Security architectures for network slice management for 5G and beyond

Sabra Ben Saad

Security architectures for network slice management for 5G and beyond

Submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in Information and Communication Engineering

by

Sabra BEN SAAD

Doctoral School of Informatics,
Telecommunications and Electronics of Paris

EURECOM

Publicly presented and defended on February, 2023

ABSTRACT

Network slicing architecture, enabled by new technologies such as Network Functions Virtualization (NFV) and Software-Defined Networking (SDN), is one of the main pillars of Fifth-generation and Beyond (B5G). In B5G settings, the number of coexisting slices with varying degrees of complexity and very diverse lifespans, resource requirements, and performance targets is expected to explode. This creates significant challenges towards zero-touch slice management and orchestration, including security, fault management, and trust. In addition, network slicing opens the business market to new stakeholders, namely the vertical or tenant, the network slice provider, and the infrastructure provider. In this context, there is a need to ensure not only a secure interaction between these actors, but also that each actor delivers the expected service to meet the network slice requirements. Therefore, new trust architectures should be designed, which are able to identify/detect the new forms of slicing-related attacks in real-time, while securely and automatically managing Service Level Agreements (SLA) among the involved actors.
In this thesis, we devise new security architectures tailored to network slicing ready networks (B5G), heavily relying on blockchain and Artificial Intelligence (AI) to enable secure and trust network slicing management.

Keywords: *B5G*, *Zero Touch Management*, *Network Slicing Security*, *Blockchain*, *Federated Deep Learning*, *eXplainable AI*.

ACKNOWLEDGEMENT

With immense pleasure and deep sense of gratitude, I wish to express my sincere thanks to my supervisor **Prof. Adlen Ksentini**, Communication system, Eurecom, for motivating me to carry out research in Eurecom, and also for providing me with infrastructural facilities and many other resources needed for my research.

I am grateful to the supervisor **Prof. Brik Bouziane**, University of Bourgogne, for his motivation and continuous encouragement. I thank him for all his useful suggestions, advice, and useful discussions throughout this journey.

Moreover, I sincerely thank the jury members, **Prof. Zeghlache Djamal, Prof. Piamrat Kandaraj, Prof. Moungla Hassine, Dr. Sedjelmaci Hichem and Prof. Chiasserini Carla Fiabana**, for their time, effort, and expertise in evaluating my thesis.

I wish to extend my profound sense of gratitude to **my parents, Rafik Ben Saad and Amna Aref,** for all the sacrifices they made during my research and also providing me with moral support and encouragement whenever required.

I would like to thank **my brother, Abd Hamid Ben Saad**, and my sister, **Ichrak Ben Saad,** for their constant encouragement and moral support along with patience and understanding.

Finally, I express my deepest gratitude to you, **God**, for your blessings and for being my guiding light throughout this endeavor.

Place: 06410, Nice, France.

Date: February 2023. **Sabra Ben Saad**

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF TERMS AND ABBREVIATIONS

2G 2^{th} generation mobile network 2

3G 3^{th} generation mobile network 2

4G Fourth generation mobile network 2, 33

5G Fifth generation networks 2, 3, 6, 18, 22, 25, 26, 30, 33, 42, 119

ADAM ADAptive Moment estimation 84

ADCs Application Distribution Controllers 11

AE Analytic Engine 5, 50, 55, 81, 83, 119

AI Artificial Intelligence . . . . . i, 3, 4

AMF Acess Mobility Function 15, 77, 81, 82

API Application Programming Interface 10, 41

AR Augmented Reality 2

B5G Fifth-generation and Beyond . . . . . i, 76, 78, 80, 81

BBU Baseband Unit 18

Cloud-RAN Cloud Radio Access Network 18

CN Core Network 3, 5, 6, 50

CPU Central Processing Unit 27, 77

CU Central Unit 27

DApp Decentralized Application 24, 25

DDoS Distributed Denial of Service 3, 6, 49, 50, 56, 119

DE Decision Engine 5, 50, 55, 81, 83, 119

DL Deep Learning 76, 77

DPI Deep Packet Inspection 53

DQN Deep Q-Network algorithm 85, 86

DRL Deep Reinforcement Learning 77, 81

NIDS Network Intrusion Detection System 54

NS Network Slicing 25, 27, 119

NSB Network Slice Broker 25

OAI OpenAirInterface 67, 77, 82

PCA Principal Component Analysis 87

PNF Physical Network Functions 26, 27

PRB Physical Resource Blocks 27

QoS Quality of Service 2, 9, 19, 22, 27, 30, 33

RAM Random Access Memory 77

RAN Radio Access Network 3, 22, 26, 51

RO Resource Orchestrator 28

RP Resource Providers 22, 23, 26–30, 32–42, 45–47

RRH Remote Radio Head 27

RRU Remote Radio Unit 27

SDN Software-Defined Networking . . . . . i, 2, 9, 52, 76

SGD Stochastic Gradient Descent 78, 84

SIM Subscriber Identification Module 19

SLA Service Level Agreements . i, 3–6, 22, 23, 26, 28–30, 32–38, 40–42, 45–47, 76, 77, 81, 119

SLO Service Level Objectives 34

SO Slice Orchestrator 28

SON Self Organizing Network 19

SP Slice Provider 22, 23, 26–38, 40, 45, 46

SSC Sub-Slice Contract 28, 29, 40

SSD Sub-Slice Deployment 28

SSDC Sub-Slice Deployment Costs 28, 29

SVM Support Vector Machines 53, 54

TCP Transmission Control Protocol 52

TD Technological Domains 22, 23, 26–29, 32, 37, 38, 40, 41, 45

TTP Third Party Trust 36, 45

UDM Unified Data Management 57

UDP User Datagram Protocol 52

UE User Equipments 4, 50, 51, 77

uRLLC ultra-Reliable Low Latency Communications . 3, 22, 25, 30, 33, 34, 38, 42, 45

V Vertical 22, 23, 26–28, 30–38, 41, 42, 45, 46

VLAN Virtual Local Area Network 27

VM Virtual Machine 11, 27

VNF Virtual Network Functions 11, 26, 27, 76, 77, 119

VPN Virtual Private Network 27

XAI eXplainable Artificial Intelligence 5, 7, 120

ZSM Zero touch network and Service Management 3–6, 50, 76, 81

CHAPTER 1

General Introduction

In this chapter, we give an overview of the cellular network evolution and introduce 5G and its related technologies. We present the challenges of threats related to the network slicing concept. In addition, we detail the objectives and the contributions. We conclude with structure of the thesis.

1.1 Context and challenges

Since their initial arrival in the late 1970s, cellular networks have evolved rapidly, with successive generations, 2^{th} generation mobile network (2G) through Fourth generation mobile network (4G), representing significant milestones in the development of mobile connectivity. This evolution was accompanied by a diversified and growing demand, and an overwhelming consumption of data and information. In fact, the cellular communication began with the transmission of voice in 2G, and then the transfer of simple images in 3^{th} generation mobile network (3G). It did not stop here, but it was able to transmit a greater/a larger amount of information such as multimedia videos in 4G. However, driven by emerging technologies, such as autonomous vehicles, Augmented Reality (AR), and the exponential growth of the Internet of Things (IoT), global demand for mobile bandwidth is growing at an explosive rate and 4G networks are already struggling to cope with the demands placed upon them.
In this context, the new fifth generation of wireless communications comes to us faster than can be imagined to deal with the technical limits of 4G. It provides a significant improvement in the perceived Quality of Service (QoS) to users compared to the 4G LTE network and, more importantly, opens new perspectives for the vertical industry [5]. Fifth generation networks (5G) focuses on several key perspectives, (cf. Fig. 1.1) mainly: (i) *Very low latency*, which is a very important metric in the definition of the 5G network. 5G aims to minimize latency up to 1 ms since it promises to support life-critical systems, services with a very low tolerance for delays, and real-time applications. (ii) *Ubiquitous connectivity*, due to the continued exploding demand of data, the 5G network aims to meet the demand for capacity. It also seeks to allow large numbers of device types to connect ubiquitously. Besides, it targets to provide an uninterrupted user experience by achieving a user-centric vision through ubiquitous connectivity. (iii) *High-speed connection*, 5G will greatly improve the high-speed connection with fast data transmission and reception. It aims to achieve download speeds of one gigabit per seconds, and more than 10 times faster than what 4G provided.

