (APIs) should be homogeneous to foster compositionality.

For these reasons, SMAll strives for standardization of data-formats and APIs. Moreover, it provides infrastructural tools to cope with most of the aforementioned issues: orchestration abstractions to streamline the composition of available services (via the Jolie programming language [98, 137]); data-format conversion functionalities; service registries and dispatchers to both store the definition and address of all the services deployed on the platform and to route requests to them; business intelligence outlets for auditing and performance indicators on the usage of services.

We now proceed to address the main characteristics of MaaS federated markets, describing the elements of SMAll that deal with them. In doing so, we follow the MaaS Stack from the bottom up. We first focus the needs of single eMobility operators within the first two tiers, broadening our view to MaaS operators in the third tier.

5.3.2 Tier I and II

Whilst tier I and II are stacked and their respective services differ from a user perspective (travellers for the first tier, operators for the second one), at the architectural level they share the same needs and components. Hence, in this section, we consider them together.

Concerning these two tiers, we fixed some basic requirements in the design of SMAll that we deem necessary within a platform for single-tenant microservice deployment:

- sandboxing [75, 157] for development and security;
- scaling both horizontally (i.e., create and remove copies of the same service) and vertically (i.e., increase and decrease the resources available to a microservice);
- publication and discovery of services and the related Application Programming Interfaces (APIs);
- orchestration of services.

Below, we detail the mentioned requirements and discuss the elements of the SMAll platform that address them.

Deployment — Sandboxing and Scaling virtualisation is the standard solution for cloud-based deployment of services [26]. The administrator of a service creates an isolated virtual machine, i.e., sandboxed, wrt the others and deploys her service on it. Then, single virtual machines can be scaled vertically and, by managing the number of copies of the same machine, horizontally.

However, virtual machines have several shortcomings: they entail important costs due to the need for dedicated resources, these resources could be wasted in idle cycles, and preparing and deploying full virtual images takes sensible time.

Hence, in SMAll we chose *containerisation* [127] as suitable solution that balances costs, time, and ease of development with a flexible and secure deployment. On these regards, SMAll can integrate techniques to optimise the deployment of microservices based on a description of the target configuration [66].

Beside deployment, microservices are mainly involved in orchestration, to which we dedicate the next two paragraphs. In the first, we describe why and how services should be indexed into a registry for discovery. In the second, we show how to support the orchestration of discovered services.

Orchestration — Registry and Discovery in cloud-based platforms like SMAll, the address of a microservice is dynamically determined at the time of the deployment and can even change during the life-cycle of the microservice (e.g., due to migrations). For this reason, SMAll does not provide the direct address of a deployed microservice.

Instead, following a pattern called registry-and-discovery — adopted by other cloud platforms [145, 147, 7, 9, 197] — when a programmer deploys a service in SMAll, she also registers it, with its description and APIs, into a Registry. As reply, the Registry returns a unique identifier of the microservice, which works as its reference address.

A registered service becomes visible to (allowed) users through the Discovery service, i.e., a SMAll helper dedicated to query the Registry to find services that match the requirements expressed by a user.

Indeed, SMAll supports users and access policies to let the owner of a service define which member can discover (and interact with) it. This is useful also within the borders of a single eMobility operator, where different departments deploy services

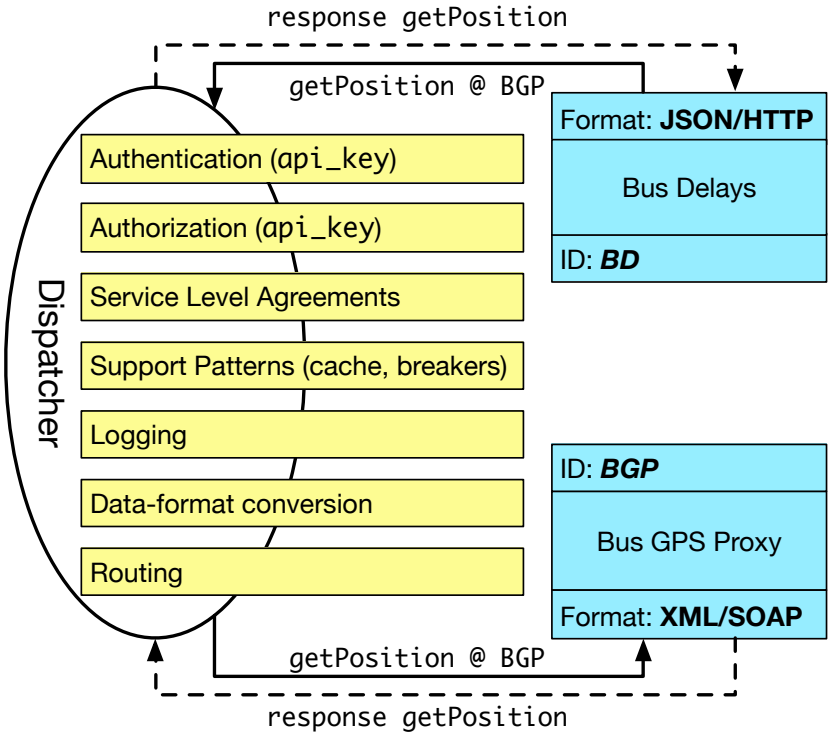


FIGURE 5.4: Example workflow of the SMAll Dispatcher.

handling confidential information, to avoid dangerous leakages [29].

Finally, regarding API definition, in SMAll we chose to support RESTful [63] and Jolie ones. On the one hand, we chose to support RESTful interfaces for compatibility, given the current adoption of RESTful technologies. On the other hand, we argue Jolie interfaces to be more flexible than RESTful ones (e.g., they are not constrained within HTTP verbs). Moreover, Jolie interfaces enjoy desirable features like outbound and in-bound checking for compatibility wrt the specified API [136]. Hence, when present, we preserve Jolie interfaces of microservices and provide tools[5] for the automatic conversion to RESTful ones.

Orchestration — Routing The last feature we consider here is routing of requests to registered services, which in SMAll is embodied by the Dispatcher service. As an example of interaction with the Dispatcher, consider Figure 5.4 that depicts the invocation of the Bus GPS Proxy from the Bus Delays service (Figure 5.1).

In Bus Delays the programmer writes the orchestration code, labelling the invocations to the Bus GPS Proxy with its identifier `BGP` — as mentioned, programmers can

[5]The Jolie REST router: https://github.com/jolie/jester

obtain microservice identifiers at deployment-time or through the Discovery.
At runtime, each invocation done by the Bus Delays service passes through the Dispatcher, which interprets the label assigned to the invocation and redirects it to the actual deployment address of the Bus GPS Proxy. The Dispatcher handles the routing of the response back to the invoker.

The Dispatcher also plays a central role in the development of microservices in SMAll (similar to API Gateways [148, 7]) and supports the advanced features, like access policies and service level agreements, of the third tier of the MaaS Stack. These features (rectangles in Figure 5.4) comprise:

- *Authentication* and *Authorisation*, forwarding only requests allowed to interact with the invoked service;
- *Service Level Agreements* which e.g., regulate the rate of calls per time unit and certify the respect of availability and responsiveness contracts;
- *Support Design Patterns* for microservices, like circuit breakers [138], caches, etc., which normally would require a direct integration within (and modification of) the microservices, as done e.g., with Netflix Hystrix [146];
- *Logging*, which is useful for debugging and security;
- *Data and Channel conversion* for the seamless integration of heterogeneous services, e.g., in Figure 5.4 the request of the Bus Delays service uses HTTP and JSON while the Bus GPS Proxy uses XML over SOAP.

5.3.3 Tier III

The third tier contains the most advanced features of SMAll, devoted to the creation of a global MaaS market.

We note that SMAll encompasses two types of federations.
The first one is at the level of eMobility operators, which can trade and access their services (becoming MaaS operators). These federations are defined *dynamic* because *i*) they exist at runtime, according to the automatic enforcement of the Access Policies and Service Level Agreements of the invoked service and *ii*) they live as long as their related transaction between the parties.

The second type of federation concerns SMAll instances. Here, the federation is *static*, i.e., the owners of SMAll instances define agreements regarding the inter-communication technologies [93], the security, reliability, and availability of their link, and the security requirements within their platforms.
With federated SMAll instances, eMobility operators belonging to distinct instances can trade their services and establish dynamic federations that span different geographic contexts.

The fundamental components that allow SMAll to form a unique market of eMobility services are the Discovery and the Dispatcher. Indeed, once these two components are aware of the presence of other instances of SMAll, they are able to automatically route discovery queries and service invocations towards the other federated platforms. To do this, both components forward their (respective discovery and invocation) requests towards their equivalent in the targeted platform. The forwarded request is handled as coming within the targeted platform. This is the context where the advanced features of the Dispatcher heavily come into play to enforce Access Control and Service Level Agreements.

Finally, dynamic federations of eMobility operators enable the support for roaming, the hallmark of Mobility as a Service, i.e., that MaaS users can integrate into their travelling experience the eMobility services and the transportation solutions of other operators. Clearing services are the last piece that completes the picture in SMAll, as they compensate usages of transport solutions as well as of eMobility services, according to the contract agreements.

5.4 Problems: Insider Threats

Statistically, insider threats are one of the most expensive security issues for business companies [142]. One prominent reason of these expensive outcomes is that companies did not foresee all possible malicious insider activities [182]. Indeed, the problem is not the lack of proper countermeasures as much as the difficulty of identifying a malicious insider in the first place. Literature abounds with guidelines and principles aimed at providing general descriptions of the context and the identity of the insiders [64, 40]. However, experts agree that the strong contextual variance

of threats [171] makes providing a general yet precise identification of all possible insiders difficult.

Thus, we deem useful to share the experience we gained in the context of services for mobility (both at software and physical level). Moreover, our background on the development of SMAll provides insights on the possible threats deriving from federated cloud architectures, built for deploying, publishing, and trading services. Federated clouds have been already analyzed in literature [102, 150, 192], however we deem important to include the related threats in the frame of the emerging Mobility-as-a-Service scenario.

In section 5.4.1–5.4.3, we illustrate, for each tier of the MaaS Stack (cf. Fig. 5.2), the insiders, the related attacks, and the possible countermeasures, as found in the state of the art and as implemented in SMAll. In Fig. 5.5 we report a table that summarizes our findings. Agents and threats are classified according to the categories identified by Casey in [34] and the CERT technical report [177]. We dedicate the last paragraph of this Section to a brief description of the methodology we followed to recognize the threats and the respective countermeasures.

Methodology As mentioned, adopting a narrow definition of insider may hinder the identification of threats specific to particular contexts. Therefore, in our investigation, we prefer to look at insiders from a general point of view [18]:

> *A trusted entity that is given the power to violate one or more rules in a given security policy [...] the insider threat occurs when a trusted entity abuses that power.*

This definition hints that an insider is determined by the role played as member of a system and related to the deployed control rules and the pursuable malicious goal(s).

In our context, the most classic scenario is one where the insider is within the service of the victim, e.g., a programmer that manipulates the behavior and the data of a service.

However, orchestrations spanning many providers, hallmark of the SMAll platform, lead to subtle yet relevant threats. Consider the case of federated partners. On one hand, the provider of a service exposes itself to threats posed by members that use

Tier	Agent	Agent Type	Insider Threat	Event Type
1 & 2	User	Competitor, Untrained/Distracted Insider, Outward Sympathizer	Fake Data Injection	Sabotage
			Service Behavior Manipulation	Product Alteration, Sabotage
	Unauthorized Guests	Competitor, Theft, Activist	Man-in-the-middle Attack	Misuse
	Developers	Competitor, Partner, Disgruntled Insider	User Impersonation	Sabotage, Espionage, Misuse
	Service Administrators	Partner, Disgruntled Insider, Untrained/Distracted Insider, Supplier	Insider Impersonation	Sabotage, Espionage, Misuse
	Service Managers	Partner, Disgruntled Insider, Untrained/Distracted Insider, Supplier	Crowdsourcing Attacks	Sabotage, Financial Fraud
	Agent after privilege escalation	Activist, Competitor	Data Leakage - Accidental - Data Theft - Resale of Data and Access - Business Intelligence Data Theft	IP Theft, Opportunistic Data Theft, Physical Theft, Accidental Leak
3	Federated MaaS Member	Competitors Nation State Partner Supplier	Data Manipulation, Trustability, Tampering of Data Provenance, Data Trustworthiness	Financial Fraud, Product Alteration
			Service Behavior Manipulation	Financial Fraud, Product Alteration
			Composition of Unverified Services and Data	Misuse, Sabotage, Espionage, Product Alteration
			Denial of Service	Sabotage
			Service Workflow Manipulation	Misuse, Sabotage, Espionage, Product Alteration
	MaaS Competitor	Nation State, Partner, Supplier	Data Analysis:	
	Helper Service	Competitors, Nation State, Partner, Supplier	- Pattern Extraction - Data Mining - Data Exploitation trough data crossing	Accidental Leak, Opportunistic Data Theft Espionage, Financial Fraud

FIGURE 5.5: Summary table relating the tiers of the MaaS Stack to their concerning insider threats.

its service — security issues span from misuse of information extracted from the service to over-usages that entail unforeseen costs or outages — on the other hand, an agent that orchestrates services of other partners is a man-in-the-middle able to leak private information, counterfeit data or use its vantage point to extract strategic patterns from partners.

Regarding countermeasures, we structured our analysis of the possible alternatives following the review compiled by Hunker and Probst [92], encompassing the three approaches: *i*) *Prevention*, comprising the definition of strong access control rules, data management systems (including data masking and data camouflage), and mechanisms to guarantee data provenance and data trustworthiness; *ii*) *Detection*, that usually goes hand-in-hand with dissuasion mechanisms such as techniques of data management and service invocation that make abuses extremely expensive in terms of computing power; *iii*) *Mitigation*, that exploits auditing and monitoring techniques, often based on machine learning, to automatically react to insiders' activities.

5.4.1 MaaS Stack | Tier I

As reported in section 5.2, the first tier of the MaaS Stack focuses on single eMobility operators and categorizes their services.

In this tier, the ecosystem of services has a flat structure and all members play the same role of providers, without any interaction between each other. Here, insiders

can be pinpointed within two types: *i*) *users* authorized to interact with services and *ii*) *managers* (also seen as owners) of the services. In the reminder, we call Users the members of the first type and Managers the members of the second one. The distinction between the two types is trivial: while Users have limited access to data and functionalities of a service, Managers can have full or partial control (depending on the responsibility level) over the life-cycle of the service and its resources. Users allowed to interact with SMAll services can basically pose two types of threats: *i*) perform fake data injection (for crowdsourcing-based services) and *ii*) sharing the access to the services or to the respective data.
Users can also exploit vulnerabilities to acquire Manager privileges (configuring an *Insider Impersonation* threat). However, we do not include a discussion on these kind of attacks as they coincide with those described for Managers. Regarding Managers, their main threats comprise:

- manipulation of the behavior of a service, i.e., the computations done by a service;
- manipulation of the workflow among services, i.e., the direction and sequence of information flow among services;
- stealing data, metadata, and performing malicious analyses;
- exposing sensitive information.

Following the first tier of the MaaS Stack, we describe the possible insider attacks of each category of services.

Information

The category of Information spans from basic services that publish raw data (e.g., timetables or the position of vehicles) to higher-level services that elaborate raw data to extract new information (e.g., the expected delay of buses whose calculation requires the position of a vehicle and its scheduled plan). As already mentioned, the Information category is the point of conjunction between high-level services for MaaS — and, by extension, MaaS markets — and sensing devices in Cloud of Things. Indeed, most of the real-time data used by this services is generated by embedded,

portable or even wearable [62] devices/sensors. Since in this work we separate the analysis on services and on CoTs, here we consider only threats at the level of services, dedicating section 5.5.1 to the analysis of threats on CoTs.

Notably, since Information services orchestrate other services to calculate and publish these refined data, they are subject to *Service Workflow Manipulation* and *Composition of Unverified Services and Data* threats. We omit to present these issues in this Section and defer the discussion to section 5.4.1.

Data Leakage Data leakage is the accidental distribution of private or sensitive data to unauthorized entities [174, 33]. In SMAll, both Users and Managers can cause data leakage. Users can share data to other, non-authorized Users. Similarly, Users can also share their access to services, which could lead to data leakage but also to other type of threats like *User Impersonation*. As expected, data leakage becomes even more serious when considered for Managers that can share or steal sources unreachable by users.

Countermeasures Data leakage poses a serious issue in open networks where the transition of data is not regulated nor monitored in their path. In these regards, SMAll holds a privileged position. In fact, all communications among the services in the platform happen through the Dispatcher (cf. Fig 5.1), which can log the quality and quantity of information required by all Users. Obviously, this guarantee ceases when data exits the platform. The same tracing system applies also to Managers.

Crowdsourcing Attacks Users can perform insider attacks on crowdsourcing services. These services handle data streamed from sensors and devices or through direct signaling of the users. An example is a crowdsourcing service where users can report architectural limits for people with disabilities [130]. In this case, insiders can feed the service with fabricated data to alter the normal behavior of services, e.g., by directing users through specific pathways.

Countermeasures For the sake of completeness and clarity, let us start from the literature regarding "classic" threat scenarios. Cho et al. [39] examined how insider attacks can exploit security holes in a trusted network of sensor nodes. This work is of interest for our platform because it shows how even trust-based approaches, in

architectures that have to unify many nodes, are not guaranteed to prevent attacks.

In [69], the authors described how access control policies for a database management system can be exploited by insiders when the control restrictions to be enforced may come from different authorities. Shatnawi et al. [175] made a similar analysis but based on the detection of malicious usage of a data source, which is equivalent to our case of a malicious influence of data source services exposed by the SMAll platform.

An interesting work that can be applied to our architecture is [181]. Here the authors implemented a pool of honeypots to catch insiders. A honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource. The high flexibility of honeypots — able to play a huge variety of SMAll-compliant services — is essential to make insiders expose themselves. Another useful method that can be easily built within SMAll is a reporting system for crowdsensing and crowdsourced data, implemented in [131]. The reporting system is based on the mapping of what the authors called Point of Interest (POI). Each POI and its related data can be added to the system by means of one or more reports. Reports are classified in three different source classes, accordingly to the reputation of the user that collects the data.

Service Behavior and Data Manipulation As expected, insider threats posed by Managers constitute a more complex scenario.
This type of insiders can access and modify the raw data of services as well as manipulating their logic to present altered results.
Notably, since in our context the physical world mixes with that of software services, we extend the role of Managers not only to the developers that can modify the actual code of the service but also to conductors and other operators: agents that can access and manipulate the physical devices that feed the services.

The manipulation of these services can have many purposes from the point of view of an insider. For example, during the development of SMAll we interacted with many industrial partners, among which there were some public transportation companies that provided real-time positioning of their vehicles. However, some of these companies did not report the actual position of buses and instead published

fake positions to mirror the exact planned schedule. In another case the service worked intermittently. In the first case, the company provided fake data to protect itself against possible penalties due to delays, in the second case the positioning service went down for certain rides due to drivers that disabled the in-vehicle positioning devices either for fraudulent purposes (to avoid being scrutinized) or even for shallow reasons such as to disable annoying automatic voice announcements.

Countermeasures Interesting works tackle the issue of how to predict insiders activities. Machine learning techniques have proven to be very effective in the detection of attacks by identifying anomalies in network traffic [119]. Furthermore, machine learning algorithms have been successfully used to detect past and ongoing attacks based on analyses of changes on the writing style of the users [86].

Althebyan [5] implemented a prediction model based on graph theory approaches, to push alert once a detection risk mechanism finds that users are performing actions that might lead to compromise the system services.

Studies also exist aimed at discovering malicious command execution. Among the most relevant works, Kamra et al. [101] and Mathew et al. [123] focus on the analysis of anomalous commands executed on databases. In particular, they proposed a syntax analysis system to detect anomalous queries; the former analyzed the submitted SQL queries, while the latter focused on data retrieved from queries. Doss and Tejay [53] conducted a similar investigation as a field study within an enterprise, where analysts were monitored while performing their jobs. Again, these results can be readily applied in our architecture, especially considering that tier I services will in any case be monitored by probes needed to build Business Intelligence services of the second tier.

In principle, the SMAll service deployment interface can verify the correctness of an application before accepting it. In practice, this operation is very hard to perform. One indicator of correctness is the compliance to a template of acceptable interfaces for the kind of service the application provides. However, it is very difficult to define templates strict enough to allow sensible compliance checks, but general enough to avoid hindering the deployment of legitimate services.

Another important detection strategy that we considered is to implement a mechanism that could guarantee, in every moment, a reproducibility of the results of a service. With provenance certifications of raw data and their propagation to results, it is possible to implement a reference monitor to verify compliance between results and expected values. In case of conflicts between the declared results and the actual ones, SMAll could discover what has been tampered with: the source data, or the service logic. This detection can also feed a data trustworthiness rating system.