To support the different requirements of 5G services in terms of low access latency, high bandwidth, and communication reliability, several research projects addressing 5G have designed new architectures and technologies. The latest development is based on network softwarization and network slicing concepts. The network softwarization relies on three key technologies: Network Functions Virtualization (NFV), Software-Defined Networking (SDN), and Cloud computing. It is based on the concept of architecture, design, deployment, and management of

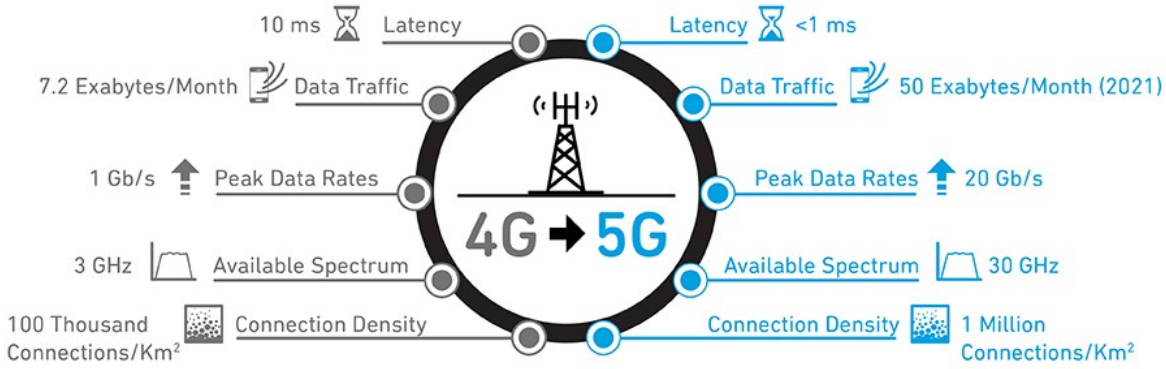


Fig. 1.1 5G Performances [1].

network components, mainly based on the programmability properties of software [6]. Thus, the network softwarization enables adaptability, flexibility, and total reconfiguration of a network, by reducing the cost and optimizing the process of the global maintenance of the network life cycle. On the other hand, network slicing is one of the key enablers of 5G, which consists of creating virtual end-to-end mobile networks tailored to the application needs [7]. It allows multiplexing virtualized and distinct logical networks on the same physical network infrastructure. Through network slicing, 5G supports three types of network slices with specific performance requirements: enhanced Mobile BroadBand (eMBB), massive Machine Type Communications (mMTC), and ultra-Reliable Low Latency Communications (uRLLC) [8]. It is worth noting that network slicing covers the 5G Radio Access Network (RAN), 5G Core Network (CN), and 5G transport network.

Although it is designed to provide significant improvements and solutions to enable the new 5G use-cases, network slicing introduces critical security research challenges and policy directions [7]. The main network slicing challenges include: management of end-to-end network slices, while ensuring security and anonymous transactions, Zero touch network and Service Management (ZSM), network slice isolation, etc. For this purpose, this thesis contributes a study that touches upon various critical security issues in network slicing. In particular, we have investigated the challenges related Service Level Agreements (SLA) management among the different network slicing stakeholders, securing network slices against in-slice and inter-slice attacks, with a special focus on the Distributed Denial of Service (DDoS) and poisoning attacks.

Moreover, distributed machine learning (Federated Learning (FL)), across multi-domain network slice management is needed, in order to predict the network slice performance. However, the aforementioned Artificial Intelligence (AI) mechanisms, in turn, heavily depend on the trained model in the distributed nodes.

That's why it is crucial to design a trust FL framework.

1.2 Objectives and contributions

This thesis has the following objectives:

- Analyze and secure relationships between the key stakeholders involved in the context of network slicing of 5G and beyond networks (tenants, technology domain operators, network slice brokers, etc.). This is critical for slice composition across technological and administrative domains, SLA monitoring and automatic arbitration, robust and accurate data collection, and robust distributed learning. In this context, we aimed to leverage blockchain and smart contract concepts to design trust architectures, maintaining historical information about the interactions between various stakeholders in a secure and non-repudiable way.

- By embracing network slicing, 5G networks can mitigate inter-slice attacks as isolation is one of the key features. Indeed, the isolation ensured by network slicing offers performance guarantees to the applications and ensures isolation, such that attacks (e.g., Distributed DoS - DDoS) remain contained and do not propagate to the network. However, network slices remain still vulnerable against in-slice attacks, where a subset of User Equipments (UE)s, attached to a specific network slice, may generate malicious traffic towards the infrastructure services. A typical example is compromised mMTC devices (i.e., IoT devices) generating a massive number of network attachment requests. In this context, we aimed to design novel approaches leveraging AI techniques, in order to deal with in-slice attacks and hence achieve ZSM vision of B5G network slicing. Specifically, we secured mMTC network slices against DDoS attacks and federated learning-based models against poisoning attacks.

- The ZSM paradigm is featured by the large usage of advanced deep learning algorithms in order to dynamically build efficient decisions. In this context, FL proved its efficiency in not only building collaborative deep learning models among several network slices but also ensuring the privacy and isolation of such network slices. Indeed, FL-based solutions give "machine-centric" decisions about running network slices and their performance, which will be then executed/applied by managers, i.e., slice manager staff/module. However, FL-enabled solutions do not provide any details about why and how such decisions were made, and thus such decisions cannot be properly

trusted/understood by slice managers. To alleviate this issue, we aimed to leverage eXplainable Artificial Intelligence (XAI) in order to make FL-based decisions more trust, transparent, and explainable.

This Ph.D. thesis includes four main contributions, described as follows:

1. In the first contribution, we propose a novel Blockchain-based trust architecture that aims to: (1) create and negotiate the deployment of an end-to-end slice network; (2) manage the SLA established automatically between the involved actors; while guaranteeing security and anonymous transactions.

2. In the second contribution, we propose ZSM solution that addresses the challenges of in-slice DDoS attack detection and mitigation, considering the case of mMTC network slices. Generally, this type of attack targets the 5G CN elements shared among the network slices. The proposed ZSM solution relies on a closed-control loop composed of a triplet (Monitoring System (MS), Analytic Engine (AE), and Decision Engine (DE)), that interacts with the 5G CN in order to detect attacks and automatically react by mitigating the attacks.

3. In the third contribution, we design a novel framework to automatically detect malicious participants in FL process. First of all, based on a real testbed, we generate a realistic dataset that focuses on the latency as service-oriented Key Performance Indicator (KPI) of running network slices. This dataset is then exploited to build an analytic function about the latency prediction, in a federated way. In addition, the basic idea of our framework is to dynamically select one participant as a trusted entity, by leveraging deep reinforcement learning. The trusted participant will then be in charge of identifying poisoning model updates using unsupervised machine learning.

4. Finally, we design a novel XAI-powered framework to explain FL-based decisions. Our framework studies the use of linear and non-linear XAI techniques to identify the most informative features and investigate their impact on the final FL model predictions. Thus, our framework enhances the level of trust, credibility (of the local data/model), transparency, and explanation of the FL-based decisions, while preserving the data privacy, of the different B5G network stakeholders.

1.3 Structure of thesis

In this section, we summarize the content of each chapter of our thesis for the ease of the reader:

- The chapter 2 introduces the state of the art on the 5G network in general and on network slicing in particular, as well as pertinent related works. In this chapter, We begin by describing a few 5G architectures that have been proposed by the research community, along with the primary difficulties that it faces. We then present an appropriate study on the new technologies used to successfully deploy 5G networks and network slicing, such as SDN and NFV. After that, we give a thorough overview of the end-to-end network slicing, including the CN, RAN, and transport network. Finally, We list the 5G network slicing challenges and open issues.

- The chapter 3 is organized as follows: The section 3.1 presents the introduction of the chapter. Section 3.2 introduces the concept of Blockchain and Smart Contracts. Section 3.3 details our proposed trust architecture for an end-to-end network slice negotiation, creation, and SLA management. Section 3.4 presents the performance evaluation of the proposed algorithms and trust model framework.

- The chapter 4 is organized as follows: The section 4.1 presents the introduction of the chapter. In section 4.2, we present a review of related works. We provide general background about the used mechanisms to detect and mitigate DDoS attacks. Section 4.3 details the contributions related to attach events and attack detection, including the devised ML method. We propose a ZSM solution that uses a closed-control loop featuring Machine Learning (ML), to detect and mitigate in-slice attacks on 5G CN components, focusing on DDoS attacks. Section 4.4 presents our attack mitigation algorithm. In section 4.4, we introduce the proof of concept implementation and extensive performance results of the proposed detection algorithm, comparing it with a statistical-based solution. Finally, section 4.7 concludes the chapter.

- The chapter 5 is organized as follows: The section 5.1 presents the introduction of the chapter. In Section 5.2, we present a review of related works. We provide general background about the used mechanisms to detect/mitigate poisoning attacks in FL. Section 5.3 describes the design and specification of the proposed framework. This section presents a novel framework to automatically detect malicious participants in FL process. In particular, our framework is first based on deep reinforcement network to dynamically select a network slice as a trusted participant based mainly on its reputation. The selected participant will then be in charge of identifying the poisoning models updates by leveraging unsupervised machine learning. In Section 5.4,

we evaluate and validate our framework. Finally, section 5.5 concludes the chapter.

- The chapter 6 is organized as follows: The section 6.1 presents the introduction of the chapter. In Section 6.2, we present a review of related works. In Section 6.3, we propose a novel trusted framework, using the XAI algorithms.We evaluate and validate our novel XAI-powered FL-enabled framework. We design a novel XAI-powered framework to explain FL-based decisions. In Section 6.4, we evaluate and validate our novel XAI-powered FL-enabled framework. Finally, section 6.5 concludes the chapter.

- Finally, chapter 7 concludes this thesis and presents different perspectives and future research directions.

CHAPTER 2

State of the Art

In this chapter, we will introduce 5G and its related technologies, mainly network softwarization and network slicing. In particular, we will detail the concept of network slicing and works already defined in the literature.

2.1 5G Networks

The advantages of lower backbone costs, adjustable QoS, the potential for quick and new service development, interoperability between wireless and fixed networks, network management, were formed as the basis for the decision to migrate networking systems from traditional wireless networks to the Next Generation Networks (5G). In comparison to earlier networking systems, 5G plans to provide the highest performance in terms of data throughput, low latency, coverage capacity, and improved security and energy efficiency over spectrum [9]. Numerous organizations, groups, and standardization forums have started the research on 5G, although it has not yet been formally defined and classified [10]. Such 5G research may be motivated by the limitations of earlier technology, namely:

- Provide wireless communication with no coverage edge or density restrictions zone, as well as access rules.
- Support innovative services.
- High inter-device connections should be managed and covered (IoTs).
- Ensure that data is transmitted at higher speeds than in earlier generations.

2.2 5G technologies

The existing network infrastructure is unfit to meet the needs of the 5G network, since it lacks the capability required by the new generation. In particular, network administration for traditional infrastructure is quite difficult and not suitable for 5G networks, because network configuration updates were usually made manually. SDN and NFV therefore propose a developing technology to address the limits of the current networks [11]. Virtualizing entire classes of network functions is made possible by SDN and NFV, which allow network control to be separated from the underlying data planes or switch hardware, accordingly. SDN allows for easy hardware deployment and flexibility when altering network policies, and promotes network innovation and development. Due to its design, which uses a centralized SDN controller, it disrupts the virtual integration between the data and control plane.
Using an open interface like OpenFlow and the centralized network controller, the combination of SDN and NFV offers a global view of the entire network. SDN can accommodate brand-new initiatives and services at every level of user demand or need. Academies and businesses have recently expressed a great deal of interest in SDN and NFV. They serve as a crucial stage in the evolution and creation of

upcoming network infrastructures. In the following subsections, we go over the virtualization technologies listed above in more detail.

2.2.1 Software Define Network, SDN

SDN is created to give existing networks flexibility and simple troubleshooting, while considering the decentralized and complex static architecture of traditional networks. SDN technology is a new framework for managing networks that makes effective network programming and dynamic configuration possible. It attempts to enhance network performance and monitoring while integrating cloud computing more seamlessly into conventional network administration. To consolidate network intelligence into a single network component, SDN divides the forwarding of network packets (data plane) from the routing process (control plane). One or more controllers compose the control plane. They are viewed as the SDN network's brain, in which all intelligence is embedded. The fundamental problem with SDN is that intelligent centralization has its own disadvantages, in terms of scalability, security, and elasticity.
There are three layers in the standard SDN architectural paradigm, as follows:

- Northbound Application Programming Interface (API)s allow SDN applications to describe their demands for traffic control in the underlying networks.
- The application plane and data plane are connected via the SDN controller, which oversees the control plane. It converts the demands of the apps into suitable forwarding rules that the underlying network switches can implement. The SDN controller can access the features offered by the SDN-capable switching devices thanks to the southbound API. These tasks could involve reporting network status and controlling packet forwarding rules.
- The data plane includes the network components (such as routers and switches) that gather network status data and handle packets in accordance with instructions from the SDN controller, such as traffic statistics and network topology.

2.2.2 Network Function Virtualization, NFV

A number of significant service providers developed NFV technology to make the switch from a hardware- to a software-oriented architecture. Software features hasten the rollout of new network services and boost sales, while reducing operational costs. Essentially, NFV technology is a new sort of manipulable infrastructure. It seeks to virtualize network functions like load balancers, Firewall web application

transcoders, Application Distribution Controllers (ADCs), etc. With virtualization, the software may be designed more flexibly. Since the current networking services are provided by a variety of network functions that are integrated in a static way, NFV provides additional dynamic strategies to develop and manage network functions.

The virtual network function is the main NFV concept. Specific network functions that are run on one or more Virtual Machine (VM)s on the hardware network infrastructure are managed using Virtual Network Functions (VNF) (switches, routers, etc.). To offer excellent network communication service, these VNFs can be joined or combined as different components. They are made to supply network components and consolidate resources to support a fully virtual environment. Furthermore, E2E services should be unaffected by the efficient deployment of VNF instances at the carrier level.
The three main distinctions between NFV and existing practice are introduced below:

- Operating dynamically: As a result of the decoupling of network functions, network operators have a great deal of flexibility when scaling the NFV performance.
- Software and hardware separation: It makes it possible to separate the software from the hardware.
- Flexible network function deployment: NFV automatically deploys network function software on a set of hardware resources that may run various functions at different times in multiple data centers.