Finally, another way to check correctness is to look at the actual behavior of the application, as it is common in anti-malware checks. These techniques are far from infallible, and their scope falls much shorter than what is required in our setting. Indeed, in this context a malicious behavior can be a subtle deviation from the correct calculation [140], which is far more difficult than the detection of traditional malicious behaviors (e.g., damaging or self-replicating ones). Promising techniques, which can benefit from the execution of all the services on the SMAll platform, are those based on the aggregation of multi-domain information [57, 6].

Travel

Services in the Travel category orchestrate Information ones to provide highly coordinated functionalities to users. Since the services in this category heavily rely on composition to provide their functionalities, their main concerns regard their workflow.

Service Workflow Manipulation Managers can modify the expected flow of information among services for many purposes. As an example, consider the Manager of a service called Bus ETA that predicts bus arrivals. In its calculations, Bus ETA uses three source-services, respectively for traffic, GPS positioning, and weather forecasts. Although the Manager preserves the logic (i.e., the behavior) of the Bus ETA service, by simply changing the workflow, i.e., the bindings of the Bus ETA to the other services, she can make (some) of the sources unreachable, either completely disabling the Bus ETA service or modifying the resulting output due to missing data.

Countermeasures SMAll already provide tools to contrast service workflow manipulations through the helper services Dispatcher and Business Policies (Fig 5.1).

Indeed, when Managers deploy their services in SMAll, they also define the related access rules (stored and retrieved in the Business Policies service). Then, all workflow compositions pass through the Dispatcher service that logs them and enforces the established access policies. In this way, unexpected workflows are detected, logged, and (depending on the access rules) forbidden. The monitoring capabilities of the Dispatcher can also be enhanced with features like the one presented in [170], which predicates the provision of enclaves (in our case, provided as a service) to reduce the surface area of data exfiltration of workflows; in [207, 208], where anomalous traffic is detected by mapping user input activities (e.g., their requests) to their generated chain of events on the interested microservices, or more in general by finding causal relationship among network events in large datasets; and in [60], where the Dispatcher can use machine learning engines, similar to the ones used in dynamic malware analysis, to detect malicious workflows. Finally, based on service specifications, we can create workflow graphs for strategic mitigation [190].

Another promising approach comes from the field of Choreographic Programming [135]. The use of choreographies to implement workflows among services is relatively new [72]. We deem choreographies an effective prevention tool that lets partners agree on a formal definition of their workflows, which can be later compiled into their respective, compliant services. Moreover, in the dynamic context of SMAll, tools like [48] can aid partners in updating their agreed workflows even at runtime (i.e., without stopping their running services). These updates would be still conditioned to a general agreement and maintain the same guarantees of the original services.

Mitigation techniques can be also developed following e.g., [74]. The idea would be to develop a SMAll helper service that monitors workflows and, once an attack by an insider is discovered, it appropriately redirects the workflow to avoid further damage.

Composition of Unverified Services and Data In the context of mobility, verified information is of paramount importance. However, in a service-oriented architecture, the tricky part to deal with is that a service invocation can be seen as a collection of workflows. These workflows can compose many levels of services, each

processing and modifying the data before its final destination. These services inherently include the logic of the composed services and, by extension, also the possible manipulations executed by insiders. As an example, consider a journey planner that uses a real-time traffic report service to avoid traffic jams and roadblocks. Since the journey planner directly integrates the information from the traffic report service, manipulating information of the latter alters the solutions of the journey planner, diverting travelers towards certain pathways. This case presents an interesting nuance: the insider is not a direct Manager of the considered service (i.e., the journey planner), instead it is the Manager of a composed service (the traffic report) that twists its contribution to alter the behavior of the planner. In this context also trustability, provenance, and trustworthiness of data and/or services should be considered as possible targets of attacks. For example, tampering with data provenance is a source of attack [178] that in a MaaS scenario can see malicious operators claiming to publish genuine data of a competitor, actually forging them.

Interfering with the certification of data trustworthiness is another possible vector. In this case, it is very difficult to block attacks in which, e.g., the creator advertises a data source of given quality, but then exposes a degraded version to keep the advantage of more precise/timely information for herself. A related trustworthiness scenario is that of an insider who succeeds in registering a rogue service. For example, a modified travel planner could deflect routes to favor or damage certain businesses; a modified delay-checking application could hide or amplify violations of agreed service levels.

Countermeasures A service must support the provision of different sources of data along with their associated metadata (e.g., used to verify their provenance). However, SMAll shall also provide techniques, embodied by helper services, to transform those data into verified information. There are different approaches that provide a solution to the problem of recognizing the source of a data stream. Literature agrees [79] that the requirements for a provenance management system are: *Verifiability:* a provenance system should be able to verify a process in terms of the actors (or services) involved, their actions, and their relationship with one another; *Accountability*: an actor (or service) should be held accountable for its actions in

a process. Thus, a provenance system should record in a non-repudiable manner any provenance generated by a service; *Reproducibility*: a provenance system should be able to repeat a process and possibly reproduce a process from the provenance stored; *Preservation*: a provenance system should have the ability to maintain provenance information for an extended period of time. This is essential for applications run in an enterprise system; *Scalability*: given the large amounts of data that an enterprise system handles, a provenance system needs to be scalable; *Generality*: a provenance system should be able to record provenance from a variety of applications; *Customizability*: a provenance system should allow users to customize it by setting metadata such as time, events of recording, and the granularity of provenance.

In these regards, it would be useful to deploy technologies to certify the metadata related to a data stream and manage its validity during time and re-elaboration [187]. According to works like [180], this problem could be solved only with the creation of a public-private key system for data stream certification. A good reference is the system developed in [198], describing a cryptographic provenance verification approach for ensuring data properties and integrity for single hosts. Specifically, the authors designed and implemented an efficient cryptographic protocol that enforces keystroke integrity. This kind of protocols can be integrated as a helper service in SMAll. However, public-key schemes are known for their significant computational load, thus existing techniques may not be suitable for high-rate, high-volume data sources. Moreover, there could be the need for an algorithm for the provenance of composed data. In some cases, data originated from the composition of raw (or otherwise lower ranked) sources should be accompanied by suitable metadata for verifying the provenance of the input values, in a cryptographically strong way. In the context of SMAll, it could be important and useful to capture and understand the propagation of data.

The combination of metadata- with key-propagation management can guarantee a good level of trust in provenance management systems. Works in the direction of [83] discuss how to support provenance awareness in spatial data infrastructure and investigates key issues including provenance modeling, capturing, and sharing, useful to implement key propagation systems.

Finally, we address trustability, provenance, and trustworthiness of services and/or data.

Trustability needs to be measured by indicators for data quality and service behavior. Values for these indicators come from a variety of considerations on basic data sources. However, it is challenging to define algorithms for source evaluation based on data resulting from services aggregating and orchestrating other sources [61, 55]. Ascertaining provenance means ensuring that the source of data is verifiable, i.e., that it corresponds to the one declared in the process of creation. Trustworthiness is intended as the possibility to ascertain the correctness of the information provided by a data source, which is loosely related to provenance [47]. Ideally, but infrequently, data samples can be independently measured by different users, thus allowing cross-checking and error correction. For original data, i.e., provided by its creator, the trustworthiness score is usually derived from the reputation of the creator. Clearly, guaranteeing data quality, provenance and trustworthiness is not enough, it is necessary to ensure that the computation is correct and that no useful results are hidden (completeness).

User

The last category of services of tier I is not specific to mobility but it contains essential functionalities for the other two categories. The most representative case is that of User Profiling and Management. User profiling is not required to create services for mobility, but it has become essential to ensure usability, to provide user assistance, and to even anticipate and plan for the next movements of the user (cf. Google Now[6]).

Data theft Here, the most obvious threat regards the possibility of stealing information derived from the profile dataset, such as preferences, recordings of movements, orders and payments.

Countermeasures In our setting, a possible approach is to empower the user with control over its profile and the related access policies [8].

[6] https://www.google.com/search/about/learn-more/now/

5.4.2 MaaS Stack | Tier II

Business Intelligence

SMAll second tier of the MaaS Stack adds a new category next to the ones of the first tier: Business Intelligence, i.e., services exclusively dedicated to provide insight on the usage and performances of services of the first tier.

This services can implement any kind of data mining algorithm useful for monitoring the profitability, sustainability, and reliability of the provided services, as well as for determining trends and making predictions on future usage, for capacity planning and policy definition. Most of this algorithm and services do not usually works on the physical devices but they are part of a Cloud Architecture.

Business Intelligence Data SMAll Business Intelligence analyses are important source of sensitive information for insiders (also in this case Managers with privileged access) that could expose relevant data to third parties. Indeed, without Business Intelligence services it would be very difficult or even impossible for insiders to obtain such data, that otherwise would require the access to massive amounts of private information over long periods.

Managers of Business Intelligence services can apply targeted analyses to infer reserved information, such as policies and business strategies of their company. An example of this type of attack is what we simulated in [30], where by just analyzing the database of validated tickets of a public transport company of the urban area of Bologna, we were able to reconstruct the distribution of the various types of tickets in the different zones of the city.

Countermeasures SMAll serves the purpose of mediating the access to relevant data for Business Intelligence. Every operator wishing to obtain statistics or performance indicators about its own services can freely create instances of the platform-approved analytics services.

Regarding mitigation, the most effective way to hinder the possibility to misuse Business Intelligence services is to properly sanitize the datasets and to control the workflow of this information. SMAllse techniques [133] aim to prevent insiders from

correlating Business Intelligence services with external data sources to derive hidden patterns or de-anonymize sensitive information.

5.4.3 MaaS Stack | Tier III

The third tier of the MaaS Stack is that of MaaS operators, i.e., eMobility operators that use services of other companies, traded within a federated market. In our case, SMAll gives support to such a market but the creation of dynamic federations of MaaS operators rises specific threats within SMAll (and MaaS markets in general).

In this scenario the main issues to consider are:

- Data service management to avoid manipulation, impersonation, and sensitive pattern discovery (Prevention and Detection);
- Service workflow management to monitor invocation trends of services (Mitigation and Detection);
- Service quality and trustability management to verify the correctness of the service results (Prevention and Detection).

Indeed, the PaaS layer in SMAll differs from most PaaS solutions. Traditionally PaaS provides offer execution environments that isolate tenants. On the contrary, SMAll is built to ease the publication, integration, and orchestration of services owned by different operators.

A simple example to clarify this characteristic is a one-stop ticketing application that orchestrates:

- a dynamic planner service providing routing options;
- a user profile manager to sort them according to user preferences;
- a real-time availability seat reservation service;
- a set of services for payment.

The hierarchy of the ticketing service spans many layers, e.g., it integrates the dynamic planner that, in turns, orchestrates many services for static (mapping, timetables) and real-time data (delays, planned extraordinary events, disruptions). The

composition of services forms a tree of dependencies that reaches the level of raw-data information services, possibly branching within the domains of different companies.

Since SMAll aims at supporting this kind of interoperability, we argue that it shall also assume responsibility for the trustworthiness and reliability of the services; this is unusual for traditional PaaS [118]. Moreover, access control policies can be heterogeneous, exchanged data can have different sensitivity levels, and the agents can be competing operators.

Clearly, the main insider threat for this scenario comes from the service providers themselves, the MaaS operators. The malicious goals can be of various kinds, spanning from the de-legitimization of services of competing operators, to the theft of stored information such as policies or business strategies, to insiders that apply mining techniques to infer these information using the data available from their vantage point.

We now proceed by focusing our analysis on the relevant insider threats within the categories of the third tier of the MaaS Stack.

Roaming and Clearing

SMAll aims at providing interoperability between different operators. In this context, interoperability means that it is possible to implement ticketing systems which seamlessly work on different operators across their zones of influence. As mentioned in § 5.2, this concept (and the category of services that supports it) takes the name of Roaming. Usually, to support at a business level the roaming for users among operators, business agreements should be put into place to implement a Clearing system for the redistribution of profits between transport operators. In this Section, we consider threats as directed to the Clearing category since it comprises also the threats to the Roaming one.

Pattern Extraction As analyzed in [30], the need for Clearing services is satisfied through a centralized (federation-wise) system able to collect all the different data sources from different operators and to perform economic evaluations. A centralized clearing system scenario is typically based on a central database that collects

all the ticket validation data from every public transport operator. This database is used both to perform economic evaluations to redistribute profits and to store a permanent proof of the validity of this evaluation. The clearing system must fulfill an effective trade-off between public verifiability of the correctness of its operation and protection of sensitive data provided by operators. As the last cited work shows, an insider can perform data mining analysis and pattern discovery on the tickets datasets in order to retrieve sensitive information about business strategies and perform unfair competition.

Countermeasures To counteract *Pattern Extraction*, it is possible to deploy sanitization techniques [134] able to mask the data enough to deny the possibility to perform pattern analysis. These sanitization techniques balance the needs of masking sensitive data and keeping enough properties and information to perform the economic evaluations. In order to do what we described, we could assemble an anonymization system, that combines masking techniques for the raw dataset (once deployed in the centralized database clearing system) and a differential privacy engine able to introduce a certain amount of noise and prevent exploit techniques as cross-combining data with external ones.

Access Control and Service Level Agreement

Service Level Agreement (SLA) and Access Control (AC) services in SMAll are meant to throttle the invocation of tier I services provided by an operator on the basis of commercial agreements with other operators. It is possible to see SLA as a contract ruling the quantity or rate of invocation of each service, and AC as a contract ruling the quality or the set of provided data or services. Obviously, malicious insiders may try to circumvent these limitations.

Countermeasures When a SLA or an AC policy is in place, all service invocations must be tracked (or even proxied) by an infrastructural service provided by SMAll. This makes evading enforcement difficult. The most common vulnerability in this context is not tied to policy enforcement, however, but rather to policy specification. To this end, SMAll could restrict acceptable policies to those drafted with an internal helper service, following a standard framework, and formally verifying their

soundness before applying them. Access control models and formal policy specification languages have been around for some time [167, 49], and they have evolved into sophisticated, standardized models like ABAC [91, 166]. Inadequate (but consistent) policy definitions due to poor understanding of the federation interactions or to carelessness cannot be tackled at this level; logging and auditing facilities integrated in SMAll provide valuable feedback at run-time about the effectiveness of installed policies.

Business Intelligence

Similarly to tier II, in tier III we have a category of services dedicated to business intelligence. The difference with respect to the services of the second tier is that here the analyses span data belonging to a multitude of operators. Indeed, as it happens for clearing services, the business intelligence services of the third tier relate to the management of data, statistics, and administration of services shared among operators. The availability of such aggregated data can give free access to companies (seen as federated insiders) to data and analyses of competitors. Referring again the case of the dynamic route planner as a running example, the service can use real-time data of different companies to take into account the average delays of transport vehicles in the calculation of its solutions. The averaged delays are the result of a business intelligence service that collects all the delays of a route within a specific area that involves several operators and calculates the delays. Finally, the recorded delays are collected into a shared dataset accessible by all the participants.

In this example, an insider can use the collected dataset to find out where the competitors operate with bigger delays and profit from this information by exposing their faults to the regional administration. Insiders can also expose cartels where operators systematically provide a bad service during rush hours to favor a specific company (e.g., because they hold some economic interest in it). Finally, the insiders can also find out if an operator hides delays making analysis on the correspondent road conditions (e.g., showing that buses could not sustain certain speeds since their routes were jammed).

Countermeasures All the countermeasures for this kind of attacks are based on a trade-off between the amount of sensitive data preserved and utility of the queries. Different anonymization and sanitization techniques have been proposed for complex datasets, but since in SMAll Business Intelligence services share the results of queries, we need to introduce a measure that indicates the maximum amount of anonymized information such that the queries still work.

Different works proposed metrics for the evaluation of the amount of privacy preserved in specific dataset. A measure introduced in [144] defined an evaluation metric about the presence of pattern in a dataset called δ-presence. We can use this metric to evaluate the presence of a specific patterns in the shared dataset. Another interesting work in this direction is [85] which operates by complementing existing techniques with post randomization methods.

5.5 Use Case

5.5.1 Cloud of Things for MaaS

After our general analysis of insider threats in MaaS, we concentrate on Cloud of Things (CoT): one of the main enabling technologies for MaaS. We mentioned the role of Cloud Platforms in MaaS and the threats linked to their ubiquity and continuous connectivity in section 5. Recognizing that a solution at the low level of CoTs could positively impact the security of MaaS services, we deepen here our analysis on insider threats of CoTs, in the context of MaaS.

In general, the Internet of Things [10, 80] and the system of networks of IoT devices that constitutes the, so called, *Cloud of Things* [12, 87], relies on the idea of a world-wide network of interconnected entities. Concretely, these entities are heterogeneous elements interacting over disparate systems of networks: the definition extends to comprise human beings and computers as well as general-purpose environmental sensors (light, humidity, temperature, sound) to specific devices like road traffic monitors or GPS trackers. Making all these entities interact with each other had and is having sensible, successful applications in multiple domains like automotive, health-care, logistics, environmental monitoring, and many others.

CoTs play an important role in enabling many of the modern features of MaaS services, indeed, entities within the context of CoT share three common denominators [161]. They are:

- *locatable* at multiple layers, spanning from their position within an interconnected network to their actual geographical location;
- *addressable* in such a way that they accept connections from other entities;
- *readable* other entities can query them to obtain some information.

If on the one hand these properties constitute the promising characteristic of CoT, on the other hand they make CoT-based networks open to many kinds of malicious attacks conducted by a plethora of possible adversaries.

Cloud of Things and MaaS: Insider Threats

Considering the categories of threats analyzed in section 5.4, we summarize a brief account of the possible attacks in the context of CoT linked to insider activities:

- *Data Leakage* entities can accidentally release private or sensitive data to unauthorized entities. Malevolent attackers could also gain information from "alternative" usage of sensors, e.g., by employing temperature, light or audio sensors to check the presence of people;
- *Crowdsourcing Attacks and Data Manipulation* where entities that publish information feed fabricated data to change the behavior of the services that rely on them;
- *Device Misbehavior* insiders can exploit weaknesses in the protocol of interaction of entities, e.g., sensors and actuators, to cause malfunctions and hardware failure.

Given the threats above, we investigated whether the networking architecture of CoT entities and its working protocols could mitigate and counteract some of the vulnerabilities analyzed in Section 5.4.

The idea has already been speculated in literature, for example, in [161] the authors dissect the issue as driven by the two principles of *location of intelligence* and

degree of collaboration. The first indicates where the intelligence resides in the network, i.e., if edges of the network provide services rather than simple data. The second regards the degree of interconnections among heterogeneous entities, i.e., if they are mainly partitioned based on their nature (sensors for probing, servers for collection and manipulation of data, etc.) or if they interact on a peer-to-peer basis.

Combined, these two factors characterize the architecture of the network interconnecting the considered entities. Mainstream solutions follow a centralizing approach, leaning towards "simple" entities that mainly collect information at the edges of the network and centralized, possibly hierarchical, hubs that aggregate, manipulate, and redistribute available data.

At the other end of the spectrum, there are decentralized systems that leave some freedom at the edges of the network, allowing entities to take some decisions. Entities can also constitute partitions with some emerging intelligence (e.g., regarding probing times, correction of sensing, etc.) without a direct control and sharing no information with a central, high-level entity.

From an insider threat standpoint, we argue that the centralizing approach constitutes the most dangerous configuration. Indeed, following such an approach, entities at the edge of the network are passive elements. They are open to any kind of query. Malicious ones could be used to infer private information and even cause malfunctions on the devices, e.g., by consuming their batteries or causing hardware failures due to overuse. Moreover, the approach is prone to the well-known issue called “central point of failure” where users of the network must connect to the central hubs, which in turn could be compromised to cause denials of service as well as to reveal sensible information of all users.

On such observation, we investigated networks characterized by a weakened degree of centralization and an increased collaboration among the entities. In such architectures, entities at the edges can process local information and provide it to both central hubs as well as to other peers. These decentralized networks enjoy a stronger degree of security as nodes can refuse to collaborate in requests that are deemed dangerous or deviate from established, secure protocols of interaction. In addition, if proper dynamic reconfiguration techniques are applied, even in case of malfunctioning central hubs, the local services remain accessible.

5.5.2 Federated Platooning

Generally platooning comprises a number of vehicles equipped with state-of-the-art driving support systems one that closely following and collaborate each other. This forms a platoon driven by smart technology, and mutually communicating. Vehicle platooning and especially truck platooning is innovative and full of promise and potential for the transport sector.

There are many ways to do platooning. One possibility is to instrument the vehicles with new and powerful hardware that work as a structure to implement the vehicle collaboration. This solution as several limits; hardware cost, upgrading cost and the main one difficulties on the automatic collaboration between different platooning platform which have to share the same standards and models.