As shown in Fig. 2.1, there are three key functional blocks in the NFV architecture:

- NFV Infrastructure (NFVI): The hardware and software resources that compose the NFV environment are all included in the NFV infrastructure. NFVI covers inter-location network connectivity, such as that between data centers and public or private hybrid clouds. Computing, storage, and networking hardware are all examples of physical resources. The virtualization layer, which is just above the hardware and summarizes the physical resources, provides processing, storage, and communication for NFVs.
- NFV-MANO: The orchestrator, VNF managers, and virtualized infrastructure managers constitute NFV Management and Orchestration (MANO).

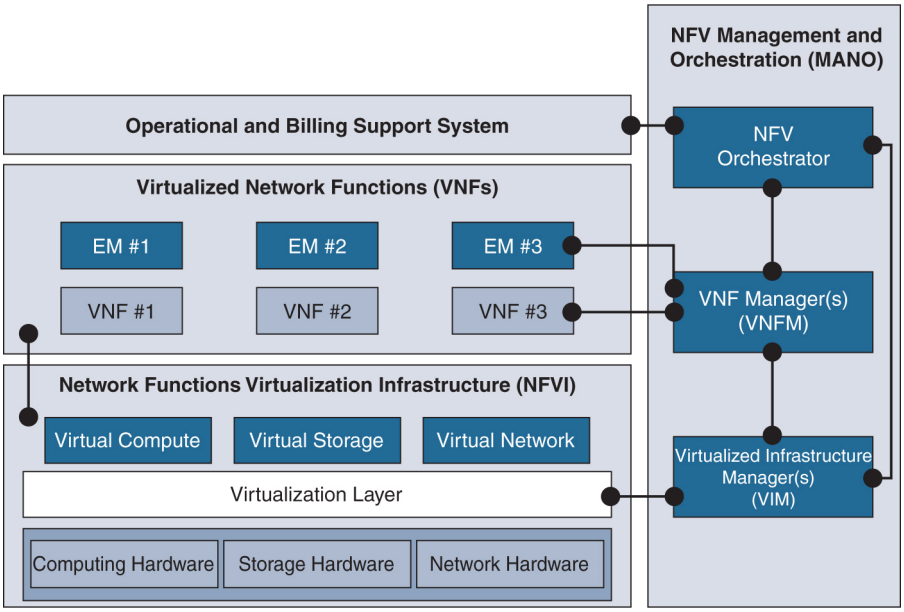


Fig. 2.1 NFV architectural framework [2].

These blocks offer the functionality needed for VNF management operations, including provisioning and configuration. In order to facilitate infrastructure virtualization and VNF lifecycle management, NFV-MANO offers orchestration and lifecycle management of physical or virtual resources.

- Services: A set of VNFs is a service. It can be set up in a single virtual machine or several. In some conditions, VNFs can directly run on hardware or on installed virtual machines. They are controlled by virtual machine monitors or native hypervisors. An EMS (Element Management System) manages the construction, configuration, monitoring, performance, and security of a VNF. An EMS offers the crucial data that the Operational Support System needs (OSS). The Business Support System (BSS) and OSS are the overarching management systems that assist providers in deploying and administering a variety of E2E communication services.

2.2.3 Open Radio Access Network,

The ORAN standards will change the confidential nature of the RAN market, where RAN vendors have proprietary equipment and software. ORAN separates software and hardware, known as disaggregation. You no longer need to use specific software for particular hardware. Therefore, an O RAN architecture makes the network more flexible and interoperable. The engineers are developing the ORAN technology using virtual RAN (vRAN) principles and technologies simply because vRAN has improved network malleability and security and reduced CAPEX and OPEX costs.

The O-RAN architecture is well documented in the O-RAN alliance. The key elements of the O-RAN architecture are:

- Service Management and Orchestration Framework (SMO)—includes an integration fabric and data services for the functions it manages. It allows managed functions to interoperate and communicate within the O-RAN. The SMO connects to and manages the RICs, O-Cloud, the O-CU, and O-DU.

- RAN Intelligent Controller (RIC) – There are two types of RICs – non-real-time and near-real-time. Both are logical functions for controlling and optimizing the elements and resources of an O-RAN. A near-real-time RIC controls and optimizes elements and resources with granular data collection and communication over the E2 interface. The E2 interface connects the near-real-time RIC with the O-CU and O-DU. O-Cloud is a cloud computing platform made up of the physical infrastructure nodes using the O-RAN architecture. It also creates and hosts the various virtual network functions (VNFs) used by the RICs and other infrastructure elements.

- O-RAN central unit (O-CU) – Logical node that hosts a handful of protocols, which are the radio resource control (RRC), service data adaptation protocol (SDAP), and packet data convergence protocol (PDCP).

- O-RAN distributed unit (O-DU) is a logical node that hosts another set of protocols, which are the radio link control (RLC) protocol, medium access control (MAC) protocol, and the physical interface (PHY).

- O-RAN Radio unit (O-RU) – It processes radio frequencies received by the physical layer of the network. The processed radio frequencies are sent to the O-DU through a front haul interface.

2.3 Network slicing

Recent technological advancements like network slicing are important to the introduction of the future 5G network generation. It is an emergent paradigm that is considered as the basis of 5G networks. In [10], the Next Generation Mobile Network (NGMN) Alliance defines network slicing in the context of 5G. It is described as a technology that makes it possible to combine physical and/or logical network resources with cloud computing resources in a multi-tenant, open software-oriented network environment. It entails deploying a number of independent logical networks on top of a shared and common physical infrastructure

platform, hence enabling a dynamic ecosystem of stakeholders, that fosters technological and commercial innovation. Network slicing, a technology that the 3GPP introduces [12], enables the operator to design customized networks that offer optimized solutions for various requirements, in different market circumstances. Network slicing, according to ITU-T, consists of dividing and sharing virtual resources based on a programmable control and data plane [13]. Network slicing offers networking, radio, and virtual resources (i.e., VNF), enabling real service differentiation and specialized network management. As a result, it creates value for application providers, third parties without a physical network infrastructure, and verticals. A network slice is comprised of a number of VNFs, that can have different functionalities depending on the slice's specific service requirements. Each slice has a different number of resources depending on the type of service it offers.

An E2E network slice is composed of three sub- slices: core network, RAN, and transport. The authors of [14] describe the architecture of the high-level network slicing, in accordance with the 3GPP specifications of [15], as follows:

2.3.1 Slicing the Radio Access Network, RAN

Each network slice utilizes a frequency spectrum or a set of resources specifically designated for it, and the slice resource management models are dependent on the necessary level of isolation. The MAC scheduler, control plane, user plane, and spectrum are the isolated dedicated resources in the RAN. Slices can share the control plane, spectrum, and MAC scheduler, thanks to the shared resource concept. The issue of spectrum sharing among Mobile Virtual Network Operators (MVNO) in the RAN has been addressed in a number of academic studies. The authors of [16] provide a method for virtualizing an LTE eNB, that is based on a hypervisor, taking the traffic volume and radio circumstances into consideration. The management of wireless resources for RAN sharing has been described in depth by the authors of [17]. Additionally, network virtualization substrate is a notion that was presented in [18], that permits flexible resource allocation by altering the MAC scheduler.