Softwarize this process without the addition of new hardware overcome this limits but is not easy to do. As we pointed out previously the collaboration between platooning system of different region can exploit the power of platooning but there is a need of central authority in charge to enable and validate a federation of platform.

A federated platform of platooning system can improve traffic safety. Platooning is a cost-saver, fuel consumption saver which means less CO2 emissions. And, lastly, platooning efficiently boosts traffic flows thereby reducing tail-backs. Meanwhile the short distance between vehicles means less space taken up on the road. At the same time the impact platooning goes far beyond the transport sector. Automated driving and smart mobility also offer realistic chances to optimise the labour market, logistics and industry.

Enabling Platooning

We argue that an enabling architecture for such platform must be mobility service enabler platform, and we argue that our proposed, developed platform SMAll can support the creation of a federation-based service and markets where operators can publish, automatically retrieve, and orchestrate functionalities for mobility, provided by different operators.

The platform builds on the concept of Federated Cloud Computing and maintains

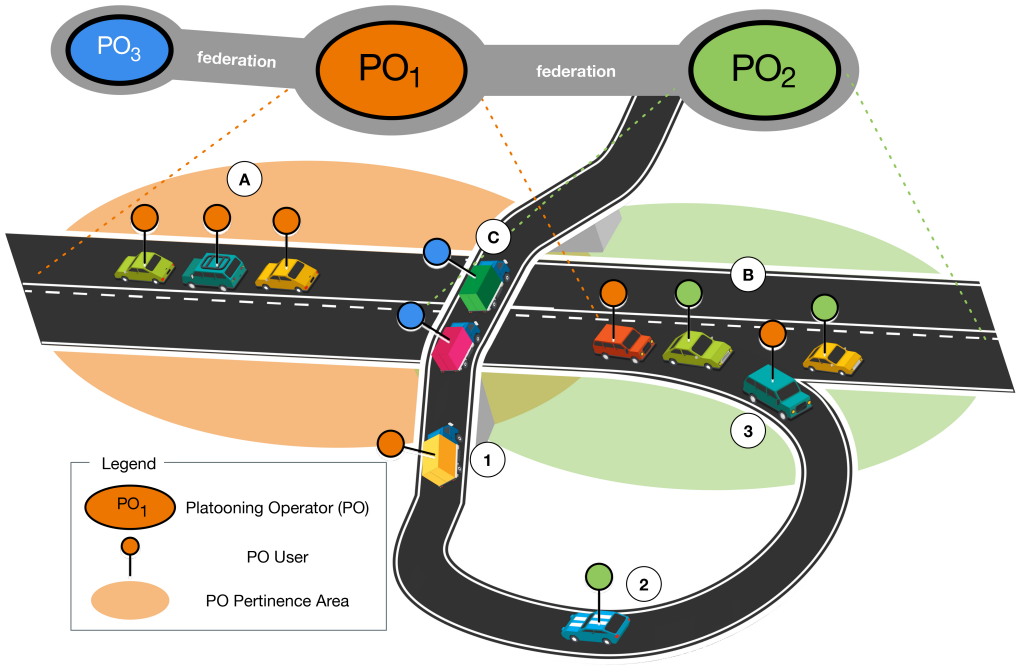


FIGURE 5.6: Example of platoon formation over federated SMAll instances.

an open approach with respect to the possible members of the federation, which correspond to any player that trades services and information linked to mobility.

To better illustrate our proposition, we report in Figure 5.6 a schematic representation of possible scenarios of a Federated Platooning with SMAll Architecture and proceed to comment them. In the figure we consider three SMAll instances, covering different geographical areas. We mark each SMAll instance with a colour, (orange for $SMAll_1$, green for $SMAll_2$, and blue for $SMAll_3$) where $SMAll_1$ and $SMAll_2$ coverages partially overlap, while the pertinence area of $SMAll_3$ is distant from the other two (and not represented in the figure). To simplify, in the figure we assumed a 1:1 correspondence between each SMAll instance and a platooning operator that belongs to that instance (in reality, many operators can belong to the same SMAll instance).

Let us first concentrate on the convoy pointed by label Ⓐ: the simplest case of a platooning service, which captures the basic occurrence in the literature where a

dedicated infrastructure or in-vehicle devices are coordinated by a centralised authority. Indeed, the scenario pictured by Ⓐ makes two assumptions: *i*) that users of the convoy belong to the same operator and *ii*) that they travel in the pertinence area of that operator (which corresponds to the pertinence area of the related SMAll instance).

The scenario pointed by label Ⓑ removes the two assumptions of Ⓐ: here vehicles in the convoy are users of different operators and some of these (marked in orange) are outside of the pertinence area of their operator. In this case, the two operators in $SMAll_1$ and $SMAll_2$ coordinated the formation of a mixed convoy. In Ⓑ, the business policies for service usage negotiation, defined by each operator, mediate between possibly conflicting formation-control logics deployed by each operator. To better understand this concept, let us suppose that the two operators, in $SMAll_1$ and $SMAll_2$, consider in their respective platoon formation algorithms the two constraints of cruise speed and fuel consumption. For example, the operator in $SMAll_1$ could offer to its users solutions that maximise cruise speed while the one in $SMAll_2$ favours fuel savings. Thanks to mechanised composition plans, defined as business policies in SMAll, the two operators can reach a dynamic distributed consensus over a mediated solution that satisfies the threshold parameters of their convoy formation algorithms.
To draw a parallel, this is similar to the negotiations conducted among mobile phone operators to allow their users to roam within their networks, without having to stipulate a specific contract with each operator. Here, the difference is that such a negotiation is dynamic, so that users of each operator can choose whether to accept or not the platooning plan negotiated by the two operators.

Moreover, while platoon formation essentially relies on a static plan where vehicles join and leave a formation at pre-determined times, the mechanised federation of SMAll instances allow for a higher degree of flexibility: platooning users can decide to switch between different convoys while travelling, as exemplified by the vehicle leaving the column in Ⓑ to join the convoy in Ⓒ. In this final scenario, the column pointed by label Ⓒ, composed of the two users of the operator in $SMAll_3$, is outside of the pertinence area of their home SMAll instance. Also in this case,

although the column is composed only of vehicles of the same operator, the platooning plan results from a negotiation between the two operator respectively in $SMAll_3$ and $SMAll_1$, where the latter manages platoon formation in its area of pertinence. Finally, thanks to the collaboration between all operators, the three vehicles ①, ②, and ③ can dynamically join the pre-formed convoy in ©.

In such a landscape, thanks to clearing services, given a travel plan, users can directly evaluate the actual gains (possibly expressed directly as monetary transactions) of many configurations, e.g., whether to join the tail of a formation or become the head of it, or switching between columns. This aspect holds also for users already in formation, which can agree to let other users join their formation and share the costs of the convoy. Beside direct user interaction, clearing costs and the related policies can be also part of the logic of composition among operators to select/propose optimal plans to their users.

Now that we presented the architectural characteristics of federated platooning operators, we shall analyze the possible security concerns of such a globalised collaboration. These span from the illicit acquisition of sensitive data of users to the deployment of malicious platooning plans, which could easily block the traffic of extensive areas, as well as threatening the safety of roads in general. To this aim, in the next section, we expose the main security threats of federated platooning and propose possible mitigations and countermeasures.

Insider Threats in Platooning

Once the main elements of platooning and federated platoon formation have been defined, it is possible to analyze the security concerns of such a global collaboration. These span from the illicit acquisition of sensitive data of users to the deployment of malicious platooning plans, which could result in a variety of consequences, ranging from inflicting economic losses to competitors, to affecting traffic over extensive areas, and even to threatening the safety of roads.

A federated environment natively foresees sharing one's own sensitive information, and access to others' information in a controlled way, for example to know basic data such as the kind of vehicles on the road, the most common routing choices, but in some cases also enough details to make tracking a vehicle or a driver possible.

This kind of sharing is the core enabler for the desired aggregation services. Consequently, security concerns are greatly expanded as the attack surface is enlarged and diversified with respect to a closed-world model of operation. This kind of threats are not entirely specific of such platooning federated platforms; they would appear as an intrinsic aspect of any approach to coordinate services of independent transport agencies. Thus, they are interesting to address because results can be applied to a wider set of scenarios, and at the same time the analysis can be built upon previous works. For example, in [30] Callegati et al. we showed how the members of a public transportation consortium can exploit its clearing system to infer strategic private information.

In the remainder, we show how these kinds of attacks are relevant in a federated freight scenario: we describe the main issues introduced by platooning and evaluate their effects from an economic point of view, showing practical abuse cases, and highlighting the key vulnerabilities. In 5.6 we present two, layered, composable technical solutions to mitigate the identified threats.

A discuessed in 5.4 the possible security problems of any platform based on controlled sharing of sensitive data are manifold and include threats such as: compromising the infrastructure where data is stored with the aim of subtracting it, intercepting data in transit by exploiting unsafe communication protocols, injecting falsified or malicious data by exploiting authentication weaknesses, and many others; these types of attack are not specific to our case study, but instead they are mainly related to the correct design and implementation of a data management infrastructure, a topic already widely covered in literature [36, 162, 109, 183], and for this reason they will not be discussed in this work.

The problem is not only the lack of proper countermeasures but also the difficulty of identifying a malicious insider in the first place [152].
Experts [168] agree that the strong contextual variance of threats makes providing a general yet precise identification of all possible insiders difficult. Besides the members of the federation (here, the platooning operators), federated contexts contain two additional broad categories of insiders: *i*) legitimate users, which could leverage access to service providers and make a malicious use of information extracted

from services or cause over-usages that entail unforeseen costs or outages; *ii*) orchestrating agents, which are needed to coordinate any complex architecture, can act as men-in-the-middle, accessing and/or leaking private information, as well as counterfeiting, throttling, hijacking or selecting data of legitimate users.

Hereinafter, we follow a categorization between privacy leakage attacks, where there is a theft of sensitive information, and disruption attacks, where an attacker interferes with the behavior or the structure of the system. We decided to focus only on these two categories of concerns for several reasons.
First, we must remember that we are analyzing the concerns from an insider threat point of view. For this reason, we assume that we already have a certain level of permission within our infrastructure. This is why all targeted user attacks for initial access to the infrastructure are not particularly interesting. Second, we do not take into account attacks on the exploit of the infrastructure for lateral goals, since, although important, they are not the main cases of insider threat that can be found.

Privacy Leakage All the issues described in this section are a consequence of the kind of information that the clearing/platooning system needs to share and use. Examples of relevant categories of data and meta-data include:

- vehicle movements details: origin, destination, middle stops, average time, average speed, etc.
- details of shipped items: origin, destination, weight, package category, etc.
- route planning and execution constraints: clearing instructions applied to the current convoy, speed limits, etc.

While needed to enable operators' participation in platoon formation, the same information can be used by insiders together with external data sources to compose a targeted picture of sensitive aspects regarding a victim. The kinds of "retrievable" information are categorized in the following, based on the insider's target.

User Attacks A single user's privacy exploit is an attack that an insider can do in many ways. There are various effective methods to retrieve information about an individual driver even if it is not explicitly shared. If any database contains pseudo-identifiers, for example, the vehicle's serial number, they can be exploited to obtain

the existing association between the vehicle id and the driver. For example, in [28] we demonstrated how to skim the various entries of a database, crossing them with external data from Open Source and publicly available OSINT tools and proceeding by exclusion to find the corresponding driver.

The process is described in principle in [31], where in a similar scenario of a data clearing system for urban public transport it was possible to trace a specific user's real identity by performing this kind of correlations. Knowing the precise movements and journeys of a vehicle driver is not only a powerful weapon against the vehicle driver's personal privacy; this information can be also used by competing companies to intercept the rivals' demand / supply patterns.

Business Policies Attacks For delivering companies, a fundamental competitive advantage derives from the methods and procedures with which they distribute the transport vehicles along their shipping routes. Indeed, it is shown [68, 153] that an efficient allocation of means of transport and drivers can reduce costs and management burdens. This information is strictly guarded by any company.

However, as explained above, to feed the necessary data into the clearing system, individual vehicle data must be (possibly partially) shared to calculate the correct redistribution of profits. In turn, illicit access or leakages of these data allow attackers to trace the distribution rules of the vehicles, as shown by us in [31], where through sequential analysis with clustering data mining algorithms the authors extracted the distribution of buses of a transport company present in a clearing system.

Unfair competition Lastly, we consider another type of sensitive information that an insider is able to retrieve. If information on vehicle routes and items carried is not properly anonymized, it is possible for insiders to cross them together and with external sources to interpolate financial details. Knowing which routes are the most profitable, what prices and which kinds of contracts companies offer on the same routes, makes it easy for unfair competitors to size contracts and deals [31].

Destructive Attacks The second category of attacks that an insider can launch goes beyond accessing an unauthorized resource and focuses on deriving advantages from the disruption of competitor's services.

An insider can try to achieve this result by injecting information in the clearing / platooning system. Carefully crafted malicious information may be the cause of a breakdown of competing services and businesses.

Routing corruption Let us consider a federated platooning system that holds all the data of the trucks and the shipped items of its members—as mentioned above, needed to calculate the correct redistribution of the profits and the management of platooning tasks. An insider might, in this case, inject malicious information aimed at excluding a specific route, causing denial-of-service targeted at one or more vehicle of a specific company. This exclusion can be achieved in several ways:

- declaring as additional meta-data fake speed limits to make the vehicle (think of time-guaranteed delivery trucks) late or to make a route plan inaccurately (in)convenient;
- posting false or incorrectly located traffic updates to force the competitor to change direction and redirect it to a wrong and / or inconvenient path;
- inserting non-existent routes that will force the vehicle to backtrack and recalculate along the way.

Competition starving The previous attack describes a targeted denial-of-service scenario, in which an insider attempts to block a competitor's specific service. The insider might wish, and be able, to do something more subtle. Instead of making a brutal intervention that causes a detectable disruption of a trip, the attacker could make a campaign of small, well-distributed manipulations that slowly impoverish the adversary economically. Such insertions, if successful, have the peculiarity that they cannot be immediately discovered, emerging only after a thorough and time-consuming analysis of the dataset. This kind of attack can be executed in several ways, for example:

- by performing the same kind of injections as described in the previous scenario, but introducing an error so small as to make it unlikely to be noticed as such, yet causing losses that end up having a disruptive cumulative effect;

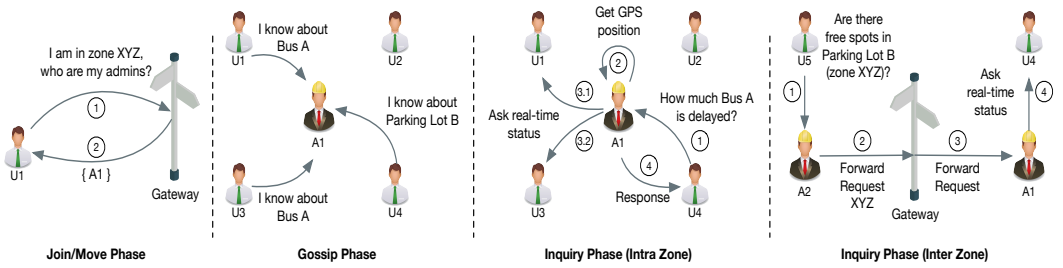


FIGURE 5.7: Phases of the Gossip Network.

- by making actual orders that force the competitor to deploy more vehicles (e.g., from a truck fleet) with lower profit margins, and possibly leveraging their presence to save more fuel from platooning.

5.6 Novel approaches to mitigation

In this section, we propose two layered technical solutions to tackle the data management issues outlined in the previous section. These two approaches stem from the observation that, from the point of view of data flow, it is not strictly necessary to have a centralized architecture to enable an effective exchange of information. On the other hand, information systems that support federation can be exploited to enforce correct behaviors of its members.

In the following, we first detail a decentralized overlay network for data safety and trustworthiness, then we describe the features of a dynamic federation platform, needed to monitor and interrupt deviant behaviors of federated members.

5.6.1 Overlay Network

A possible alternative to centralized data dispatching is to implement an overlay network created by the same entities of the federation. In this section we propose a solution based on a gossip protocol [81, 22], with the intent of mitigating the risk of sensitive information theft (eliminating the need of a centralized controller), and the risk of malicious data injection (implementing a trustworthiness source system that can guarantee the provenance of data).

Before describing the details of our proposed architecture, we provide a brief account on gossip-based networks, whose principles are at the basis of our overlay

network. Gossip communication is a style of computer-to-computer communication protocol inspired by the form of gossip seen in social networks. Modern, large-scale distributed systems often use gossip protocols to solve problems that might be difficult to solve in other ways [97], e.g., because the underlying network either has an inconvenient structure or is extremely large. Computer systems typically implement this type of protocol with a form of random "peer selection": with a given frequency, each machine picks another machine at random and shares any hot rumors.

In our gossip protocol, users choose to gossip some information anonymously to a local administrators. Although anonymous, gossips are signed by a ranking grade owned by each user. In turn, local administrators aggregate (in a weighted manner) the gossips they received regarding the same objects (roads, point of interest, etc.) and re-gossip that information to users. By aggregating data (received within a certain time span), administrators mitigate the diffusion of false information in the network, mediating (or ruling out) contrasting information.
In our gossip protocol, nodes disseminate knowledge of their surroundings — e.g., a user is on a route and notifies the status of that road. Moving nodes periodically probe the network to join the partition to which they geographically belong, following a background dissemination approach. This recalls anti-entropy approaches that focus on providing a system-wide consistent observation as aggregate of many local responses.
Thanks to the gossip protocol, we also anonymize the identity of users that query for information. In a traditional network, users would query a central server to retrieve some information, exposing their query (and themselves, by extension) to possible privacy attacks. On the contrary, in our setting it is the gossip protocol that is responsible for spreading information that might become relevant in the future to users. Hence we accept a trade-off of some overhead information to gain a privacy-by-design guarantee.

The concepts above are exemplified in Figure 5.7. From left to right, when users want to join for the first time or move between zones, they query known gateways to obtain the address of the administrator of the geographical region to which they currently belong. Administrators act as local zone authorities to route information

about the same zone, and they can also provide (as gossip) their trustworthy data to users. In the gossip phase, users disseminate to administrators information regarding their surroundings. In Figure 5.7, U1 and U3 declare to know something about the route A congestion (e.g., rough road, slow traffic), while U4 has some information about a specific incident on Route B (e.g., closed roadways). The remaining phase regards the inquiry of available data. Users periodically receive aggregate (and possibly enriched) gossips from the local administrator and query the knowledge they acquired through gossiping to extract relevant information.

Figure 5.8 shows the two planes, over the physical one, that characterize the proposed overlay network. From the physical plane, users join the middle Neighborhood plane where they generate new gossip (in the Figure, the thicker the arrow, the more trustable the peer). On top, we find the Inquiry plane, where gossip spreads and where users inquiry their acquired knowledge.

A real world use case application for this overlay network can be a rough-road check. A service that exposes the real-time information about the road conditions is typically based on an algorithm that calculates an estimate, considering some previous information and the GPS position of the vehicle. This information is then saved on a centralized storage where the delay is calculated. As described above, however, this methodology introduces security problems on the storage of data and quality of service, entailed by relying exclusively on the users GPS location, which could possibly be inaccurate. With the proposed approach, we show how it is possible to improve the safety and performance of this type of service.

In Figure 5.8, U1 (Inquiry Plane) has been informed of the condition of the route it is currently following, so it could decide to perform a route deviation. This happened because it joined its local Neighborhood network, where other users (the blue and yellow cars that are further ahead on the route) have been gossiping ① (Neighborhood Plane) about its same route. In its turn, the administrator collected the gossips, aggregated them, and re-gossiped the information to the users ②.

Although users U2 and U3 remain anonymous, U1 can rate the quality of the aggregated information; a metric that can feed the system to assign the degree of trustability to users, as specified in the previous paragraph[7]. Besides security, the

[7] Assignment of rankings to users can be spread back through gossip.

system enjoys a finer grade of precision on the reporting of the information as the machine-generated data from the administrator is integrated with user-generated, close-to-the-source information.

5.6.2 Automatic Business Policies and Contracts Enforcement

In 5.5.2 we detailed how the platooning federation relies on the collaboration between operators, and how this gives the opportunity to malicious agents to adopt damaging behaviors. To prevent this, we want to introduce a mechanism to enforce the respect of the agreed behavior in the federation.

This is a common problem in all kinds of Service Oriented Architectures (SOA), as explicitly addressed in OASIS's SOA Reference Model [117]. According to OASIS definitions, policies define constraints for single services, and our approach is to let each provider and each consumer of services deal with data-related policies without forcing a centralization which could become the most valuable target for attacks. Contracts, on the other hand, "represent an agreement between two or more participants [...] a service contract is a measurable assertion that governs the requirements and expectations of two or more parties. Unlike policy enforcement, which is usually the responsibility of the policy owner, contract enforcement may involve resolving disputes between the parties to the contract. The resolution of such disputes may involve appeals to higher authorities. Like policies, contracts may be expressed in a form that permits automated interpretation."

Our proposal is to let that higher authority be represented by the platform enabling the federation. Rather than merely acting as an enabler, the information system supporting the federation shall connect each member based on the behavior that each operator declared and agreed to respect to join the federation. The support encompasses automated checking of contracts for compliance to general principles at design and deploy time, and the enforcement of contract provisions at run time. It is a part of a more comprehensive strategy for SOA management, as outlined in [169].

This shift moves federation from a static coalition of companies into a federated market, where operators dynamically partner with each other and trade and use services of one another on-demand, knowing that the agreed terms of usage will be respected. As remarked, this last guarantee is enforced by the platform itself, which

will monitor the flow of requests and responses among the members, detect possible behaviors that deviate from the declared policy and limit or block the cooperation with an offending member.

In practice, contract specification is done when the platoon operator wants to join a federation. In this sense, enabling platforms shall provide respectively:

- a formal model with which to interpret policies and contracts;
- a set of specific requirements to satisfy in order to guarantee a safe environment;
- a set of policies regarding the use and the misuse of the services exposed;

In this context, business policies can have different purposes. For example, they can establish rules to mechanize the trading of the services of federated members, as well as to establish standards used in their interaction, like security and communication protocols, and define the terms of the quality of service. Automated policy processing allows automated contract design, to enable dynamic participation to the federation activities. In turn, contract enforcement at run time ensures all parties that everyone is held accountable against the designed contracts.
Beside security, we underline how these business policies are particularly important in the context of platooning operators, where each of them controls a specific, restricted area, while the platooning plan of a convoy usually spans areas controlled by many operators. With automatized business policies platooning operators can collaborate in synthesizing a common platooning plan, possibly comprising a mixed set of their users.

A prototypical example of one such platform has been made in our work in [32], where we employ machine-processable business policies to federate remote instances in a federated platform and where transport operators that agree to the same (or better, to compatible) business policies can partner with each other and trade each other's transport services.

5.7 Conclusions

In this chapter, we presented the concept of Mobility as a Service and how MaaS operators shall facilitate the dynamic provisioning of multi-modal transportation to their users. To support such flexibility we developed a federated marketplace of services called SMAll, aimed at harmonizing data flows and service invocations.

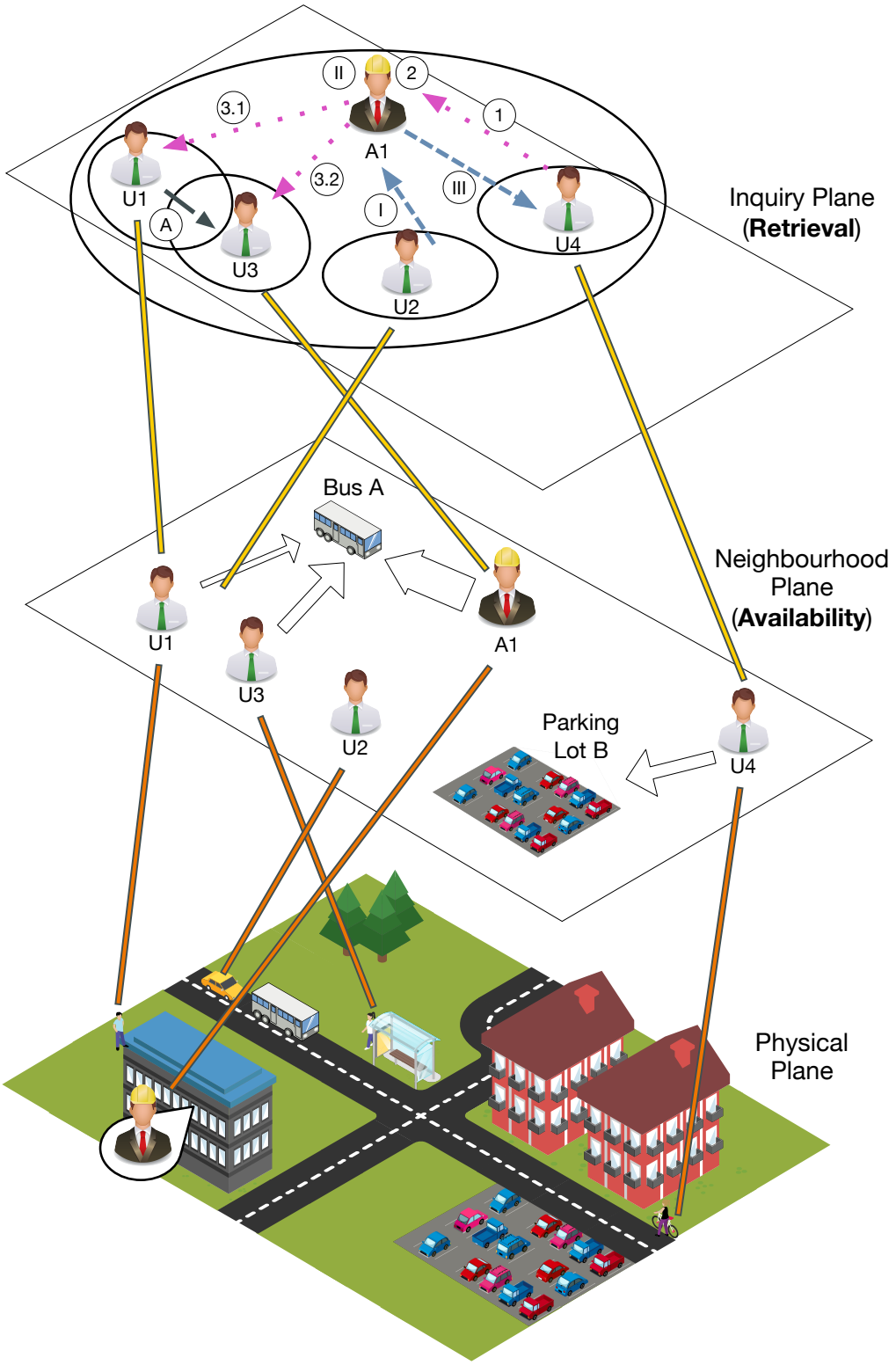
This kind of federated platform is particularly sensitive to insider threats, which emerge at different layers, targeting both the constituent components provided by users and operators and the services provided by the platform itself.

The MaaS Stack, our tiered proposed view on the components of MaaS markets, allowed us to treat in isolation the security issues of each tier. Often, these issues turn out to be instances of well-known threats in the fields of cloud computing, service-oriented architectures, supply chain management, and trusted business partnerships.

In principle, the platform allows to implement context specific versions of the solutions proposed in the literature regarding the aforementioned fields, as well as novel solutions inspired by their cross-fertilization. We argue that the central role of SMAll in mediating every interaction and in collecting their traces makes the platform fit to host solutions to the presented security issues of MaaS markets.

In addition to our general treatment on insider threats related to the context of MaaS markets, we also show-cased two real world case scenarios; the perspective of the Cloud of Things, presenting an architecture that constraints the quality and quantity of data that an insider could obtain from users, also optimizing the routing of requests to only those users able to answer them; and federated platooning, i.e., a freight organization system where a consortium of platooning operators collaborate and coordinate their users to constitute freights.

From our threat analysis, we detailed novel technical solutions to the predominant threats of trustworthiness of data flows and deviant behaviors of federation members. In particular we showed a over-layered architecture able to manage the real data flow in a safe way, and a policy enforcement methodology to prevent malicious and fraudulent behaviors and the service platform.


II
2
3.1
1
A1
U1
3.2
III
I
A
U3
U4
U2
Inquiry Plane
(Retrieval)
Bus A
U1
A1
U3
U2
Parking
Lot B
U4
Neighbourhood
Plane
(Availability)
Physical
Plane
Legend
Trustability
wrt object
Covered
Reachable
Area
U1
User
Inquiry Direction
A1
Administrator
1
I
A
Inquiry Steps

Chapter 6

TechNETium: a SDN tool to Verify Security Network Policies

The upcoming 5G promises to the next big thing in networking, making networks able to adapt to the specific requirements of vertical applications; while the most common examples consider machine-to-machine applications such as autonomous driving and massive IoT, social applications will be significantly affected by the improved interactivity and capacity, e.g. enabling immersive real-time gaming and broadband multimedia communications.
These improvements are achieved through the concept of network slicing, according to which a single network infrastructure may be partitioned in virtual sub-networks tailored to the specific requirement of a vertical application, thus achieving an optimal tradeoff between cost (shared) and performance (dedicated).
Network slicing requires a very high level of flexibility of the network infrastructure, that is supported by emerging technologies such as the intensive exploitation of Network Function Virtualization (NFV) and the centralized control of the network forwarding functions, i.e. Software Defined Networking (SDN).
These technologies are a real innovation in networking and allow innovative approaches to the solution of many problems.

6.0.1 Contributions

This chapter focuses on the network security issues, that are indeed very relevant to the aforementioned vertical applications. It shows how the formal verification of the networking forwarding rules, which can be achieved by interacting with the SDN

network controller may be used to detect possible security issues in the network data path.

Formal network verification requires a formal representation of the network and a software architecture working together to efficiently verify the policies and properties of the network. At the state of the art, there are a few tools and techniques to achieve this result. An example of a software package which can verify formal policies is Netplumber; it can verify many different properties, but the complexity of the underlying methodology (header space analysis) causes important performance issues. Using techniques such as Atomic Predicates, the fundamental property of reachability can instead be verified in microseconds, i.e. orders of magnitude more efficiently than with header space analysis. The main drawback of this approach is the lack of native expressiveness for policies other than reachability.

The contributions given in this chapter focus on the development of an SDN-based architecture able to meet and solve this challenges. We called this architecture TechNETium: an original software platform which exploit atomic predicates and binary decision diagrams to verify network reachability. TechNETium can be used to express complex policies by compiling high-level rules into compositions of reachability verification, enabling the creation of auxiliary policies such as FullReachability (the verification of bilateral communication between nodes) and ToWayPoint (used to enforce traffic flows trough a specific intermediate node), which is something new, to the best of our knowledge in the literature.

This chapter is organized as follows: the principles behind TechNETium are discussed in section 6.2, and a use case is presented on a SDN network based on the ONOS controller in section 6.3. Then we presented results showing a significant performance improvement when compared with other tools in the literature in section 6.2.4, in addition to several possible use case example of such architecture in section 6.4.

6.1 Context Scenario

6.1.1 SDN and formal verification

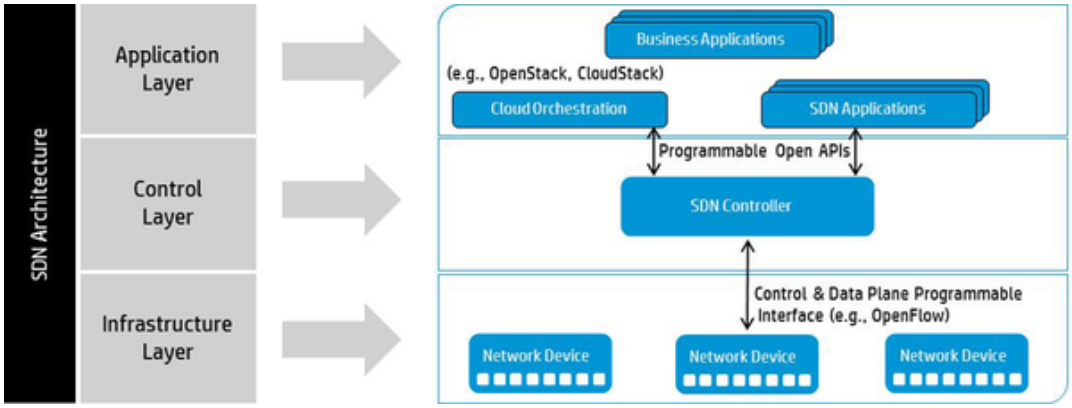


FIGURE 6.1: SDN Layer Architecture

The SDN architecture can be seen as a set of different levels (each with specific tasks) and interfaces for communication between levels. Figure 6.1 shows the 3 levels and API point representation of a common SDN Architecture:

On the Infrastructure layer we can find the forwarding devices (hardware or software) that can be used. There are different standards for this level, but the most common is certainly OpenFlow, which defines specifications for devices (called OpenFlow switches) which maintain a set of tables designed to filter packets based on some fields.

The Southbound API is the interface that allows communication between the Infrastructure layer and the Control layer. This API are a crucial tool for SDN, although they are strongly linked to underlying elements. It defines a series of event messages for communication between switches and controller, in addition to generating statistics generated by the switches and collected by the controller.

The Control layer represents the brain of the network. Here are one or more controllers (or NOS), which provide a global view of the network, essential services, and common APIs for developers. Thanks to this level, applications do not have to deal with network implementation details. There are different types of controllers, and one of the criteria for which they can be distinguished from one another is whether or not they are distributed.

A distributed controller can also handle networks a lot articulated. In addition, distributed controllers must address the issue of data consistency, and often this is guaranteed only in relative time long. One of the main advantages of distribution, however, is certainly the resistance to failures. If a node fails, replication makes it not a problem.

Centralized controller : a centralized controller has full network control, but it represents a single point of failure, and may not be able to to set up complex networks with a large number of nodes. Often centralized controllers take advantage of multi-threading for better performance. Another criterion for which two controllers can be differentiated is the implementation model policy implementation:

Reactive: a reactive controller requires the forwarding devices to consult the controller whenever a decision is needed. This may turn out to be a problem, above all if the flows arriving to the network are short (hence of for each new flow the switch must call the controller), or it would lead to scalability problems in the case of large networks.

Proactive: a protective approach ensures that policies are implemented at the beginning, without the explicit request of the switches. In addition to communication through the southbound and northbound APIs, the controllers, in the in case there are more than one in the network, they communicate with each other through APIs dedicated, called westbound eastbound APIs.

Northbound API This level allows external applications to use the controller's core services to configure, monitor, and manage the network. Unlike the southbound interface, there is no common standard, but in general we use REST API, or languages of specific programming such as Frenetic, and NetCore.

Application layer Here are all the applications that interact with the network. Thanks to the abstraction provided by the control layer, it is very easy to develop and implement new ones software in SDN. Applications communicate the controller with the northbound APIs, and not therefore they need to know the implementation details of the network.

6.1.2 Formal Data Plane Verification

In recent years we have seen a huge development of security tools based on formal verification in various use case scenarios. In particular network verification gained a lot of interest [15, 89, 2, 132]. It is built around an automated detection of violations of network reachability invariants on the data plane, possibly while they are happening in real-time.

A recent work by Beckett et al. [15] proposes a tool called Minesweeper following the study of several tools developed by the research community with the goal of finding network misconfigurations. The studies are classified in two categories: control plane oriented, i.e., able to discover flawed configurations proactively, and data plane oriented, i.e., able to discover misconfigurations reactively, by observing the events happening in the network while traffic is flowing. Both categories have pros and cons. Proactive approaches are particularly useful to predict potential network misconfigurations that might lead to security issue (e.g., BGP prefix hijacking). But security breaches in the most general sense are hard to predict a priori, therefore, the reactive approach is essential to detect unwanted (dynamic) network behavior occurring as a consequence of malicious activity.

Following the same line, we argue that both proactive and reactive approaches should be adopted and combined to exploit synergies between them at best. *The SDN paradigms provides the perfect platform to pursue this goal, because in a SDN network what happens in the data plane is the outcome of what is produced in the control plane but, at the same time, the control plane has a view of what is happening in the data plane.*

6.1.3 SDN and formal verification

The spread of the SDN has sparked a debate about the security of *software defined* solutions. Here we will not address the whole of this large subject, concentrating our attention on some issues regarding the correctness of topology and routing.

Studies like [3, 108] have analyzed the attack surface highlighting the vulnerabilities and security requirements of the architecture. In [3] it is suggested the combined adoption of *proactive* and *reactive* approaches, however works like [126] emphasize the importance of reactive solutions for verification of invariants in the data plane:

the problem consists in checking if certain security policies (*policy*) are respected starting from the *Forwarding Information Base*, that is from the rules of *forwarding* and from the topology of the network.

This is a complex problem, since it requires analysing the behavior of all possible packet headers. The *header space*, as it is called in [104], can be very large (the IP header being at least 160 bit long), therefore, since the union and intersection of packets are computationally expensive operations (complexity (O (2^n)) in the worst case, where n is the number of bits in the header [201]), verification of properties in a network with thousands of devices can take a very long time.

To improve performance, Yang and Lam in [201] propose to reduce the space to be explored: instead of working on individual headers, thy use sets of packets which are equivalent from the viewpoint of forwarding, called *classes of equivalence* or *Forwarding Equivalence Class (FEC)*, and a symbolic representation is adopted to reduce computational complexity.

Experiments conducted by the same researchers, as well as other research groups [20], support their idea. The code of these experiments is available at [203].

6.1.4 The Atomic Predicates Verifier

As it can be seen from [201, 204, 205] AP Verifier is not specifically designed for *software defined* networks, however works such as [194, 94, 114] show that this methodology is also applicable in the of the SDNs.
The methodology is explained below by analyzing the following steps:

1. Definition of a model of the network suitable for verification.
2. Definition of atomic predicates.
3. Algorithm for calculating atomic predicates
4. Check the network using atomic predicates.
5. System update in real time.

In step 1 the model of the network adopted in the system is defined, in steps 2 and 3 the theoretical bases necessary to carry out the verification and updates referred to in steps 4 and 5 are set.

Specifically, we get to show that each network predicate can be uniquely represented as the logical dis-junction of a set of atomic predicates; moreover, since each element has an integer identifier, it is possible to represent the state of the network by a set of integers, thus obtaining a symbolic representation.

In this new space, indicated by [94] like *quotient space*, the operations of union and intersection of headers map to operations of union and intersection of sets of integers, thus improving temporal performances with respect to [104], which uses *ternary bit vectors* instead.

The network is modeled as a direct graph of devices, so the links are unidirectional.

Each device has a set of full duplex physical ports and a list of rules applied according to the *longest prefix match* criterion. Inside a device the ports are all connected together.

There are two types of rules:

- **Access Control Lists** (ACL), consisting of a predicate of match and action.
- **Forwarding Rules** or *forwarding* consist of a match predicate, based on a prefix and its length, and an output port.

The boolean functions, or predicates, used for the *match* of the packets are represented by *Binary Decision Diagrams (BDD)*, an acyclic direct graph data structure whose nodes represent Boolean variables; each node of a BDD has an *low child* and an *high child* and the arcs linked to child nodes represent the assignment of the false and true value to the node variable, respectively. The evaluation of a predicate proceeds recursively by choosing the child up or down based on the value of the variable, up to the result of the evaluation reaching a terminal node *true* or *false* [1].

The choice of BDD is the result of a comparative analysis carried out in [204], which takes into consideration three other data structures: *packet sets* [65], *Firewall Decision Diagrams (FDD)* [78] and finally the *wildcard expressions* [104]; the results of the study show that BDDs enjoy the following properties:

1. **Uniqueness of representation**: the representation of a set of packages with BDD or FDD is unique, while there are multiple *wildcard expression* and *packet set* for the same set.

2. **efficiency in compute calculation**: the computation of the BDD and FDD complement is very simple because it is enough to exchange terminal nodes of the graph, while for the other data structures one could have an increase in size.

3. **small size**: to represent a set of packages identified by a header suffix, the FDDs and emphs packet sets require structures whose dimensions are exponential with respect to the suffix length. It is shown instead that the BDD representation of an ACL rule has at most (2 + 2h) nodes where (h) is the length of the headers, while for a forwarding rule we have at most (2 + n nodes with (n) equal to the number of bits in an IP. The linear dependency of the dimensions from the length of the headers makes memory consumption and computational complexity manageable, since the logical conjunction and disjunction requires a time proportional to the size of the BDD [204].

In algorithm [201] the algorithms used to extract BDDs from ACLs and forwarding tables are illustrated. In the following section, the theory of atomic predicates is explained in more detail, and some improvements for its application to network verification are proposed.

6.2 Atomic Predicates for Transformations

6.2.1 Definition of Atomic Predicates

The fundamental idea of the method of atomic predicates consists in the representation of port predicates through the equivalence classes that are forwarded through the corresponding port. The representation with BDD is flanked by the set of equivalence classes that pass through the predicate gate. This is possible because within each device the packets belonging to an FEC are forwarded on a single port, therefore the predicates are seen as disjoint sets of equivalence classes. At this point a further step is taken: the equivalence classes are uniquely associated with integer identifiers. It follows that the set of identifiers of the equivalence classes that pass through a port is equivalent to the original information of the predicate and can be used for calculating the properties of the network. The equivalence between these

representations is demonstrated in [201, 204, 205] and is indicated in [156] as *equivalence of Yang and Lam*.