2.3.2 Slicing the core network, CN

The increased throughput need should be addressed by 5G, which should also offer more elasticity, scalability, and adaptability. The CN architecture established by earlier generations aims to be better adaptable to these new 5G network specifications. In order to accomplish these specifications, the 3GPP in [19] introduces a new CN design, named Next Generation core, where they rearrange the EPC entity into more fine-grained network functions. According to 3GPP specifications,

the new 5G core employs a service-based architecture, that is cloud-aligned and handles all 5G functions, interactions and services, including security, authentication, session management, and traffic aggregation from end devices. Additionally, the 5G new core places emphasis on NFV as an integrated design concept with software capabilities, that can be deployed using the MEC infrastructure [20]. Some novel network functions that were not present in LTE are defined, by the current generation core architecture, although LTE-specific versions of some services are also retained.

The following are the main network functions listed in the new CN, as shown in Fig.2.2:

- Acess Mobility Function (AMF) is responsible for registration management, connection management, mobility management, access authentication, and authorization management.
- Session Management function (SMF) to manage session establishment, UEs' IP address allocation, and traffic steering configuration.
- User plane Function (UPF) in charge of packet routing and forwarding, packet inspection, and QoS handling.
- Policy Control Function (PCF) for providing policy rules to Control Plane (CP) functions, and access subscription information for policy decisions.
- AUthentication Server Function (AUSF) acts as an authentication server.
- Application Function (AF) supports application influence on traffic routing, and the interaction with policy framework for policy control.
- Network Exposure Function (NEF) for exposure of capabilities and events, secure provision of information from external application to 3GPP network, translation of internal/external information.

2.3.3 Slicing the transport network

In order to meet the networks' increased bandwidth demands, the transport network in 5G is completely redesigned. The integration of the backhaul and fronthaul segments on the same transport substrate and the inclusion of deep programming in the transport network are the foundations for the transport network's evolution.

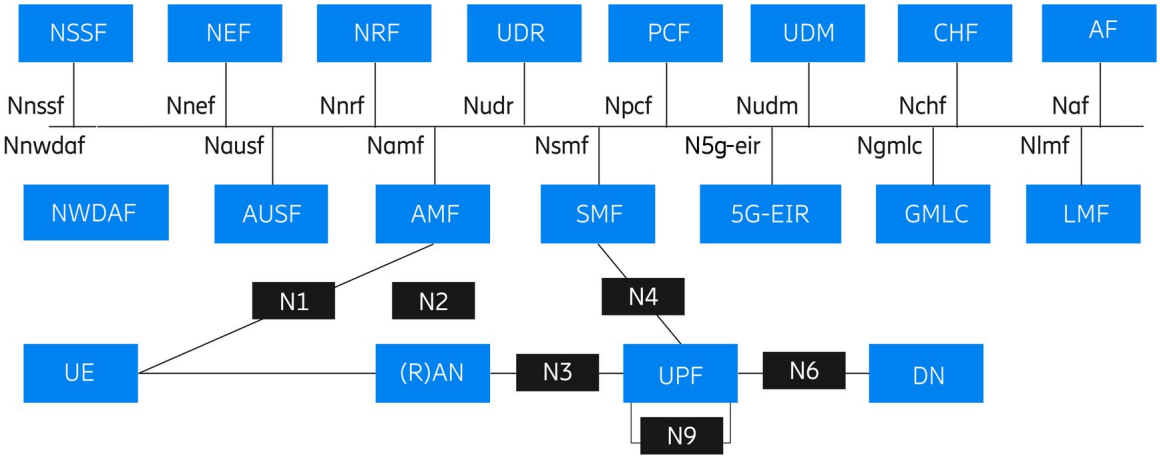


Fig. 2.2 The new core architecture [3].

2.3.3.1 Fronthaul/Backhaul and network slicing

Fronthaul refers to the network that connects remote radio heads to Baseband units (BBU)s, and the Backhaul represents the link between the Centralized Unit (CU) and the core network. The upcoming 5G Backhaul/Fronthaul must be capable of integrating different transmission technologies, such as optical Ethernet, IP, and millimeter-wave, in order to satisfy the 5G requirements. Additionally, while considering the needs of the use case and the network conditions, a flexible split of base stations is crucial. The split of base station functions affects the roles of backhaul and fronthaul, resulting in an integrated transport network architecture that mixes both the flexibility and reflecting of RAN centralization. The authors of [21] introduce an architecture based on the development of an overlay network. It is a combined control plane and data plane-based upstream and downstream transport architecture. Network nodes that can combine several upstream and downstream transport technologies using the same data frame are necessary for it.

An architecture based on merging an optical and wireless backhaul/fronthaul was presented by the authors of [22]. Moreover, the network circumstances and service requirements determine the functional distribution of RAN and the level of centralization. Additionally, implementing network slicing can be a crucial enabler for ensuring performance and isolation between various logical networks, that employ various functional RAN distributions. In this context, network operators employ slicing techniques and jointly optimize VNF embedding based on the links' availability and the transport network's architecture.

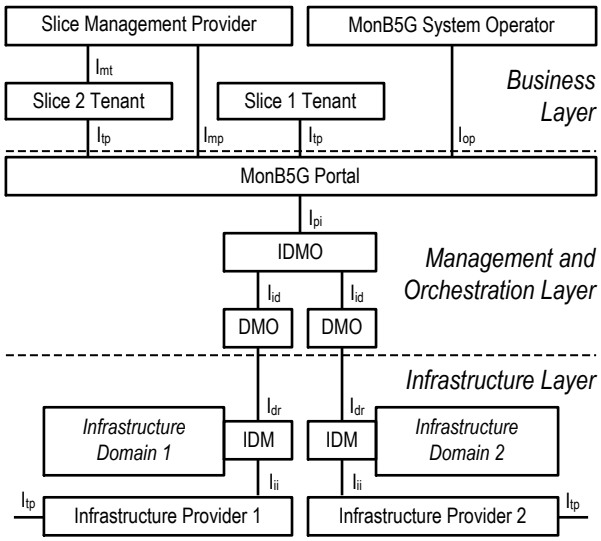


Fig. 2.3 Components and business actors of MonB5G architecture [4].

2.4 Scalable management and orchestratıon of network slices

Future network slicing is anticipated to support a large number of heterogeneous vertical services over a shared infrastructure. To provide seamless support for contemporary telco-based services, high levels of automation, flexibility, and programmability are becoming important architectural components. In [4], the MonB5G project presents a novel management and orchestration framework for network slices. The proposed framework includes a distributed and programmable AI-driven management architecture, and offers a highly scalable solution for network slicing management and orchestration. Its main objective is to give network slices a highly scalable and efficient management plane to enable quick responses to events.
The framework has three defined layers, as illustrated in the Fig.2.3:

- *Business Layer:* Through the MonB5G Portal, it communicates with the Management and Orchestration Layer and relevant business stakeholders. The actors include system operator, slice tenants and slice management providers. Based on the same template, the latter (slice tenant) may operate or manage multiple network slice instances.

- *Management and Orchestration Layer* comprises the MonB5G System Portal as well as domain and intra-domain orchestrators. The MonB5G system operator administers this layer.

- *Infrastructure Layer*, which combines physical and virtual resources with a resource-focused management platform. Infrastructure Providers manages this layer.