In this way the disjunction and conjunction operations on the predicates are mapped on union and intersection operations on sets of integers, with a considerable reduction in complexity.

Definition 5. ***Atomic Predicates*** *Considered P as a set of Predicates, the corresponding Atomic Predicates set $\{a_1, a_2, \ldots, a_k\}$ follow this rules:*

1. *For each $i = 1...k$; $a_i \neq false$.*
2. *$\bigvee_{i=1}^{k} a_i = true$.*
3. *$a_i \wedge a_j = false$ se $i \neq j$.*
4. *Every predicates $p \in P$ is the results of logical disjunction of a unique set of atomic predicates:*

$$p = \bigvee_{i \in S(p)} a_i, \text{ where } S(p) \subseteq \{1,...,k\} \tag{6.1}$$

   This implies that p can be represented as $S(p)$ and there a direct correspondence from disjunction and conjunction operator p_1 awn p_2 and the union and intersection operation $S(p1)$ and $S(p2)$.

5. *k is the minimum natural integer $\{p_1, p_2, ..., p_k\}$ that follow those 4 conditions.*

Theorem 1. ***Atomic Predicates and Equivalence Classes***

It has been proven [201] that in a given a set of Predicates P , the P Atomic Predicates set is the minimum set of equivalence class of every packet.

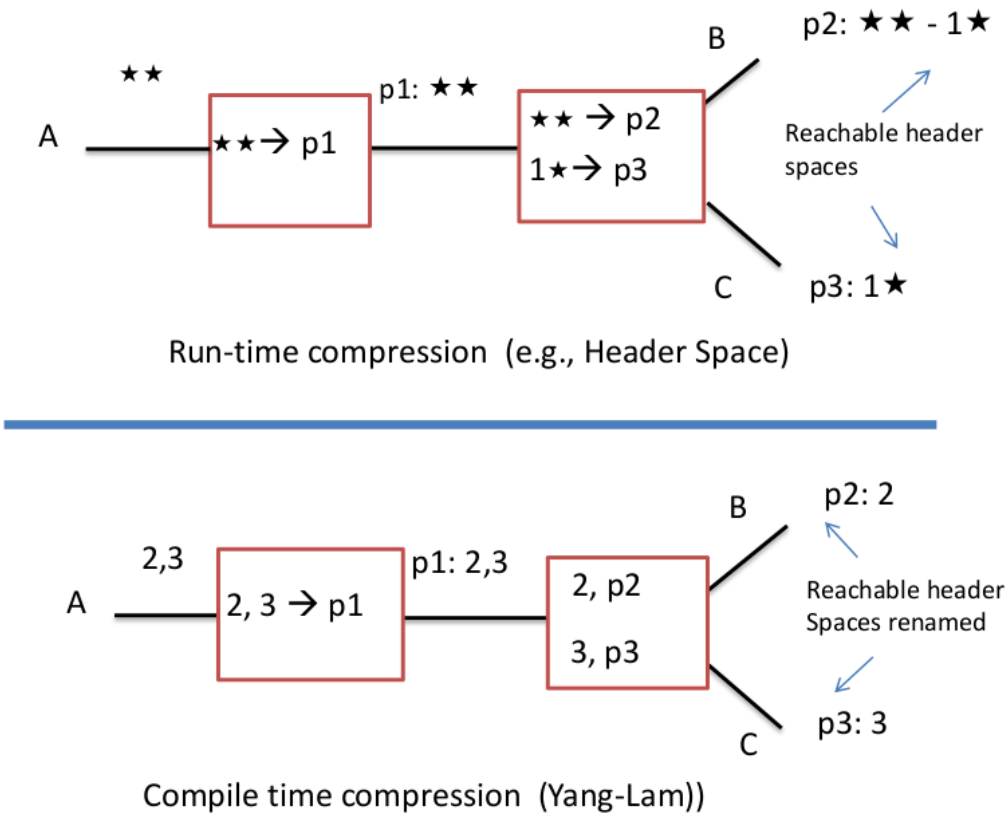


FIGURE 6.2: Atomic predicates Verifier - Differences between header space and quotient space

In the example shown in Figure 6.2, taken from [20], we illustrate the differences between atomic predicates and analysis in *header space*:
For simplicity, suppose we have 2-bit long headers. In the image we have two routers that adopt longest match semantics: in the first one there is only one rule that involves the forwarding of all the input (wildcard expression (**)) of the p0 port on the p1 port, in the second router is specified that the traffic with prefix (1 *) must be forwarded to port 3, the rest of the traffic must flow to the p2 port instead.

Using NetPlumber [105] we would have 3 *rule nodes*, one for each rule, an *intra-table dependency* in the second router table and an *pipe filter* for port 2 of the type: (** - 1 *). Furthermore, the difference between the two prefixes, which in this simple case is (0 *), must be computed at runtime, requiring an expensive operation of intersection between *wildcard expressions*.

We will now evaluate the verification with atomic predicates. First, the port predicates are calculated from the forwarding rules, then the FECs that are only two are calculated: the set of packages that are forwarded to p2 ((** - 1 *), or (0 *)), labeled with " 2 "and the set of packets forwarded to p3, labeled with"3" . After that each port predicate is represented as the union of the FECs:

1. $** = \{2,3\}$
2. $1* = \{3\}$
3. $**-1* = \{2\}$

As again pointed out by [20] and by the tables in [201, 204] the approach to atomic predicates is in most cases 1 or 2 orders of magnitude faster than header space analysis in calculating reachability, which has been our choice for the data plane verification tool.

6.2.2 Calculation of atomic predicates

Once demonstrated that atomic predicates correspond to equivalence classes, we need a way to extrapolate them quickly from a set of port predicates. For this purpose we need the following two formulas and the algorithm inspired by [201].

Theorem 2. ***Atomic Predicates, Cardinality Set 1*** *Given a Predicate p we denote $A(\{p\})$ as the set of atomic predicates of the set compose only by the predicate p and:*

$$A(\{p\}) = \begin{cases} \{true\} & se\ p = true\ or\ false \\ \{p, \neg p\} & else \end{cases} \tag{6.2}$$

Theorem 3. ***Atomic Predicates of the union of two sets of predicates*** *Given $P_1 = \{b_1, b_2, \ldots, b_m\}$ e $P_2 = \{c_1, c_2, \ldots, c_n\}$ 2 Atomic Predicates set. $AP = \{a_1, a_2, \ldots, a_k\}$ is the result of:*

$$\{a_i = b_{i_1} \wedge c_{i_2} | a_i \neq false, i_1 = \{1,...,m\}, i_2 = \{1,\ldots,n\}\} \tag{6.3}$$

$AP = A(\{P_1 \cup P_2\})$ in [201].

The following algorithm applies the given formulas given to calculate the atomic predicates of a set of one element and the union of two sets to obtain the atomic predicates of an arbitrary set of predicates.

Algorithm 1 Atomic Predicates calculation algorithm

Input: set of predicates $\{p_1, p_2, \ldots, p_N\}$

Output: atomic predicates input set, $A(\{p_1, p_2, \ldots, p_N\})$

1: **for** $i = 1$ to N **do**
2: Compute $A(\{p_i\})$ using (6.2)
3: **end for**
4: **for** $i = 2$ to N **do**
5: Compute $A(\{p_1, ..., p_i\})$ using (6.3) with $A(\{p_1, ..., p_{i-1}\})$ e $A(\{p_i\})$
6: **end for**
7: **return** $A(\{p_1, \ldots, p_N\})$;

Note that it is possible to apply the formulas (6.2) and (6.3) within a single cycle. The system maintains two distinct sets of predicates: one for ACLs, one for forwarding rules. Each of these two sets can be calculated through the algorithm following two criteria that provide different performances.

1. criteria for sorting of the MBF ACL:
   - **Random selection**: ACLs are processed in random order.
   - **Smallest ACL first**: first select the ACLs with a smaller number of rules.
2. Criteria for ordering the **forwarding rules**:
   - **Random selection**: the rules are processed in random order.
   - **Selection by box**: the rules related to the same are processed together *middlebox*.

loop and black holes detection

As already seen, the loops are detected during the construction of the policy tree. We consider exclusively the *packet-drops* due to the absence of forwarding rules in the routers, not the ACL that explicitly discard the traffic. If the difference between the set of all the atomic predicates and the union of all the atomic predicates of a forwarding table is empty, then the table has no black holes, because it means that all the traffic received is treated. Or:

$$S(true) - \bigcup_{i=1}^{k} S(F_i) = \emptyset \implies \text{No black holes in the table} \tag{6.4}$$

Where $S(true)$ is the set of every atomic predicates and k is the number of port indicates on the table.

Network slicing

A *slice* network is a set of ports and a set of forwarding and access control rules. Given two *slice* $Slice_1$ and $Slice_2$ we consider T_1 and T_2 the sets of gates, S_{F_1} and S_{F_2} the sets of forwarding rules and S_{A_1} and S_{A_2} the sets of the ACL rules of the two *slice*; these overlap if and only if:

$$T_1 \cap T_2 \neq \emptyset \vee S_{A_1} \cap S_{A_2} \neq \emptyset \vee S_{F_1} \cap S_{F_2} \neq \emptyset \tag{6.5}$$

6.2.3 Graph updates

Updating links

Alterations in the state of the topology do not modify the atomic predicates, however the police tree could vary and an update is therefore necessary:

- **Link down**:retrieve the nodes from the policy tree hashtable and delete the entire subtree from the table from the link.
- **Link up**: retrieve the nodes from the policy tree hashtable and extend the tree by searching *depth first* starting from the port involved.

Update of rules

When the rules are updated, three operations must be performed:

1. Verifies that the port predicates are altered and possible recalculation.
2. Updating atomic predicates.
3. Updating the reachability tree item.

Operations 2 and 3 can be carried out concurrently.

Update of port predicates

To make updating possible, the rules are organized in a forest. Each tree is constructed by iterating over the forwarding table of a device and setting the rules whose prefix is contained in the current prefix as daughters of the current rule. In this way a set of tuples is saved in each tree:

$$< Prefix_i, Length_i, Port_i, R_i >$$

Where $Prefix$, $Length$ are used for the match, $Port$ is the output interface, while R_i is given by the predicate obtained by eliminating the rules prefixes daughters from $Prefix_i$ according to the following formula:

$$R_i = Prefix_i \wedge (\bigvee_{j \in C(i)} \neg Prefix_j) \tag{6.6}$$

Where $C(i)$ is the set of child rules of the rule (i).

The predicate of the port x, P_x, is obtained then as the disjunction of the predicates R_i of the rules that direct the forwarding on it:

$$P_x = \bigvee_{port_i = port_x} R_i \tag{6.7}$$

Let's see how the tree is used in the event of rule updates:

1. **rule deletion**: if the rule i is deleted, its children are assigned to the parent rule j, and the tree is updated by replacing it in R_j the predicate of the removed rule i.

   $$R_j \leftarrow R_j \vee R_i$$

   If the rule i forwards packets to the port x and the parent rule j forwards to the port y, it must be removed from the port predicate P_x. The predicate of the

deleted rule R_i and add it instead to P_y:

$$P_x \leftarrow P_x \wedge \neg R_i$$

$$P_y \leftarrow P_y \vee R_i$$

2. **Adding a rule**: if a rule is added i, it is first added to the reference tree according to the criteria seen above. If the rule j is the parent of i, the match predicate of the new rule must be removed from the parent rule:

$$R_j \leftarrow R_j \wedge \neg Prefix_i$$

The predicates of port are then modified in a dual manner to the previous case:

$$P_x \leftarrow P_x \vee R_i$$

$$P_y \leftarrow P_y \wedge \neg R_i$$

Updating of atomic predicates

There are of course two cases:

1. **Addition of a predicate**: just apply the formula (6.3), to calculate the atomic predicates of the union of two sets.

2. **Deletion of a predicate** P_j: the old set of atomic predicates is still representative of the set of port predicates, but some predicates could be redundant.

To minimize the set of atomic predicates we consider all the atomic predicates that represented P_j: of these the ones that are not used in the description of other gate predicates are eliminated. The complete algorithm is described in [204].

6.2.4 Improvements Introduced

With the update algorithm we are able to maintain a temporary tree in the event of a rule update. Reachability queries thus can be answered anytime without waiting for the atomic predicates to be updated. Updating, taking 10 ms on average in our

test-beds, occurs concurrently. The temporary tree is expanded with the new unresolved predicates, causing a slight decrease in efficiency as for [106].

If the unsolved predicates are in a number less than a certain threshold (configurable) and there are no changes of atomic predicates it is sufficient to calculate the AP representation of the new predicates. Vice versa if the number of unresolved predicates exceeds the threshold or if the set of atomic predicates has changed, the temporary tree is eliminated and a new one is calculated.

Essential for the functioning of the system is the concept of *forwarding equivalence class*, already applied in [106] and cited in [105] for the implementation of a distributed version of NetPlumber. The demonstration that the representation in atomic predicates is a representation of the network equivalent to the original one is equally fundamental. These two principles have inspired numerous further studies including [20, 99, 156, 194, 94, 114, 210, 88].

Some limitations emerge at this point. First of all, the network model excludes a priori any device that modifies packets, cutting out widespread cases such as NAT, MPLS, IP-in-IP tunnels. However, the theory has been extended to include some types of changes in [202], although, to the best of our knowledge, a version of this upgraded library is not yet available. Furthermore, as we have seen, the forwarding rules only use the IP address to construct the match predicates. A solution for the application of the method to the SDN consists in the use of the headers indicated in the OpenFlow specification [185], instead of just the IP address.

Finally, considering the BDDs, [20] suggests the use of a more efficient data structure for the representation of predicates, attributing to BDDs an overhead due to the excessive number of calls to library functions. It is also known from [1] that the BDDs are sensitive to the ordering of the variables: an incorrect ordering can lead to a sensible efficiency reduction of the system.

For these reasons, an important part of the work presented in this chapter has been focused on improving the current state of the art tools to attenuate or remove such limitations. Our contributions include:

1. **HashTable**: a "reverse" tree is kept in a hash-table for each reachability tree. This table correlates a port and the reachability tree nodes that refer to it, to

update trees more quickly.

2. **Ports along the route**: all common ports are kept in the path from s a d, to ease loop detection.

3. **Complementary set of atomic predicates**: if a set of atomic predicates, used to represent a predicate, becomes larger than half of $S(true)$, save the complement of the set with respect to $S(true)$ instead of the original set, to save space.

4. **OpenFlow Integration**: is it possible now to feed the BDD with OpenFlow rules taken from the SDN controller (as it is shown in the next section using ONOS).

5. **Multiple Match Key Forwarding Rule**: Thanks to the OpenFlow and ONOS integration we are also able to use three different type of matching keys for the forwarding rules: exact-ip, ternary, and long-prefix match.

6. **Improved domain atomic predicates to integer translation**: We rewrote the algorithm for the domain conversion from to integer tree implementing a cache that keep track of the last conversions.

6.3 TechNETium, a verification data plane tool

In this section we will show TechNETium our security policy checker tool.
A core security design of our tool is Model Driven Development (MDD) [52, 45, 11], an approach that adopts Unified Modeling Language (UML) as domain-specific modeling language. The goal of MDD is to model entities, relations and behaviors of the elements of the system as high-level abstractions.
In the Model Driven Development approach, the definition of the specific domain of the problem to be modeled is required. In TechNETium, this involves both security and networking verification tools. We herein summarize the general concepts behind the MDD implementation useful to understand our contributions; further details can be found in our work in [126].
. TechNETium exploits a network model-checking based on policies. By policy we

mean a variant aspect of any network (forwarding) mechanism, i.e., a desirable high-level goal expressed in the form of a rule or a configuration. We build a network model to formally check if those policies are met. The basic idea is to create and maintain a model of the network and a set of policies that such model must satisfy. Consequently, the model allows to continuously verify whether or not the policy set is satisfied on the network.

As specified in the previous section, we consider the SDN paradigm as the enabler for this technology, since it natively exhibits the separation between the data plane level and the control plane.

The aim of TechNETium is to be totally portable and architecture-independent. For this purpose, it is written in Java; however, to perform a first test we choose a specific architecture on which to deploy it: ONOS[16]. This choice was motivated by several factors. First of all, ONOS offers a large set of APIs and methods for managing the network, a necessary requirement to be able to quickly obtain data about network topology elements such as: devices, links, ports, hosts and forwarding rules.

ONOS is also one of the best open source controllers in terms of performance [16].

- High Throughput: up to 1M requests/second
- Low Latency: 10 - 100 ms event processing
- Global Network State Size: up to 1TB of data
- High Availability

ONOS has been also proven one of the best controller for DoS attack tolerance [4] compared to the main open source alternatives.

TechNETium is a architecture composed of two main modules that work on two levels of an SDN architecture stack. At the Data Plane level, through the predicates of a typical SDN controller, generates a representation of the network model using the previously described atomic predicate technique. The result of this graph will therefore represent a snapshot of all the possible paths of a data traffic divided according to the flow rules of the switches.

The Control Plane level module, on the other hand, will input a "high level" security policy defined and described by the user and / or network administrator, and will

perform the verification of this by computing it on the atomic predicate graph, in the form of Reachability and ToWayPoint queries.

6.3.1 Atomic Predicates BDD generator

The first part of TechNETium creates the atomic predicates trees. Following what has been described in section 6.2 according to the atomic predicates theory, we need two distinct sets of data to generate it:

- Physical data related to switches and hosts (links, ports, interfaces);
- All the installed flow rules.

From these two sets a PolicyGraph is then created. It represents the whole set of possible paths of a packet within the network, according to the currently installed flow rules. As underlined in 6.2 we have been inspired by the AP Verifier tool described in [205], but we introduced many improvements in terms of performance and efficiency.
The BBD generator has been integrated as a core service of the ONOS Engine. In this way we can easily query the extended classes of the Topology module of ONOS to query it and obtain the dataset of the network devices, links and hosts and the correspondent flow rules, to create the BDD tree of atomic predicates.

6.3.2 Policy Checker Application

Once the BDD tree has been created, it is possible to verify the security network policy. The strength of TechNETium is that a policy is simply seen as a property of the network. A property can then be formally verified on its representation as BDD. However, to make the whole process automatic and user-customizable, TechNETium can interpret the high-level policy defined by the user by breaking it down into elementary graph operations. The user is enabled to define a policy as desired properties regarding the routing of network traffic through the various nodes, while the policy verification is efficiently implemented as a sequence of Reachability checks. In our case study, the policy manager has been created as an ONOS application, with its own graphical interface allowing to insert the security policy and to

launch the verification.

To validate our approach we implemented the Reachability policy (obviously) and we provided two examples showing how to define composite policies based on Reachability, namely the FullReachability and ToWayPoint policies:

- FullReachability (A,B) holds true if there is Reachability from A to B AND from B to A
- ToWayPoint (A,B,C) holds true if every existing path from A to C must pass through B.

The Reachability policy can be verified as a property of the BDD tree. The atomic predicates representation allows us to calculate the set of packets that can reach a destination port starting from source port using the following algorithm:

Algorithm 2 Reachability algorithm from port s to port d

Input: $S(F_1),\ldots,S(F_j)$: quotient space representation of forwarding predicates among the path s and d $(F_1, F_2, ..., F_j)$

Input: $S(A_1),\ldots,S(A_k)$: quotient space representation of access control predicates among path between s and d $(A_1, A_2, ..., A_k)$

Output: set of packets that can effectively reach d from s

1: $S_F \leftarrow S_F \cap S(F_1) \cap \ldots \cap S(F_j)$
2: **if** $S_F = \emptyset$ **then**
3: return $false$
4: **end if**
5: $S_A \leftarrow S_A \cap S(A_1) \cap \ldots \cap S(A_k)$
6: **if** $S_A = \emptyset$ **then**
7: return $false$;
8: **end if**
9: **return** S_F, S_A

We can then use the verification algorithm to calculate the reachability tree of a given port by performing a depth-first search by injecting all packets (S_A and S_F) into the port. A search branch is interrupted at a port x if:

1. (S_A or S_F) becomes empty.

2. The port x is an output port and it is not connected to any input port.

3. The port x is the entry port of a box without exit ports. The port x has already been visited (*loop detected*).

In this example (taken from [201]) we have both forwarding and ACL rules, indicated for a port such as S_F and S_A or as two separate integer lists from ";" following "forwarding_ap ; ***acl_ap***".