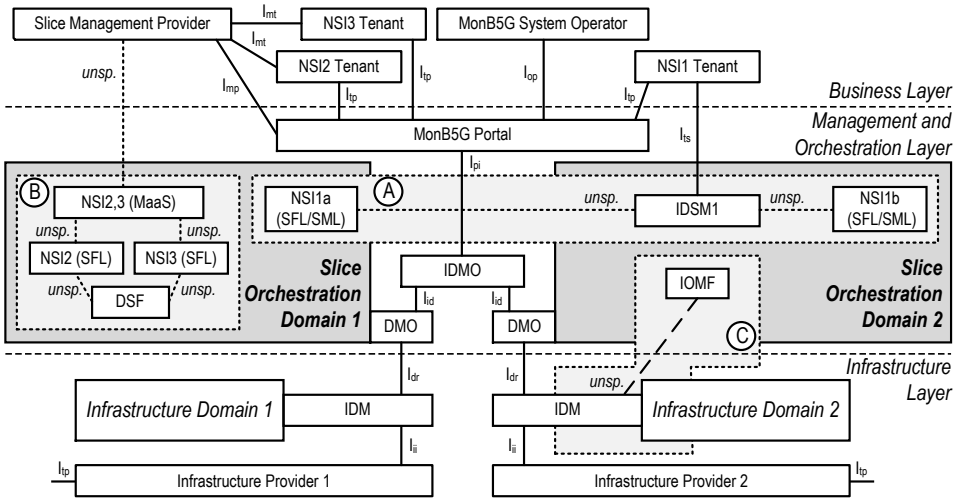


Fig. 2.4 Multi-domain slice management [4].

The proposed slice management approach can also be used for end-to-end slice management, when slices are implemented across multiple domains as shown in Fig. 2.4. In such a case, the entity called Inter-Domain Slice Manager (IDSM) is responsible for the end-to-end slice management. It interacts with SMLs of all domain slices (DSM), that compose the E2E slice. The IDSM is a part of the slice template (a set of VNFs), and in some cases, it can be generated automatically by the IDMO. When IDSM is in use, it provides to the slice tenant with the management interface. The IDSM is also responsible for the calculation of slice-related KPIs. Fig. 2.4 shows a multi-domain slice, with the IDSM is deployed (Domain 1 and Domain 2). The single domain slice is the internal structure of a self-managed. It includes functions for monitoring (Monitoring System Function (MS)), anomaly detection (Analytic Engine Function (AE)), and decision-making (Decision Engine Function (DE)).

2.5 Blockchain in the 5G networks

A blockchain is a type of distributed ledger technology (DLT) that consists of growing lists of records, called blocks, that are securely linked together using cryptography. Each block contains a cryptographic hash of the previous block, a timestamp, and transaction data . The timestamp proves that the transaction data existed when the block was created. In [23], Blockchain was used to build a verification platform between IoT devices and Baseband Unit (BBU) where user access information is stored on the chain and the Smart Contract is used to perform automatic user authentication. Similarly, the authors in [24] apply Blockchain to build a trusted authentication architecture for Cloud Radio Access Network (Cloud-RAN) in the 5G era. In [25], the authors propose an architecture

for the core network of Long Term Evolution (LTE), where a distributed database is used instead of having a centralized database Home Subscriber Server (HSS) to maintain subscriber information. Indeed, when an operator configures a new Subscriber Identification Module (SIM) card, the SIM information is distributed within the Consortium Blockchain through a new block. Then, this block will be available to all the nodes (core networks) in the chain. Furthermore, when a user performs any activity such as subscribing to a new offer or charging the SIM card, this information is stored in another block. After its validation by the other nodes, the block will be published to all the other nodes within the chain as a copy in the distributed ledger. The ledger can have the same role as HSS to store user subscription information. When a user sends an attach request to Evolved Packet Core (EPC), the latter retrieves the information of Mobile Subscriber Identity (IMSI) from the message and performs a null transaction to the Smart Contract. Then, the Smart Contract will check if the user belongs to one of the operators who are already in agreement with Consortium Blockchain. Hence, the Smart Contract acts as distributed Self Organizing Network (SON) features to handle self-transactions among mobile operators in return for sharing small cells' infrastructure.

2.6 Machine/Deep Learning techniques in the 5G networks

Machine/Deep Learning (ML/DL) is expected to be required for 5G networks and new use cases. In general, ML/DL algorithms can analyze enormous amounts of data, find abnormalities, foresee future events, and quickly adjust to changing settings.
ML can also enhance and automate network management, within the network environment, with the use of such functionalities. In fact, there are several scenarios where ML/DL can be advantageous, including fault management, coverage performance, maintenance, network design, and QoS prediction. In general, ML/DL algorithms can be categorized into three groups:

- Unsupervised Learning: The data used to train the algorithm in unsupervised learning is not labeled. As a result, without any instruction, it attempts to identify subgroups of the data that share comparable traits. The unsupervised learning technique, for example, can be helpful when identifying an anomaly or issue in wireless networks [26].

- Supervised Learning: supervised learning is helpful for categorizing or predicting tasks, using a labeled dataset.

- Reinforcement learning: reinforcement learning (RL) algorithms are useful in stochastic environments under uncertainty. The RL algorithm is used to find the optimal policy by trying possible actions and learning from the feedback in a given state [26].

ML/DL algorithms have also shown considerable advances in boosting communication reliability for a variety of applications, including radio resource allocation, physical security, signal decoding, and channel estimation [27]. Recent research has also demonstrated that ML algorithms, such as Bidirectional LSTM, KNN, and Random Forest [28], can beat conventional techniques in this regard in terms of estimating the performance of 5G networks.
In [29], the authors proposed a new framework, called BoostIDS, that leverages ensemble machine learning to efficiently detect and mitigate security threats in SDN-based 5G systems. BoostIDS comprises two main modules: (1) Monitoring module in order to optimize training time complexity; and (2) Ensemble learning-based threats detection module that implements a Lightweight Boosting Algorithm (LBA). Moreover, [30] developed a distributed approach that uses multilayer support vector machines (SVMs) to detect anomalous behaviors in the 5G network. Authors in [31] developed an Iot attack detection approach based on three common ML models, namely Decision Tree (DT), Naive Bayes (NB), and AI neural network to detect cyber attacks in 5G networks, including Domain Name System (DNS) and Message Queue Telemetry Transport (MQTT) based attacks.

CHAPTER 3

An end-to-end trusted architecture for network slicing in 5G and beyond networks

3.1 Introduction

The evolution of 5G comes to us faster than can be imagined offering a great opportunity for new business model innovations [32]. Network Slicing is one of the key enablers of 5G, which consists of creating virtual end-to-end mobile networks tailored to the application needs [33, 34]. The network slicing concept allows multiplexing virtualized and distinct logical networks on the same physical network infrastructure. Through network slicing, 5G supports three types of network slices supporting services with specific performance requirements: eMBB, mMTC, and uRLLC [35, 36]. Usually, an end-to-end network slice comprises three main sub-slices: RAN, Core, and Transport networks [33] corresponding to three distinct Technological Domains (TD). Therefore, to deploy an end-to-end network slice, the Slice Provider (SP) has to lease resources for each sub-slice on each TD and from different Resource Providers (RP)s.

First of all, the Vertical (V) chooses the network slice type that perfectly matches its needs, requesting the creation of a new network slice using a blueprint or a network slice template provided by the SP. Next, this template will be translated by SP into specific slice resource requirements for each sub-slice component or TD, such as needed computing resources, network resources, radio resources, etc. SP uses a resource broker mechanism to select which RP will deploy a sub-slice for each TD. The selection process relies on SP's specific algorithm, which considers several criteria [37], such as cost, QoS, reputation, etc.