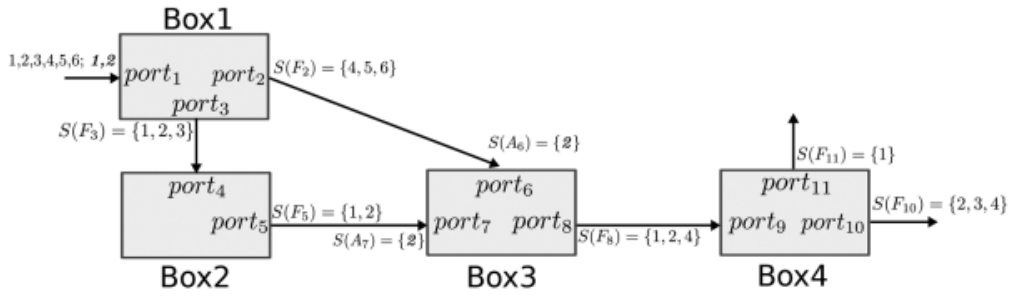


FIGURE 6.3: Atomic Predicates Verifier - Example reachability network

The network has six atomic predicates for forwarding and two for access control. The input port $port_1$ accepts all packets as input and forwards the predicates 4,5,6 to the port $port2$ and the predicates 1,2,3 to the port $port_3$.

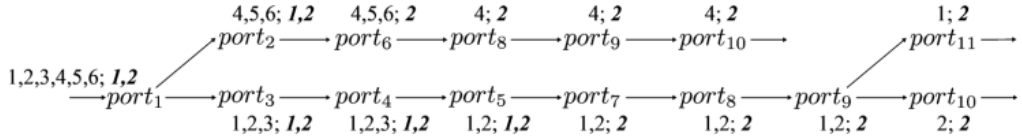


FIGURE 6.4: Atomic Predicates Verifier - Port reachability tree $port_1$ for the network shown in the figure (6.3)

The reachability tree saves in each node: a port, and the identifiers of the predicates that can reach the port (*forwarding*) and of the predicates that can pass through it (*ACL*).

In any case the *backtracking* is performed and the search is carried out until there are no more ports to analyze. At the end we get a tree whose nodes are given by a port number and the set of packets that can arrive there.

The FullReachability and the ToWayPoint policy as a consequence are decomposed

as high-level policies to a set of reachability ones. This because when a model checking tool is able to construct the reachability graph of a system, it can in principle answer any reachability question by simply examining this graph [27].
The FullReachability policy is just a simple example of this kind of decomposition. It express the Reachability from A to B in a duplex way, for this reason is it simply calculated as:

$$FullReachability(A,B) = Reachability(A,B) \wedge Reachability(B,A) \tag{6.8}$$

and no need for an additional reachability graph is necessary.
The ToWayPoint policy is more complex and it calls for a modification of the reachability graph. In order to be expressed in terms of Reachability, ToWayPoint is defined as follows. Considering:

$$\alpha \neq \beta, \omega \neq \alpha, \omega \neq \beta \tag{6.9}$$

The figure 6.5 shows an extract of the policy verification management interface. Once the policies are defined and installed it is possible to launch the verification.

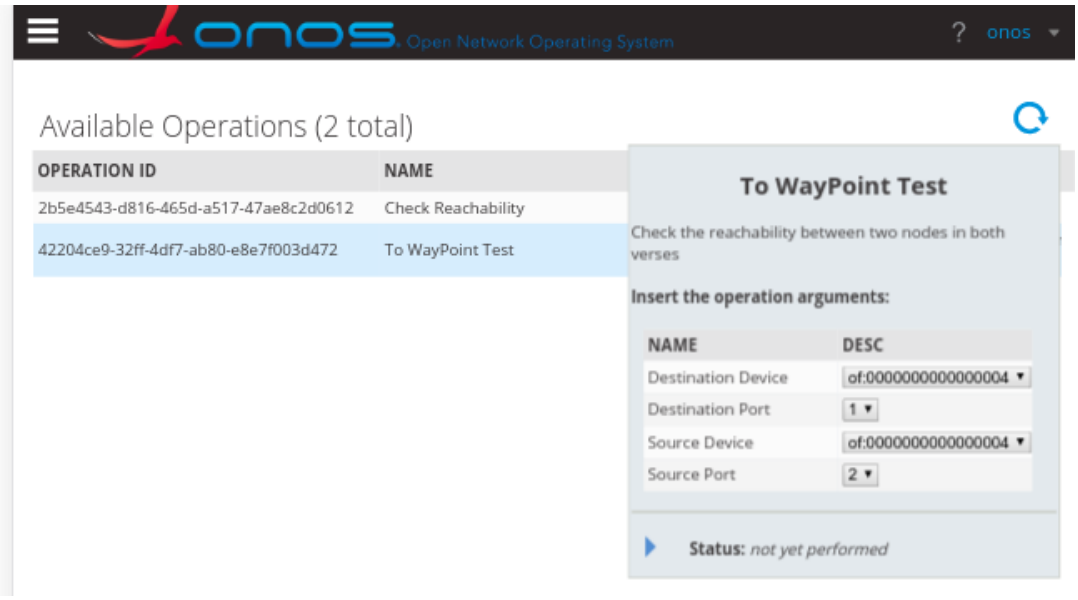


FIGURE 6.5: Policy Verifier ONOS Interface

These policies, however simple, can be considered the first step for a network administrator to define his security policies for network management. In fact, considering the most common network attacks such as distributed / reflection denial of

service, network administrators can use this policy in a proactive or reactive mode: *Reactively*, e.g. by isolating an entire network or sub-network or single node from which malicious traffic originates, and verifying their effective isolation with reachability policies.

Proactively, e.g. by redirecting a suspicious traffic flow on a specific subnet, or a honey-net, where NFVs can be installed with diagnostic / deep packet inspection tools to analyse the aforementioned traffic in more detail. An example of such use case scenario has been shown in our in work [125].

6.4 Test and Evaluation

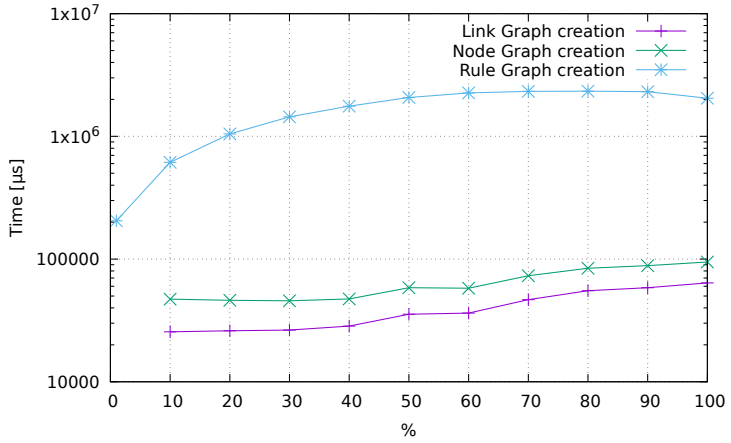


FIGURE 6.6: Average BDD creation time.

In this section we will describe some of the main experiments/test we performed. The goals of such tests were mainly 3:

- Showing the performances of the network graph creation.
- Showing the performance difference for reachability verification with related works such NetPlumber.
- Showing the performance difference for generic policy verification with related works such NetPlumber.

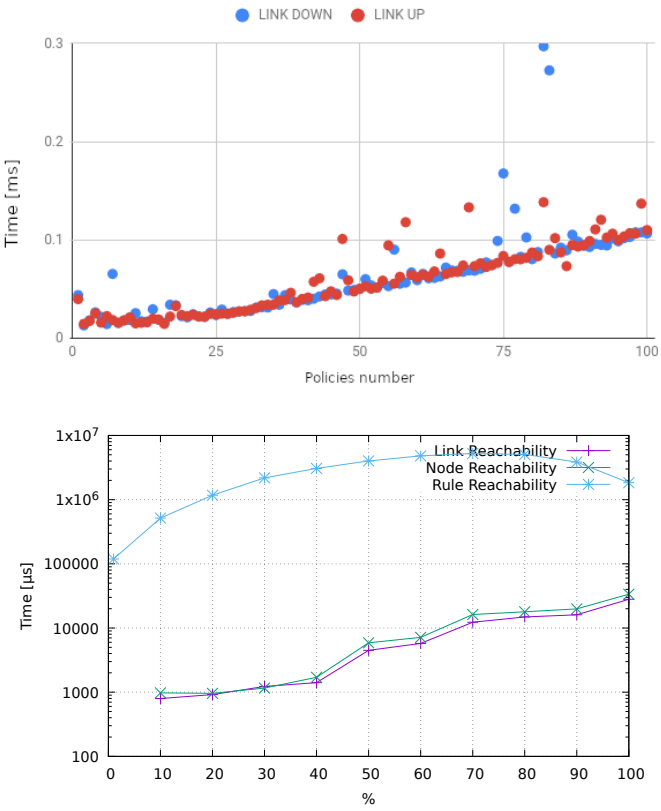


FIGURE 6.7: Policy time check after a link up / link down upgrade and node, links and forwarding rules increase.

The experiments were performed on several virtualbox dual-core virtual machines with 4 GB of Ram and Ubuntu Linux 18.04 operating system. The network sample that we used is the i2 Standford Network[1].

This network is composed of 161 Nodes, 11450 Links, and 77451 Forwarding Rules.

The first set of tests has been made to verify the performances on the BDD graph creation. We executed TechNETium increasing the amount of 3 different items: links, nodes and rules. As a result of these numbers, we have therefore measured the creation time of the BDD fee.

On graphs 6.6 we can see the required time for TechNETium to create the BDD in 3 different ways. Increasing the number of nodes, of links, and of forwarding rules given as a incremental percentage of the total network. As we predicted, the growth of creation time is roughly linear in the number of links and nodes.

Considering the forwarding rules, instead, the time increases linearly up to 60% of

[1]https://uit.stanford.edu/service/network/internet2

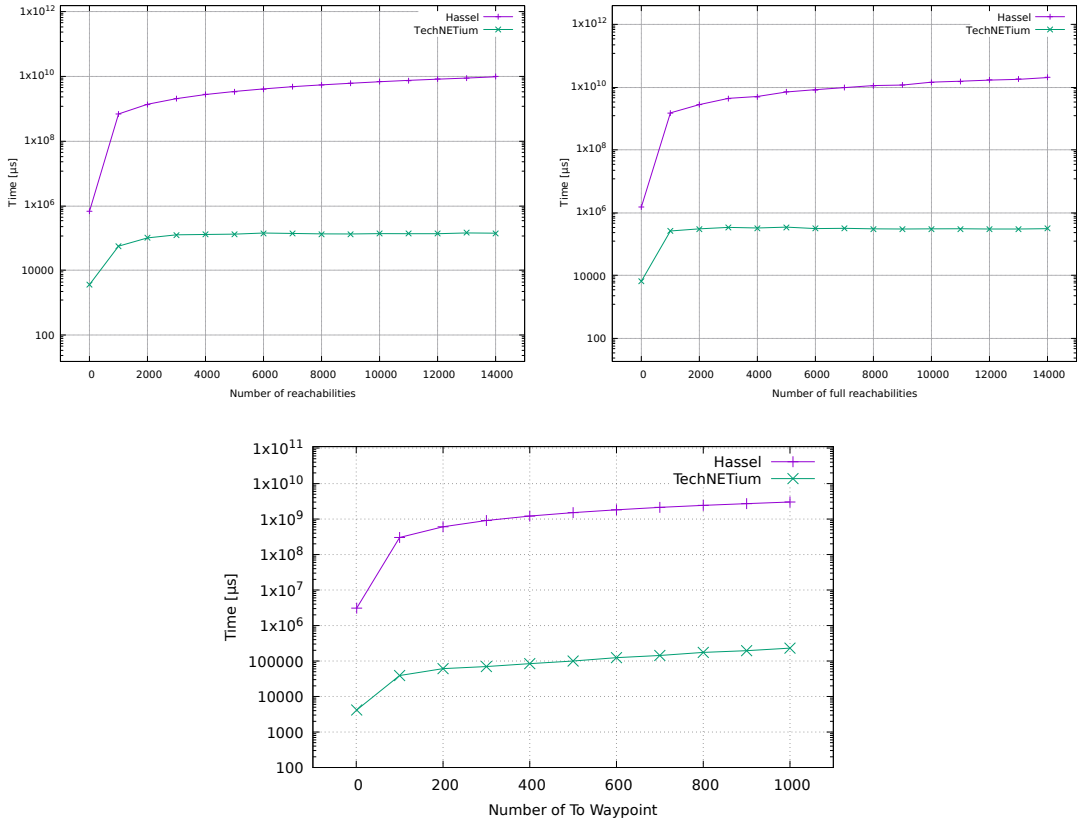


FIGURE 6.8: Policy time check after a link up / link down upgrade and node, links and forwarding rules increase.

the rules, then it stabilizes. This is due to the fact that we increase the forwarding rules shuffling them at every step but in a equivalent number for each device. For this reason, after a certain threshold the complexity of the network due to the calculation of all possible paths of a packet becomes constant.

The second set of tests that we performed aimed to calculate the policy verification time while upgrading the reachability graph. We calculated the time required to verify the same policy changing the network graph as we previously did it, increasing the number of nodes, links and forwarding rules.

Graphs 6.7 shows the results of such tests. The left graph shows the time to verify an increasing number of Reachability policies when adding or removing one link. These events modify the reachability graph, yet the verification time still increases linearly with the number of policies only. This is a relevant test that demonstrates that the atomic predicates techniques implemented by TechNETium is efficient with respect to graph updates, which is one of the main requirements for this kind of tools.

The graph on the right confirms this result generalizing the test approach. The verification time linearly increases with the cardinalities of the sets of nodes, links, and forwarding rules. There is only an anomalous behavior after the inclusion of the 70% of forwarding rules set. In this case the behavior is due to the fact that, as the forwarding rules increase, many more direct links are configured so that the reachability is much more likely to be successful in less time.

Finally we tested the performance difference between the NetPlumber tool (based on Hassel header space analysis) and TechNETium with atomic predicates. Those tests aims to proof that the BDD based policy verification on a large scale is definitely more efficient. For this reason we tested the three developed policies (Reachability, FullReachability and ToWayPoint) increasing the policies number between 1 and 1000.

From graphs 6.8 we can therefore infer some important data. Policy verification with TechNETium is 3 orders of magnitude more efficient. The growth is also linear for both, therefore the performance difference grows with the number of policies to be verified. Furthermore, linear growth is different. While NetPlumber shows a growth, albeit small, in the verification times with the increase in the number of policies due to the size of the reachability graph this is much less evident on TechNETium, whose growth is much less pronounced.

This analysis is valid for Reachability and FullReachability but it is even more evident in the case of the ToWayPoint policy. In fact, in this policy a reachability test including a unreachable node is part of the policy definition (as defined in 6.3). Verifying a false reachability property for NetPlumber is very expensive in the worst case, while on the tree of atomic predicates, being a logical AND, the complexity in the worst case does not change.

6.5 Future Works

Future developments of this work include several improvements at every stack level. From the application point of view we are currently working on a better user improve the Policy creation and its management, with an improved and simpler graphic

interface where policies are defined already at the application level through a "language of reachabilities".
From a development point of view another interesting future development can be the deployment of TechNETium into other SDN controller as Ryu, OpenDayLight and POX.
Finally additionally, recent years have seen how P4 language is becoming increasingly popular among SDN data plane solutions. P4 is a language to configure switches. Precisely, P4 is an open source programming language that lets end users describe how networking gear should process the packet. It controls processor chips and network forwarding devices. The main paradigm change is to switch from a bottoms-up approach where fixed-function switches are built-in, to a programmable top-down approach where the user decide which functionalities wants and install.
The possibilities it offers to be able to modify the data plane and display new information and aggregated data at the switch level offer numerous opportunities to plan significant improvements in performance and usability. For this reason we are currently working on a integration of TechNETium into a programmable data plane P4-based. The main idea will be to use the advantages of PDP to produce, expose and use additional metadata information produced by the data plane.

6.6 Conclusions

In this chapter we presented TechNETium, a architecture that exploits several techniques such as atomic predicates transformation[204], SDN and model driven development[45] to formally verify custom security policies. The architecture is composed of two parts:

1. The Atomic Predicates graph generator, that builds the Atomic Predicates Graph which represent the state of the network.

2. The Policy Manager, which enable the user to define its own security policy. TechNETium will then compile the output of this phase in a sequence of reachability policies and verify them.

In order to validate our architecture we deployed it into a well known SDN controller, ONOS and we performed several tests to proof the improvements of the performances compared to the state of the art tools.
We argued that an abstracted development model approach for the policy management level is necessary to exploit the power of SDN, and we show-cased it implementing two additional custom policies. We argued with some use case that this architecture is suited for a typical system network administrator who wants to formally verify his own custom policies.
Related work includes similar tools with different techniques for the network verification level [88, 2, 132]. As we already discussed NetPlumber[103] is the tool which has been create with our same goal, but we showed that we definitely improved the performances. Veriflow [106] do not focus on custom policies but only on network invariant in order to detect network changes. MineSwepper [15] otherwise use an SMT solver to analyse all possible routing paths in order to verify network properties, but without the possibility to implement custom ones. The most similar tool in the literature is obviously AP Verifier[205]. As we pointed out TechNETium has been inspired by this tool, it uses the atomic predicates techniques and the main algorithm for the reachability graph creation. As specified in 6.2.4 we introduced several improvements defined by the same authors of AP Verifier and we also built the Policy Manager module. To the best of our knowledge, there isn't in the literature such tool which include a Formal Verification network tool based on Atomic Predicate technique, with a Policy Manager integrated into an SDN controller created with a Model Driven Development Approach.

Chapter 7

Conclusions

This thesis illustrates the main outcomes of the research activities conducted throughout the three years of the Ph.D. program. An original cybersecurity analysis of the main threats in service enabler architectures.
In chapter 2 I started with a review of the state-of-the-art of the current new service oriented software architecture.
After that, I defined an analysis path, based on a top-down approach of the software architecture based on the orchestration level of the enabled services.
Next, I developed a real-world use case for each level identified, that is:

- a Clearing System in chapter 4.1.
- a Service Enabler Platform called SMAll in 5.
- a Software Defined Full-Stack Architecture in 6.2 called techNETium.

The main contributions consists of an analysis of all the major threats and vulnerability on such architectures. In order to do that I firstly studied a categorization of the architecture components.
For this reason in 4.1.2 I proposed a snapshot management system to organize the clearing system components. In 5.2 I proposed a MaaS Stack for a generic mobility service enabling platform and, finally, in 6.3 I developed application layer for security policy definition in a software defined network environment.
I supported and proved these contributions with practical attacks on the use cases developed. As a consequence, I also proposed a set of architectural solutions for aforementioned threats.

I believe that this work opens up a plethora of novel possibilities in research as well as in any entity interested in building new service enabler software architecture.

Bibliography

[1] S. B. Akers. "Binary Decision Diagrams". In: *IEEE Trans. Comput.* 27.6 (1978), pp. 509–516. ISSN: 0018-9340. DOI: 10.1109/TC.1978.1675141. URL: https://doi.org/10.1109/TC.1978.1675141.

[2] Adnan Akhunzada et al. "Secure and dependable software defined networks". In: *Journal of Network and Computer Applications* 61 (2016), pp. 199–221.

[3] Adnan Akhunzada et al. "Secure and Dependable Software Defined Networks". In: *J. Netw. Comput. Appl.* 61.C (Feb. 2016), pp. 199–221. ISSN: 1084-8045. DOI: 10.1016/j.jnca.2015.11.012. URL: http://dx.doi.org/10.1016/j.jnca.2015.11.012.

[4] T. Alharbi, S. Layeghy, and M. Portmann. "Experimental evaluation of the impact of DoS attacks in SDN". In: *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)*. 2017, pp. 1–6. DOI: 10.1109/ATNAC.2017.8215424.

[5] Qutaibah Althebyan. *Design and analysis of knowledge-base centric insider threat models*. ProQuest, 2008.

[6] Qutaibah Althebyan et al. "Mitigating insider threats in a cloud using a knowledgebase approach while maintaining data availability". In: *ICITST*. IEEE. 2015, pp. 226–231.

[7] Amazon. *API Gateway*. https://aws.amazon.com/api-gateway/. (Visited on 09/2016).

[8] Kai Kuikkaniemi Antti Poikola and Harri Honko. *MyData A Nordic Model for human-centered personal data management and processing*. Tech. rep. Ministry of Transport Finland, 2010.

[9] Apache. *Zookeeper*. https://zookeeper.apache.org/. (Visited on 09/2016).

[10] Kevin Ashton. "That 'internet of things' thing". In: *RFiD Journal* 22.7 (2009), pp. 97–114.

[11] Swaminathan Balasubramanian et al. *Selecting a development associate for work in a unified modeling language (uml) environment*. US Patent App. 13/546,301. 2014.