Once the needed resources by each sub-slice are allocated, meaning that the end-to-end network slice is ready to be deployed, SLA contracts are signed between V and SP as well as between RPs and SP. The SLA specifies what customers can expect from SP. SLA is used to check if a defined service is delivered as contracted and helps managing QoS degradation. Indeed, SLA defines QoS requirements such as bandwidth, throughput, and latency, without specifying the technology to be used to deliver a particular service. Moreover, SLAs include different sections on the concerned parties, the service fee, the validity period, and the compensation value used in case of SLA is not respected (i.e., when the performance levels are not met). In 5G, SLA should indicate the QoS related to the pre-defined slice types, i.e. mMTC, eMBB, and uRLLC.

In this chapter, we propose a trust architecture to create and manage end-to-end network slices in 5G ensuring security and anonymous transactions. The proposed architecture relies on Blockchain and Smart Contracts to (1) negotiate and deploy an end-to-end network slice; (2) manage automatically SLA signed between all the involved actors, the slice tenant, SP, and RPs. First, we propose a trust architecture model that allows the resource broker, used by SP, to select the appropriate

RP that will deploy a sub-slice for each TD. The resource selection is formulated as an Integer Linear Program (ILP), where the aim is to maximize the RP's reputation while minimizing the cost.
Once the RPs are selected and SLA established, the proposed trust architecture framework allows monitoring the KPI of the deployed end-to-end network slices, in order to verify if SLAs are violated by RPs, and hence automatically compensate V and SP. Finally, the RP reputation is build using the information on the SLA violation; RP's reputation increases if it has respected the established SLA, otherwise it decreases. Computer simulation has been used to validate the proposed framework in terms of (1) resource negotiation and RP selection to deploy an end-to-end network slice; (2) ability to detect SLA violation and compensate both V and SP.

A previous work [38] has introduced a Blockchain-based brokering mechanism to deploy end-to-end network slices. Although we share the same concept of using a Blockchain-based broker, in this chapter we devise a selection algorithm based on RP reputation, which is an important criterion to establish trust relations among the involved actors. Regarding SLA management, the work in [39] introduces SLA trust management in the context of a simple scenario of a Cloud Resource Provider and a tenant. Despite the fact that we share the concept of using Blockchain, we address in this chapter a more complex scenario where not only one SLA has to be managed, but different SLAs involving different stakeholders.

The next sections of this chapter are organized as follows: section 3.2 introduces the concept of Blockchain and Smart Contracts. Section 3.3 details our proposed trust architecture for an end-to-end network slice negotiation, creation and SLA management. Section 3.4 presents the performance evaluation of the proposed algorithms and trust model framework.

3.2 Blockchain and Smart Contract

Blockchain [40] is a chain of blocks, in other words a database that stores information about transactions like date, persons participating in transactions, time, and amount transferred. A copy of the Blockchain is spread over many computers. These computers are called nodes constituting a network. It also stores unique codes called "hash" which are cryptographic codes created by special algorithms such as "Proof of work" used in Bitcoin transactions [41]. Actually, when a transaction occurs, it will be checked by all nodes. For each transaction, a new block is created. This block is sent to every node in the network. If a transaction is approved by the majority of the nodes using a special algorithm, then the block

is added to the existing Blockchain.

More than that, Blockchain platforms are of different types: (*i*) Permissioned, which means that the Blockchain network is private and its access can be granted by particular participants, and the management of the network is typically done by a specific administrator; (*ii*) Permission-less, which means that the Blockchain network is public; or (*iii*) Consortium, which is a hybrid of permissioned and permission-less [42]. Additionally, Consortium or hybrid Blockchains are similar to the permissioned networks; but the management of the chain is carried out by multiple administrators.

Blockchain can be used in several fields, such as healthcare, IoT [43], online shopping, etc. Likewise, Blockchain implementation has extended far beyond the technology's applications in cryptocurrencies, moving toward more mainstream adoption and launching new opportunities. Take the example of smart cities, where the Blockchain can be used for user authentication [44]. We can especially mention the case of Estonia, where citizens or "e-residents" can access several public services by using a Blockchain-based digital IDentity (ID) card, while all-access history to the public's health record is registered in the Blockchain. Also, it is used for medical information management.

On the other hand, a Smart Contract is a feature associated with Blockchain. The definition of a Smart Contract based on Nick Szabo, taken from the original publication [45], is an agreement between several parties in the form of computer code. It allows automated transactions when one or more contract conditions are met without resorting to a third party. They are distributed and, therefore, stored in a public database that cannot be modified, such as a Blockchain. Plus, Smart Contract can control valuable things like the balance of user account (number of Ether (ETH)) or other parameters, as well as the transfer of value among users. ETH is a unit of currency, serving to compensate the nodes for storage and processing of Smart Contracts (1 Ether equals 213.20 EUR, 2021).
Smart contracts are used in Decentralized Application (DApp), which is a serverless peer-to-peer application that can run on a particular Blockchain written for a specific application. Moreover, Smart Contract is a self-executing contract where the terms of an agreement between the buyer and the seller are previously agreed before and are directly written into lines of code. Similar to a programming language, Smart Contract is a collection of functions and data residing at a specific address on the Blockchain.
In the same way, data can be queried or altered by calling functions of the Smart Contract or by making a transaction to it. These functions are also executed automatically on every node in the network according to the data included in the triggering transaction [46]. Then, the Smart Contract's execution outcome

is validated and agreed on by all distributed and trusted nodes of the network. Moreover, DApps are frontend user apps that interact with the Smart Contracts by initiating specific transactions that call functions within the Smart Contract.
Several works advocate for the usage of Blockchain and Smart Contracts in 5G. In [47], the authors summarize the key opportunities offered by Blockchain and Smart Contract technologies in 5G networks. The authors propose using Blockchain-based security techniques for 5 scenarios in 5G. The first scenario, "5G Infrastructure for Crowdsourcing", allows smaller infrastructure investors to roll out cellular towers that will be a part of the overall operator's infrastructure. These investors are registered, managed, and also automatically paid upon the use of the resources implicating secure technologies such as Blockchain and Smart Contracts. The second scenario, " 5G Infrastructure Sharing", in which a seller Mobile Network Operator (MNO) offers telecommunication services either (*i*) cellular towers or (*ii*) a subset of these towers. Here, Blockchain can be used in managing and tracking resource usage and sharing. The third scenario, "International Roaming", is one of the challenging issues in the telecom sector as it involves brokers and third parties to settle down payment, and charges rules among them. In 5G, many parties are going to be involved in the usage of networks. These parties may include multiple operators, intermediary networks, and international intermediary exchanges.
Smart Contracts can be implemented to accomplish a Blockchain-based payment and roaming in which charges and consumption per usage are recorded and tracked. The fourth scenario,"Network Slicing (NS)", where a Network Slice Broker (NSB) is used to perform network slicing by exposing service capabilities of the mobile operator network during the brokering mechanism. A Blockchain and Smart Contract with decentralized storage, like Storj[1] or IPFS[2], can be used to replace NSB functionalities. The fifth scenario is "Management and Authentication of mMTC/ uRLLC" where millions of IoT devices are expected to be connected, and some scenarios may require few milliseconds of latency. Incorporating such a large number of different IoT devices opens up the possibility for new business models and services to be offered to mobile customers. Blockchain and Smart Contracts, with decentralized storage, are more powerful and attractive alternatives to these centralized operators, in which such management functionalities can be performed in a decentralized manner.