[12] Debasis Bandyopadhyay and Jaydip Sen. "Internet of Things: Applications and Challenges in Technology and Standardization". In: *Wireless Personal Communications* 58.1 (2011), pp. 49–69.

[13] MICHAEL BARBARO and TOM ZELLER. *A Face Is Exposed for AOL Searcher No. 4417749*. 2006. URL: https://www.nytimes.com/2006/08/09/technology/09aol.html (visited on 10/30/2019).

[14] N.C. Batista, R. Melício, and V.M.F. Mendes. "Services enabler architecture for smart grid and smart living services providers under industry 4.0". In: *Energy and Buildings* 141 (2017), pp. 16 –27. ISSN: 0378-7788. DOI: https://doi.org/10.1016/j.enbuild.2017.02.039. URL: http://www.sciencedirect.com/science/article/pii/S0378778816316358.

[15] Ryan Beckett et al. "A general approach to network configuration verification". In: *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM. 2017, pp. 155–168.

[16] Pankaj Berde et al. "ONOS: towards an open, distributed SDN OS". In: *Proceedings of the third workshop on Hot topics in software defined networking*. ACM. 2014, pp. 1–6.

[17] Deepavali Bhagwat et al. "An annotation management system for relational databases". In: *The VLDB Journal* 14.4 (2005), pp. 373–396.

[18] Matt Bishop. "Position: Insider is relative". In: *Proceedings of Workshop on New security paradigms*. ACM. 2005, pp. 77–78.

[19] Matt Bishop et al. "Relationships and data sanitization: A study in scarlet". In: *Proceedings of the 2010 New Security Paradigms Workshop*. ACM. 2010, pp. 151–164.

[20] Nikolaj Bjørner et al. "ddNF: An Efficient Data Structure for Header Spaces". In: *Hardware and Software: Verification and Testing*. Ed. by Roderick Bloem and Eli Arbel. Cham: Springer International Publishing, 2016, pp. 49–64. ISBN: 978-3-319-49052-6.

[21] Scanbot Blog. *Government digitization trends*. 2019. URL: https://scanbot.io/blog/government-digitization-trends/ (visited on 10/30/2019).

[22] Stephen Boyd et al. "Randomized Gossip Algorithms". In: *IEEE/ACM Trans. Netw.* 14.SI (June 2006), pp. 2508–2530. ISSN: 1063-6692.

[23] Justin Brickell and Vitaly Shmatikov. "The cost of privacy: destruction of data-mining utility in anonymized data publishing". In: *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2008, pp. 70–78.

[24] Peter Buneman, Sanjeev Khanna, and Wang-Chiew Tan. "FST TCS 2000". In: ed. by Sanjiv Kapoor and Sanjiva Prasad. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000. Chap. Data Provenance: Some Basic Issues, pp. 87–93. ISBN: 978-3-540-44450-3. DOI: 10.1007/3-540-44450-5_6. URL: http://dx.doi.org/10.1007/3-540-44450-5_6.

[25] Rajkumar Buyya, Rajiv Ranjan, and Rodrigo N Calheiros. "Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services". In: *AAPP*. Springer, 2010, pp. 13–31.

[26] Rajkumar Buyya et al. "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility". In: *FGS* 25.6 (2009), pp. 599 –616. ISSN: 0167-739X. DOI: http://dx.doi.org/10.1016/j.future.2008.12.001. URL: http://www.sciencedirect.com/science/article/pii/S0167739X08001957.

[27] Béatrice Bérard et al. "Systems and Software Verification: Model-Checking Techniques and Tools". In: (Jan. 2001).

[28] Franco Callegati et al. "Data security issues in maas-enabling platforms". In: *Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI), 2016 IEEE 2nd International Forum on*. IEEE. 2016, pp. 1–5.

[29] Franco Callegati et al. "Insider Threats in Emerging Mobility-as-a-Service Scenarios". In: *HICSS*. AIS Electronic Library (AISeL), 2017. URL: http://aisel.aisnet.org/hicss-50/eg/insider_threat/4.

[30] Franco Callegati et al. "Privacy-Preserving Design of Data Processing Systems in the Public Transport Context". In: *Pacific Asia Journal of the Association for Information Systems* 7.4 (2015).

[31] Franco Callegati et al. "Privacy-preserving design of data processing systems in the public transport context". In: *Pacific Asia Journal of the Association for Information Systems* 7.4 (2015).

[32] Franco Callegati et al. "Smart Mobility for All: A Global Federated Market for Mobility-as-a-Service Operators". In: *20th International Conference on Intelligent Transportation*. 2017.

[33] Sabrina Capitani Di Vimercati, Sara Foresti, and Pierangela Samarati. "Data Protection in Cloud Scenarios". In: *Revised Selected Papers of the 10th International Workshop on Data Privacy Management, and Security Assurance-Volume 9481*. Springer-Verlag New York, Inc. 2015, pp. 3–10.

[34] Tim Casey. *A Field Guide to Insider Threat*. Tech. rep. Intel, 2015.

[35] CEN. *Service Interface for Real Time Information*. (Visited on 08/2016).

[36] D. Chen and H. Zhao. "Data Security and Privacy Protection Issues in Cloud Computing". In: *ICCSEE*. Vol. 1. 2012, pp. 647–651. DOI: 10.1109/ICCSEE.2012.193.

[37] Hsinchun Chen, Roger HL Chiang, and Veda C Storey. "Business Intelligence and Analytics: From Big Data to Big Impact." In: *MIS quarterly* 36.4 (2012), pp. 1165–1188.

[38] Rui Chen et al. "Publishing set-valued data via differential privacy". In: *Proceedings of the VLDB Endowment* 4.11 (2011), pp. 1087–1098.

[39] Youngho Cho, Gang Qu, and Yuanming Wu. "Insider threats against trust mechanism with watchdog and defending approaches in wireless sensor networks". In: *SPW*. IEEE. 2012, pp. 134–141.

[40] William R Claycomb and Alex Nicoll. "Insider threats to cloud computing: Directions for new research challenges". In: *ACSAC*. IEEE. 2012, pp. 387–394.

[41] José Cordeiro et al. *ICEIS*. Vol. 3. Springer Science & Business Media, 2008.

[42] Graham Cormode et al. "Empirical privacy and empirical utility of anonymized data". In: *2013 IEEE 29th International Conference on Data Engineering Workshops (ICDEW)*. IEEE. 2013, pp. 77–82.

[43] George F Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed systems: concepts and design*. pearson education, 2005.

[44] Rick Crawford et al. "Sanitization models and their limitations". In: *Proceedings of the 2006 workshop on New security paradigms*. ACM. 2006, pp. 41–56.

[45] Alberto Rodrigues Da Silva. "Model-driven engineering: A survey supported by the unified conceptual model". In: *Computer Languages, Systems & Structures* 43 (2015), pp. 139–155.

[46] M. C. Dacier et al. "Security Challenges and Opportunities of Software-Defined Networking". In: *IEEE Security Privacy* 15.2 (2017), pp. 96–100. ISSN: 1540-7993. DOI: 10.1109/MSP.2017.46.

[47] Chenyun Dai et al. "SDM". In: ed. by Willem Jonker and Milan Petković. Berlin, Heidelberg: Springer, 2008. Chap. An Approach to Evaluate Data Trustworthiness Based on Data Provenance, pp. 82–98. ISBN: 978-3-540-85259-9. DOI: 10.1007/978-3-540-85259-9_6. URL: http://dx.doi.org/10.1007/978-3-540-85259-9_6.

[48] Mila Dalla Preda et al. "AIOCJ: A choreographic framework for safe adaptive distributed applications". In: *SLE*. Springer. 2014, pp. 161–170.

[49] Nicodemos Damianou et al. "The Ponder Policy Specification Language". In: *IWPDSN*. POLICY '01. London, UK, UK: Springer-Verlag, 2001, pp. 18–38. ISBN: 3-540-41610-2. URL: http://dl.acm.org/citation.cfm?id=646962.712108.

[50] T. Dargahi et al. "A Survey on the Security of Stateful SDN Data Planes". In: *IEEE Communications Surveys Tutorials* 19.3 (2017), pp. 1701–1725. DOI: 10.1109/COMST.2017.2689819.

[51] Andrew DeOrio et al. "Machine Learning-based Anomaly Detection for Post-silicon Bug Diagnosis". In: *Proceedings of the Conference on Design, Automation and Test in Europe*. DATE '13. Grenoble, France: EDA Consortium, 2013, pp. 491–496. ISBN: 978-1-4503-2153-2. URL: http://dl.acm.org/citation.cfm?id=2485288.2485411.

[52] Elisabetta Di Nitto et al. *Model-driven Development and Operation of Multi-cloud Applications: The Modaclouds Approach*. Springer, 2017.

[53] Gary Doss and Guvirender Tejay. "Developing insider attack detection model: a grounded approach". In: *ISI*. IEEE. 2009, pp. 107–112.

[54] Nicola Dragoni et al. "Microservices: Yesterday, Today, and Tomorrow". In: *Present and Ulterior Software Engineering*. Ed. by Manuel Mazzara and Bertrand Meyer. Cham: Springer International Publishing, 2017, pp. 195–216. ISBN: 978-3-319-67425-4. DOI: 10.1007/978-3-319-67425-4_12. URL: https://doi.org/10.1007/978-3-319-67425-4_12.

[55] Schahram Dustdar et al. "Quality-aware Service-oriented Data Integration: Requirements, State of the Art and Open Challenges". In: *SIGMOD Rec.* 41.1 (Apr. 2012), pp. 11–19. ISSN: 0163-5808. DOI: 10.1145/2206869.2206873. URL: http://doi.acm.org/10.1145/2206869.2206873.

[56] Cynthia Dwork, Aaron Roth, et al. "The algorithmic foundations of differential privacy". In: *Foundations and Trends® in Theoretical Computer Science* 9.3–4 (2014), pp. 211–407.

[57] H. Eldardiry et al. "Multi-Domain Information Fusion for Insider Threat Detection". In: *SPW*. 2013, pp. 45–51. DOI: 10.1109/SPW.2013.14.

[58] Thomas Erl. *Service-Oriented Architecture: Analysis and Design for Services and Microservices*. 2nd. Upper Saddle River, NJ, USA: Prentice Hall Press, 2016. ISBN: 0133858588, 9780133858587.

[59] Thomas Erl. *SOA Principles of Service Design (paperback)*. Prentice Hall Press, 2016.

[60] Michael D Ernst. "Static and dynamic analysis: Synergy and duality". In: *WODA 2003: ICSE Workshop on Dynamic Analysis*. Citeseer. 2003, pp. 24–27.

[61] C. Falge, B. Otto, and H. Österle. "Data Quality Requirements of Collaborative Business Processes". In: *HICSS*. 2012, pp. 4316–4325. DOI: 10.1109/HICSS.2012.8.

[62] S. Fickas, G. Kortuem, and Z. Segall. "Software organization for dynamic and adaptable wearable systems". In: *Digest of Papers. First International Symposium on Wearable Computers*. 1997, pp. 56–63. DOI: 10.1109/ISWC.1997.629920.

[63] Roy Thomas Fielding. "Architectural styles and the design of network-based software architectures". PhD thesis. UC Irvine, 2000.

[64] Lori Flynn, Greg Porter, and Chas DiFatta. "Cloud Service Provider Methods for Managing Insider Threats: Analysis Phase II, Expanded Analysis and Recommendations". In: (2014).

[65] Eric G. W. Wong. "Validating Network Security Policies via Static Analysis of Router ACL Configuration". In: (Dec. 2006), p. 169.

[66] Maurizio Gabbrielli et al. "Self-Reconfiguring Microservices". In: *Theory and Practice of Formal Methods*. Springer, 2016, pp. 194–210.

[67] Srivatsava Ranjit Ganta, Shiva Prasad Kasiviswanathan, and Adam Smith. "Composition attacks and auxiliary information in data privacy". In: *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2008, pp. 265–273.

[68] Damianos Gavalas, Charalampos Konstantopoulos, and Grammati Pantziou. "Design and management of vehicle sharing systems: a survey of algorithmic approaches". In: *arXiv preprint arXiv:1510.01158* (2015).

[69] Michael Gertz and Sushil Jajodia. *Handbook of database security: applications and trends*. Springer, 2007.

[70] Moein Ghasemzadeh et al. "Anonymizing trajectory data for passenger flow analysis". In: *Transportation research part C: emerging technologies* 39 (2014), pp. 63–79.

[71] Gabriel Ghinita, Yufei Tao, and Panos Kalnis. "On the anonymization of sparse high-dimensional data". In: *2008 IEEE 24th International Conference on Data Engineering*. IEEE. 2008, pp. 715–724.

[72] Saverio Giallorenzo. "Real-World Choreographies". PhD thesis. Università degli studi di Bologna, 2016.

[73] Dino Giuli et al. "Toward a Cooperative Approach for Continuous Innovation of Mobility Information Services". In: *IEEE Systems Journal* 7.4 (2013), pp. 669–680.

[74] Henry G Goldberg et al. "Explaining and Aggregating Anomalies to Detect Insider Threats". In: *HICSS*. IEEE. 2016, pp. 2739–2748.

[75] Li Gong et al. "Going Beyond the Sandbox: An Overview of the New Security Architecture in the Java Development Kit 1.2." In: *USENIX - USITS*. 1997, pp. 103–112.

[76] Google. *Google Transit Feed Specification*. https://developers.google.com/transit/. (Visited on 06/2016).

[77] Google. *Google Transit Feed Specification | Realtime Transit*. https://developers.google.com/transit/gtfs-realtime/. (Visited on 08/2016).

[78] M. G. Gouda and X. A. Liu. "Firewall design: consistency, completeness, and compactness". In: *24th International Conference on Distributed Computing Systems, 2004. Proceedings.* 2004, pp. 320–327. DOI: 10.1109/ICDCS.2004.1281597.

[79] Paul Groth, Michael Luck, and Luc Moreau. "A protocol for recording provenance in service-oriented grids". In: *Principles of Distributed Systems*. Springer, 2004, pp. 124–139.

[80] Jayavardhana Gubbi et al. "Internet of Things (IoT): A vision, architectural elements, and future directions". In: *Future generation computer systems* 29.7 (2013), pp. 1645–1660.

[81] Zygmunt J. Haas, Joseph Y. Halpern, and Li Li. "Gossip-based Ad Hoc Routing". In: *IEEE/ACM Trans. Netw.* 14.3 (June 2006), pp. 479–491. ISSN: 1063-6692.

[82] Mark Hall et al. "The WEKA data mining software: an update". In: *ACM SIGKDD explorations newsletter* 11.1 (2009), pp. 10–18.

[83] Lianlian He et al. "Adding geospatial data provenance into SDI—a service-oriented approach". In: *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of* 8.2 (2015), pp. 926–936.

[84] Yeye He and Jeffrey F Naughton. "Anonymization of set-valued data via top-down, local generalization". In: *Proceedings of the VLDB Endowment* 2.1 (2009), pp. 934–945.

[85] Thomas S Heydt-Benjamin et al. "Privacy for public transportation". In: *IW-PET*. Springer. 2006, pp. 1–19.

[86] Shuyuan Mary Ho et al. "Demystifying insider threat: Language-action cues in group dynamics". In: *HICSS*. IEEE. 2016, pp. 2729–2738.

[87] Jan Holler et al. *From Machine-to-machine to the Internet of Things: Introduction to a New Age of Intelligence*. Academic Press, 2014.

[88] Alex Horn, Ali Kheradmand, and Mukul Prasad. "Delta-net: Real-time Network Verification Using Atoms". In: *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. Boston, MA: USENIX Association, 2017, pp. 735–749. ISBN: 978-1-931971-37-9. URL: https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/horn-alex.

[89] Alex Horn, Ali Kheradmand, and Mukul R Prasad. "Delta-net: Real-time Network Verification Using Atoms." In: *NSDI*. 2017, pp. 735–749.

[90] Jeff Howe. "The rise of crowdsourcing". In: *Wired magazine* 14.6 (2006), pp. 1–4.

[91] Vincent C Hu et al. "Guide to attribute based access control (ABAC) definition and considerations (draft)". In: *NIST Special Publication* 800.162 (2013).

[92] Jeffrey Hunker and Christian W Probst. "Insiders and Insider Threats-An Overview of Definitions and Mitigation Techniques." In: *JoWUA* 2.1 (2011), pp. 4–27.

[93] Kasun Indrasiri. "Microservices in pratice - key architectural concepts of an MSA". In: *Wso2 White Paper* (2016).

[94] T. Inoue et al. "An efficient framework for data-plane verification with geometric windowing queries". In: *2016 IEEE 24th International Conference on Network Protocols (ICNP)*. 2016, pp. 1–10. DOI: 10.1109/ICNP.2016.7784412.

[95] Raj Jain and Subharthi Paul. "Network virtualization and software defined networking for cloud computing: a survey". In: *IEEE Communications Magazine* 51.11 (2013), pp. 24–31.

[96] P. Jamshidi et al. "Microservices: The Journey So Far and Challenges Ahead". In: *IEEE Software* 35.3 (2018), pp. 24–35. DOI: 10.1109/MS.2018.2141039.

[97] Márk Jelasity, Alberto Montresor, and Ozalp Babaoglu. "Gossip-based Aggregation in Large Dynamic Networks". In: *ACM Trans. Comput. Syst.* 23.3 (Aug. 2005), pp. 219–252. ISSN: 0734-2071.

[98] Jolie Team. *Jolie Programming Language*. http://jolie-lang.org. (Visited on 06/2016).

[99] Garvit Juniwal, Sanjit A. Seshia, and George Varghese. "Quantitative Network Analysis". In: 2015.

[100] Audun Jøsang, Roslan Ismail, and Colin Boyd. "A survey of trust and reputation systems for online service provision". In: *Decision Support Systems* 43.2 (2007), pp. 618 –644. ISSN: 0167-9236. DOI: http://dx.doi.org/10.1016/j.dss.2005.05.019. URL: http://www.sciencedirect.com/science/article/pii/S0167923605000849.

[101] Ashish Kamra, Evimaria Terzi, and Elisa Bertino. "Detecting anomalous access patterns in relational databases". In: *The VLDB Journal* 17.5 (2008), pp. 1063–1077.

[102] Miltiadis Kandias, Nikos Virvilis, and Dimitris Gritzalis. "The Insider Threat in Cloud Computing". In: *Critical Information Infrastructure Security: 6th International Workshop, CRITIS 2011, Lucerne, Switzerland, September 8-9, 2011, Revised Selected Papers*. Ed. by Sandro Bologna et al. Springer, 2013, pp. 93–103. ISBN: 978-3-642-41476-3. DOI: 10.1007/978-3-642-41476-3_8. URL: http://dx.doi.org/10.1007/978-3-642-41476-3_8.

[103] Peyman Kazemian. *NetPlumber, Bitbucket repository*. https://bitbucket.org/peymank/hassel-public/branch/hassel-dev. 2013.

[104] Peyman Kazemian, George Varghese, and Nick McKeown. "Header Space Analysis: Static Checking for Networks". In: *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*. NSDI'12. San Jose, CA: USENIX Association, 2012, pp. 9–9. URL: http://dl.acm.org/citation.cfm?id=2228298.2228311.

[105] Peyman Kazemian et al. "Real Time Network Policy Checking Using Header Space Analysis". In: *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*. Lombard, IL: USENIX, 2013, pp. 99–111. ISBN: 978-1-931971-00-3. URL: https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/kazemian.

[106] Ahmed Khurshid et al. "VeriFlow: Verifying Network-Wide Invariants in Real Time". In: *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*. Lombard, IL: USENIX, 2013, pp. 15–27. ISBN: 978-1-931971-00-3. URL: https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/khurshid.

[107] D. Kreutz et al. "Software-Defined Networking: A Comprehensive Survey". In: *Proceedings of the IEEE* 103.1 (2015), pp. 14–76. ISSN: 0018-9219. DOI: 10.1109/JPROC.2014.2371999.

[108] Diego Kreutz, Fernando M.V. Ramos, and Paulo Verissimo. "Towards Secure and Dependable Software-defined Networks". In: *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*. HotSDN '13. Hong Kong, China: ACM, 2013, pp. 55–60. ISBN: 978-1-4503-2178-5. DOI: 10.1145/2491185.2491199. URL: http://doi.acm.org/10.1145/2491185.2491199.

[109] Ronald L. Krutz and Russell Dean Vines. *Cloud Security: A Comprehensive Guide to Secure Cloud Computing*. Wiley Publishing, 2010. ISBN: 0470589876, 9780470589878.

[110] Heiner Lasi et al. "Industry 4.0". In: *Business & Information Systems Engineering* 6.4 (2014), pp. 239–242.

[111] N. Li, T. Li, and S. Venkatasubramanian. "t-Closeness: Privacy Beyond k-Anonymity and l-Diversity". In: *2007 IEEE 23rd International Conference on Data Engineering*. 2007, pp. 106–115. DOI: 10.1109/ICDE.2007.367856.

[112] Hyo-Sang Lim, Yang-Sae Moon, and Elisa Bertino. "Provenance-based trustworthiness assessment in sensor networks". In: *IWDMSN*. ACM. 2010, pp. 2–7.

[113] Fang Liu et al. "NIST cloud computing reference architecture". In: *NIST special publication* 500.2011 (2011), p. 292.

[114] Y. Liu, C. Lei, and H. Zhang. "MR-Verifier: Verifying Open Flow Network Properties Based on MapReduce". In: *2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*. 2015, pp. 546–553. DOI: 10.1109/CyberC.2015.57.

[115] Lyft. *Lyft.com*. https://www.lyft.com/. (Visited on 06/2016).

[116] A. Machanavajjhala et al. "L-diversity: privacy beyond k-anonymity". In: *22nd International Conference on Data Engineering (ICDE'06)*. 2006, pp. 24–24. DOI: 10.1109/ICDE.2006.1.

[117] C. Matthew MacKenzie et al. *Reference Model for Service Oriented Architecture 1.0*. Tech. rep. OASIS, 2006. URL: http://docs.oasis-open.org/soa-rm/v1.0/.

[118] Stuart E. Madnick et al. "Overview and Framework for Data and Information Quality Research". In: *J. Data and Information Quality* 1.1 (June 2009), 2:1–2:22. ISSN: 1936-1955. DOI: 10.1145/1515693.1516680. URL: http://doi.acm.org/10.1145/1515693.1516680.

[119] Matthew Vincent Mahoney. "A Machine Learning Approach to Detecting Attacks by Identifying Anomalies in Network Traffic". AAI3081393. PhD thesis. Melbourne, FL, USA, 2003.

[120] Mark W Maier. "Architecting principles for systems-of-systems". In: *INCOSE*. Vol. 6. 1. Wiley Online Library. 1996, pp. 565–573.

[121] Mark W Maier. "Research challenges for systems-of-systems". In: *2005 IEEE SMC*. Vol. 4. IEEE. 2005, pp. 3149–3154.

[122] Ross A. Malaga. "Web-Based Reputation Management Systems: Problems and Suggested Solutions". In: *ECR* 1.4 (), pp. 403–417. ISSN: 1572-9362. DOI: 10.1023/A:1011557319152. URL: http://dx.doi.org/10.1023/A:1011557319152.

[123] Sunu Mathew et al. "A data-centric approach to insider attack detection in database systems". In: *RAID*. Springer. 2010, pp. 382–401.

[124] David Kupfer Matthias Finger Nadia Bert. *Mobility-as-a-Service: from the Helsinki experiment to a European model?* Tech. rep. Observer European Transport Regulation, 2015.

[125] A. Melis et al. "A Policy Checker Approach for Secure Industrial SDN". In: *2018 2nd Cyber Security in Networking Conference (CSNet)*. 2018, pp. 1–7. DOI: 10.1109/CSNET.2018.8602927.

[126] Andrea Melis et al. "A Policy Checker Approach for Secure Industrial SDN". In: *2018 2nd Cyber Security in Networking Conference (CSNet)*. 2018, pp. 1–7.

[127] Dirk Merkel. "Docker: Lightweight Linux Containers for Consistent Development and Deployment". In: *Linux J.* 2014.239 (Mar. 2014). ISSN: 1075-3583. URL: {\url{http://dl.acm.org/citation.cfm?id=2600239.2600241}}.

[128] Ralph C. Merkle. "CRYPTO". In: ed. by Carl Pomerance. Berlin, Heidelberg: Springer Berlin Heidelberg, 1988. Chap. A Digital Signature Based on a Conventional Encryption Function, pp. 369–378. ISBN: 978-3-540-48184-3. DOI: 10.1007/3-540-48184-2_32. URL: http://dx.doi.org/10.1007/3-540-48184-2_32.

[129] Adam Meyerson and Ryan Williams. "On the complexity of optimal k-anonymity". In: *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM. 2004, pp. 223–228.

[130] Silvia Mirri et al. "A Microservice Architecture Use Case for Persons with Disabilities". In: *CVSJ*. Hindawi. 2016, p. 5.

[131] Silvia Mirri et al. "A Service-Oriented Approach to Crowdsensing for Accessible Smart Mobility Scenarios". In: *Proceedings ICCTS*. IEEE. 2016, p. 5.

[132] Saber Mirzaei and Flavio Esposito. "An alloy verification model for consensus-based auction protocols". In: *Distributed Computing Systems Workshops (ICDCSW), 2015 IEEE 35th International Conference on*. IEEE. 2015, pp. 17–22.

[133] Bhoomika R Mistry and Amish Desai. "Privacy preserving heuristic approach for association rule mining in distributed database". In: *ICIIECS*. IEEE. 2015, pp. 1–7.

[134] David Molnar et al. *Automatic context-sensitive sanitization*. US Patent 8,898,776. 2014.

[135] Fabrizio Montesi. "Choreographic Programming". PhD thesis. IT University of Copenhagen, 2013.

[136] Fabrizio Montesi. "Process-aware web programming with Jolie". In: *Science of Computer Programming* 130 (2016), pp. 69–96.

[137] Fabrizio Montesi, Claudio Guidi, and Gianluigi Zavattaro. "Service-oriented programming with jolie". In: *Web Services Foundations*. Springer, 2014, pp. 81–107.

[138] Fabrizio Montesi and Janine Weber. "Circuit Breakers, Discovery, and API Gateways in Microservices". In: *CoRR* abs/1609.05830 (2016). URL: http://arxiv.org/abs/1609.05830.

[139] Steven Morrison and Clifford Winston. *The evolution of the airline industry*. Brookings Institution Press, 1995.

[140] A. Moser, C. Kruegel, and E. Kirda. "Limits of Static Analysis for Malware Detection". In: *ACSAC*. 2007, pp. 421–430. DOI: 10.1109/ACSAC.2007.21.

[141] Michel Mouly, Marie-Bernadette Pautet, and Thomas Foreword By-Haug. *The GSM system for mobile communications*. Telecom publishing, 1992.

[142] Hyeran Mun et al. "Yet another intrusion detection system against insider attacks". In: *Proc. of SCIS* (2008).

[143] Andrew Nash et al. "RailML–a standard data interface for railroad applications". In: *Computers in Railways IX, WIT Press, Southampton* (2004), pp. 233–240.

[144] Mehmet Ercan Nergiz, Maurizio Atzori, and Chris Clifton. "Hiding the presence of individuals from shared databases". In: *SIGMOD*. ACM. 2007, pp. 665–676.

[145] Netflix. *Eureka*. https://github.com/Netflix/eureka. (Visited on 09/2016).

[146] Netflix. *Hystrix*. https://github.com/Netflix/hystrix. (Visited on 09/2016).

[147] Netflix. *Ribbon*. https://github.com/Netflix/ribbon. (Visited on 09/2016).

[148] Netflix. *Zuul*. https://github.com/Netflix/zuul. (Visited on 09/2016).

[149] Sam Newman. *Building microservices: designing fine-grained systems*. " O'Reilly Media, Inc.", 2015.

[150] Nicola Nostro et al. "Insider threat assessment: A model-based methodology". In: *ACM SIGOPS* 48.2 (2014), pp. 3–12.

[151] James M Parker. "Applying a system of systems approach for improved transportation". In: *SAPIENS* 3.2 (2010).

[152] Sean Peisert et al. "Quis Custodiet Ipsos Custodes?: A New Paradigm for Analyzing Security Paradigms with Appreciation to the Roman Poet Juvenal". In: *Proceedings of the 2009 Workshop on New Security Paradigms Workshop*. NSPW '09. Oxford, United Kingdom: ACM, 2009, pp. 71–84. ISBN: 978-1-60558-845-2. DOI: 10.1145/1719030.1719041. URL: http://doi.acm.org/10.1145/1719030.1719041.

[153] J. Pfrommer et al. "Dynamic Vehicle Redistribution and Online Price Incentives in Shared Mobility Systems". In: *IEEE Transactions on Intelligent Transportation Systems* 15.4 (2014), pp. 1567–1578. ISSN: 1524-9050. DOI: 10.1109/TITS.2014.2303986.

[154] J. Weston Phippen. *Who Drives a Driverless Car?* URL: http://www.theatlantic.com/national/archive/2016/02/google-driverless-car/462153/ (visited on 02/10/2016).

[155] S. Pippuri, S. Hietanen, and K. Pyyhtiä. *MaaS Finland*. http://maas.fi/. (Visited on 03/10/2016).

[156] Gordon D. Plotkin et al. "Scaling Network Verification Using Symmetry and Surgery". In: *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. POPL '16. St. Petersburg, FL, USA: ACM, 2016, pp. 69–83. ISBN: 978-1-4503-3549-2. DOI: 10.1145/2837614.2837657. URL: http://doi.acm.org/10.1145/2837614.2837657.

[157] Vassilis Prevelakis and Diomidis Spinellis. "Sandboxing Applications." In: *USENIX Annual Tech. Conf., FREENIX Track*. 2001, pp. 119–126.

[158] D. B. Rawat and S. R. Reddy. "Software Defined Networking Architecture, Security and Energy Efficiency: A Survey". In: *IEEE Communications Surveys Tutorials* 19.1 (2017), pp. 325–346. ISSN: 1553-877X. DOI: 10.1109/COMST.2016.2618874.

[159] Paul Resnick et al. "Reputation Systems". In: *Commun. ACM* 43.12 (Dec. 2000), pp. 45–48. ISSN: 0001-0782. DOI: 10.1145/355112.355122. URL: http://doi.acm.org/10.1145/355112.355122.

[160] Benny Rochwerger et al. "The reservoir model and architecture for open federated cloud computing". In: *IBM JRD* 53.4 (2009), pp. 4–1.

[161] Rodrigo Roman, Jianying Zhou, and Javier Lopez. "On the features and challenges of security and privacy in distributed internet of things". In: *Computer Networks* 57.10 (2013), pp. 2266–2279.

[162] Sherif Sakr et al. "A survey of large scale data management approaches in cloud environments". In: *IEEE Communications Surveys & Tutorials* 13.3 (2011), pp. 311–336.

[163] Pierangela Samarati. "Protecting respondents identities in microdata release". In: *IEEE transactions on Knowledge and Data Engineering* 13.6 (2001), pp. 1010–1027.

[164] Kaj Pyyhtiä Sami Pippuri Sampo Hietanen. *MaaS Finland*. URL: http://maas.fi/ (visited on 03/10/2016).

[165] ITS-Finland Sampo Hietanen CEO. *'Mobility as a Service' – the new transport model?* Tech. rep. MaaS Finland, 2014.

[166] Ravi Sandhu. "Attribute-Based Access Control Models and Beyond." In: *ASIACCS*. 2015, p. 677.

[167] Ravi S Sandhu et al. "Role-based access control models yz". In: *IEEE computer* 29.2 (1996), pp. 38–47.

[168] A. Sarkar et al. "Insider Attack Identification and Prevention in Collection-Oriented Dataflow-Based Processes". In: *IEEE Systems Journal* 11.2 (2017), pp. 522–533. ISSN: 1932-8184. DOI: 10.1109/JSYST.2015.2477472.

[169] T.G.J. Schepers, Maria Eugenia Iacob, and Pascal van Eck. "A lifecycle approach to SOA Governance". In: *SAC '08*. Special track on Enterprise Information Systems. ACM Press, 2008, pp. 1055–1061. ISBN: 978-1-59593-753-7. DOI: 10.1145/1363686.1363932.

[170] Bob G Schlicher, Lawrence P MacIntyre, and Robert K Abercrombie. "Towards Reducing the Data Exfiltration Surface for the Insider Threat". In: *HICSS*. IEEE. 2016, pp. 2749–2758.

[171] Bruce Schneier. *Secrets and lies: digital security in a networked world*. John Wiley & Sons, 2011.

[172] S. Scott-Hayward, G. O'Callaghan, and S. Sezer. "Sdn Security: A Survey". In: *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*. 2013, pp. 1–7. DOI: 10.1109/SDN4FNS.2013.6702553.

[173] Ravi Seethamraju. "Adoption of software as a service (SaaS) enterprise resource planning (ERP) systems in small and medium sized enterprises (SMEs)". In: *Information systems frontiers* 17.3 (2015), pp. 475–492.

[174] Asaf Shabtai, Yuval Elovici, and Lior Rokach. *A survey of data leakage detection and prevention solutions*. Springer, 2012.

[175] Nahla Shatnawi, Qutaibah Althebyan, and Wail Mardini. "Detection of Insiders Misuse in Database Systems". In: *ICECS*. Vol. 1. 2011.

[176] Siddle. *I Know Where You Were Last Summer: London's public bike data is telling everyone where you've been*. 2014. URL: https://vartree.blogspot.com/2014/04/i-know-where-you-were-last-summer.html (visited on 10/30/2019).

[177] George Silowash et al. *Common sense guide to mitigating insider threats 4th edition*. Tech. rep. DTIC Document, 2012.

[178] Yogesh L. Simmhan, Beth Plale, and Dennis Gannon. "A Survey of Data Provenance in e-Science". In: *SIGMOD Rec.* 34.3 (Sept. 2005), pp. 31–36. ISSN: 0163-5808. DOI: 10.1145/1084805.1084812. URL: http://doi.acm.org/10.1145/1084805.1084812.

[179] Yogesh L Simmhan, Beth Plale, and Dennis Gannon. "A survey of data provenance in e-science". In: *ACM Sigmod Record* 34.3 (2005), pp. 31–36.

[180] Yogesh L Simmhan, Beth Plale, and Dennis Gannon. "A survey of data provenance techniques". In: *Computer Science Department, Indiana University, Bloomington IN* 47405 (2005).

[181] Lance Spitzner. "Honeypots: Catching the insider threat". In: *CSAC*. IEEE. 2003, pp. 170–179.

[182] Vasilis Stavrou et al. "Business Process Modeling for Insider Threat Monitoring and Handling". In: *Trust, Privacy, and Security in Digital Business: 11th International Conference, TrustBus 2014, Munich, Germany, September 2-3, 2014. Proceedings*. Ed. by Claudia Eckert, Sokratis K. Katsikas, and Günther Pernul. Cham: Springer International Publishing, 2014, pp. 119–131. ISBN: 978-3-319-09770-1. DOI: 10.1007/978-3-319-09770-1_11. URL: http://dx.doi.org/10.1007/978-3-319-09770-1_11.

[183] S. Subashini and V. Kavitha. "A survey on security issues in service delivery models of cloud computing". In: *Journal of Network and Computer Applications* 34.1 (2011), pp. 1 –11. ISSN: 1084-8045. DOI: http://dx.doi.org/10.1016/j.jnca.2010.07.006. URL: http://www.sciencedirect.com/science/article/pii/S1084804510001281.

[184] Manolis Terrovitis, Nikos Mamoulis, and Panos Kalnis. "Privacy-preserving anonymization of set-valued data". In: *Proceedings of the VLDB Endowment* 1.1 (2008), pp. 115–125.

[185] The Open Networking Foundation. *OpenFlow Switch Specification v 1.5.1*. 2015.

[186] J. Thönes. "Microservices". In: *IEEE Software* 32.1 (2015), pp. 116–116. DOI: 10.1109/MS.2015.11.

[187] Wei-Tek Tsai et al. "Data provenance in SOA: security, reliability, and integrity". In: *Service Oriented Computing and Applications* 1.4 (2007), pp. 223–247.

[188] Zhen Tu et al. "Beyond k-anonymity: protect your trajectory from semantic attack". In: *2017 14th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE. 2017, pp. 1–9.

[189] Uber. *Uber.com*. https://www.uber.com/. (Visited on 06/2016).

[190] Vijaya Bhaskar Velpula and Dayanandam Gudipudi. "Behavior-anomaly-based system for detecting insider attacks and data mining". In: *IJRTE* 1.2 (2009), pp. 261–266.

[191] M. Villamizar et al. "Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud". In: *2015 10th Computing Colombian Conference (10CCC)*. 2015, pp. 583–590. DOI: 10.1109/ColumbianCC.2015.7333476.

[192] Sabrina De Capitani di Vimercati, Sara Foresti, and Pierangela Samarati. "Data security issues in cloud scenarios". In: *International Conference on Information Systems Security*. Springer International Publishing. 2015, pp. 3–10.

[193] Marco Viviani. *Mobility as a service*. 2015. URL: http://www.webnews.it/2015/08/07/mobility-as-a-service/ (visited on 03/10/2016).

[194] H. Wang et al. "Practical Network-Wide Packet Behavior Identification by AP Classifier". In: *IEEE/ACM Transactions on Networking* 25.5 (2017), pp. 2886–2899. ISSN: 1063-6692. DOI: 10.1109/TNET.2017.2720637.

[195] Pingshui Wang and Jiandong Wang. "L-diversity algorithm for incremental data release". In: *Applied Mathematics & Information Sciences* 7.5 (2013), p. 2055.

[196] Ye Wang, Bing-rong Lin, and Shantanu Rane. *Privacy Preserving Statistical Analysis on Distributed Databases*. US Patent App. 13/804,701. Sept. 2014.

[197] WSO2. *WSO2 API Manager*. http://wso2.com/api-management/. (Visited on 09/2016).

[198] K. Xu et al. "Data-Provenance Verification For Secure Hosts". In: *IEEE Transactions on Dependable and Secure Computing* 9.2 (2012), pp. 173–183. ISSN: 1545-5971. DOI: 10.1109/TDSC.2011.50.

[199] Yabo Xu et al. "Publishing sensitive transactions for itemset utility". In: *2008 Eighth IEEE International Conference on Data Mining*. IEEE. 2008, pp. 1109–1114.

[200] Q. Yan et al. "Software-Defined Networking (SDN) and Distributed Denial of Service (DDoS) Attacks in Cloud Computing Environments: A Survey, Some Research Issues, and Challenges". In: *IEEE Communications Surveys Tutorials* 18.1 (2016), pp. 602–622. ISSN: 1553-877X. DOI: 10.1109/COMST.2015.2487361.

[201] H. Yang and S. S. Lam. "Real-time verification of network properties using Atomic Predicates". In: *2013 21st IEEE International Conference on Network Protocols (ICNP)*. 2013, pp. 1–11. DOI: 10.1109/ICNP.2013.6733614.

[202] H. Yang and S. S. Lam. "Scalable Verification of Networks With Packet Transformers Using Atomic Predicates". In: *IEEE/ACM Transactions on Networking* 25.5 (2017), pp. 2900–2915. ISSN: 1063-6692. DOI: 10.1109/TNET.2017.2720172.

[203] Hongkun Yang and Simon S. Lam. *Networking Research Lab website*. http://www.cs.utexas.edu/users/lam/NRL/.

[204] Hongkun Yang and Simon S. Lam. *Real-time Verification of Network Properties Using Atomic Predicates*. Tech. rep. Computer Science Dept., University of Texas at Austin, Austin, TX, USA, 2013.

[205] Hongkun Yang and Simon S. Lam. "Real-time Verification of Network Properties Using Atomic Predicates". In: *IEEE/ACM Transactions on Networking* 24.2 (2016), pp. 887–900.

[206] Bendert Zevenbergen et al. "Ethical privacy guidelines for mobile connectivity measurements". In: *Available at SSRN 2356824* (2013).

[207] H. Zhang et al. "User Intention-Based Traffic Dependence Analysis for Anomaly Detection". In: *2012 IEEE Symposium on Security and Privacy Workshops*. 2012, pp. 104–112. DOI: 10.1109/SPW.2012.15.

[208] Hao Zhang, Danfeng Daphne Yao, and Naren Ramakrishnan. "Detection of Stealthy Malware Activities with Traffic Causality and Scalable Triggering Relation Discovery". In: *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security*. ASIA CCS '14. Kyoto, Japan: ACM, 2014, pp. 39–50. ISBN: 978-1-4503-2800-5. DOI: 10.1145/2590296.2590309. URL: http://doi.acm.org/10.1145/2590296.2590309.

[209] Dongfang Zhao et al. "Distributed data provenance for large-scale data-intensive computing". In: *CLUSTER*. IEEE. 2013, pp. 1–8.

[210] Jinghan Zhou et al. "APF: Fast Network All-pair Reachability Calculation". In: *Proceedings of the 2018 Symposium on Architectures for Networking and Communications Systems*. ANCS '18. Ithaca, New York: ACM, 2018, pp. 172–173. ISBN: 978-1-4503-5902-3. DOI: 10.1145/3230718.3232112. URL: http://doi.acm.org/10.1145/3230718.3232112.

[211] Shao Ying Zhu et al. *Guide to Security in SDN and NFV: Challenges, Opportunities, and Applications*. Springer, 2017.