[1]https://storj.io/
[2]https://ipfs.io/

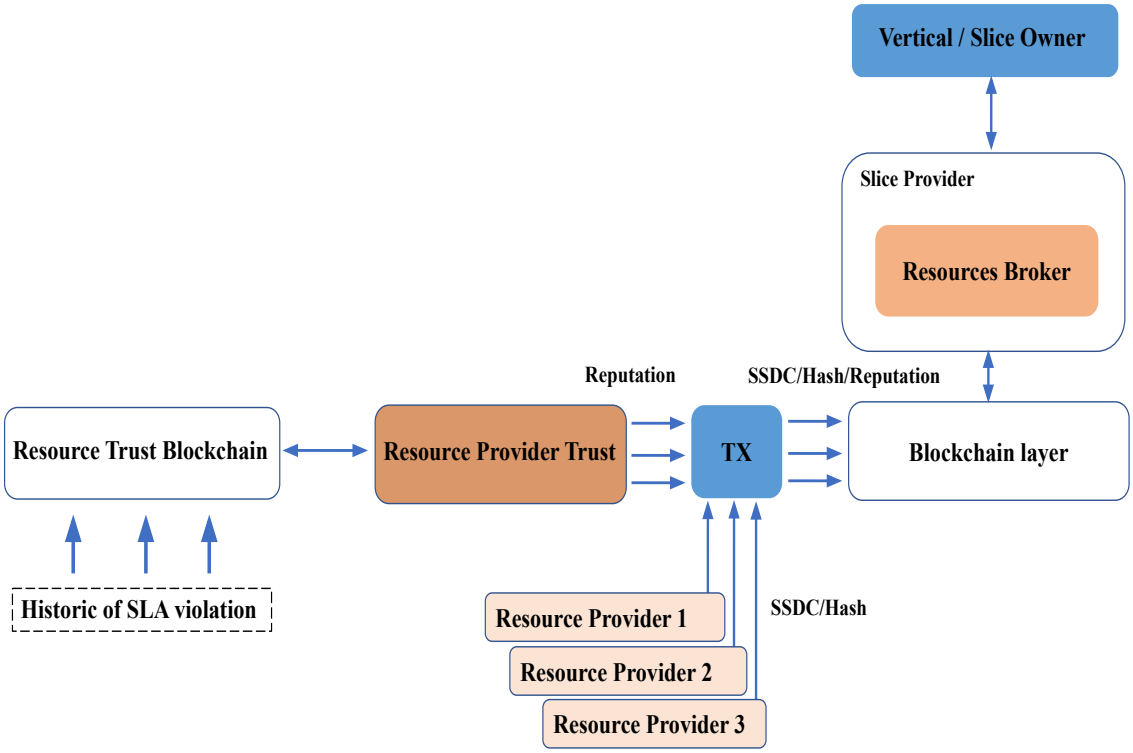


Fig. 3.1 Trust architecture for end-to-end network slice creation.

3.3 Proposed Trust Architecture

As stated earlier, there are two separate steps to create and manage an end-to-end network slice. Firstly, SP selects the RPs in which the different sub-slices composing the end-to-end network slice requested by V will be deployed. This step ends when the SLAs are signed between, on one hand the SP and V actors and on the other hand, between the SP and RPs actors. The second step concerns the SLA management, which consists of monitoring the SLA violation and applying for penalties and compensation if deemed appropriate. For both steps, we are proposing a trust architecture to guarantee the well-functioning of such a complex system that involves different stakeholders and actors.

3.3.1 Trust architecture for end-to-end network slice negotiation and creation

3.3.1.1 System overview

It is well established in 5G that a network slice is composed of sub-slices (virtual or physical resources) that belong to different TD; i.e., RAN, Core Network, Data Center domain. An end-to-end network slice is composed of VNF and Physical Network Functions (PNF) deployed on different technological domains and stitched together to build an end-to-end slice through specific mechanisms or protocols. When a V requests the creation of an end-to-end network slice to SP, the

latter decomposes it to sub-slices and negotiates for each sub-slice the necessary resources from the TD Resource Providers. This procedure is defined in [38] and is as follows. V or the slice owner requests the creation of a network slice using a template or a blueprint. This template may contain high-level information. The SP will translate the template to specific slice resource requirements, such as the number and types of sub-slices, PNF, VNF, Central Processing Unit (CPU), I/O, memory, storage, etc. The sub-slice components are translated to resources of a TD. Actually, these resources of a TD can be computing (such as CPU, I/O), storage, radio evolved Node B (eNB), Central Unit (CU), Distributed Unit (DU), Remote Radio Head (RRH) / Remote Radio Unit (RRU)), and transport (e.g., Virtual Local Area Network (VLAN), Virtual Private Network (VPN). We assume that a slice is composed of several TDs, noted as $NS = \{TD_1, TD_2, TD_3, \ldots, TD_n\}$. For each TD, SP describes the needed resources according to the slice type. For instance, in the computing resource domain, they could include the number of CPUs, number of VM or container instances, etc. For the radio domain, resources could be related to the functional split type, the Medium Access Control (MAC) scheduler algorithm, the number of Physical Resource Blocks (PRB), etc. On the other hand, transport domain resources may include the type of link (bandwidth, latency), number of VLANs, front haul link capacity, VPN links, QoS, etc. We define $R(TD_i) = \{p_1, p_2, p_3, \ldots, p_m\}$, as the set of parameters requested by TD_i. In this context, a trusted model is needed to guarantee the creation and the instantiation of the sub-slices to build the end-to-end network slice, particularly knowing the presence of different business actors, namely V or slice owner, SP, RP.

We illustrate in Fig. 3.1 a novel trust architecture that allows negotiating and deploying an end-to-end network slice. The proposed architecture includes all the above-mentioned actors in addition to a third-tier entity that creates and maintains the reputation of RP, namely Resource Provider Trust. SP owns the Blockchain layer, which is used to generate and negotiate contracts between the slice and RPs. The negotiation between the SP and RPs is done per sub-slice. The sub-entity at the SP in charge of this negotiation is the resources broker. The latter selects the different RPs, per TD, that maximizes a specific objective function. An example of the latter could be the reduction of the deployment cost [38]. In this chapter, we envision a sub-slice deployment brokering mechanism as series of small contracts. Each contract has a unique identifier and some data fields and can also perform actions such as creating a new contract or updating the Blockchain state. Contract actions are triggered by on-chain data updates (i.e., the creation of a new contract). Each sub-slice generates a contract. The end-to-end slice is ready for deployment once all the contracts regarding its sub-slices are negotiated and finalized.

3.3.1.2 Sub-slice Brokering

Once each sub-slice is described in terms of resources (i.e., R($\boldsymbol{TD}_i$)), the request for each sub-slice is sent to the Blockchain layer. Each sub-slice will generate a contract to be negotiated. Once the query for a sub-slice arrives at the Blockchain layer, a Sub-Slice Contract (SSC) is created and published. This contract specifies the necessary resources needed by the sub-slice (i.e., R($\boldsymbol{TD}_i$)) and the duration of the sub-slice. The different RPs are notified of the new SSC. Here, all the SSCs are visible on the Blockchain. RPs respond by publishing Sub-Slice Deployment Costs (SSDC), which specify the cost that the RPs are willing to charge for providing the necessary resources for each component of the sub-slice R($\boldsymbol{TD}_i$).
The Resource Provider Trust module records the trust value of the corresponding resource in the SSDC as appended information in the Blockchain layer. The original SSC collects all the related SSDC and arbitrates according to specific objectives (e.g., cheapest, best in terms of quality, or other criteria). All other contracts are terminated, and the winning contract is used to deploy the sub-slice components. All the related information about the sub-slice deployment are recorded in the Blockchain managed by SP. Relevant information on the different interfaces allowing to access to the sub-slices, such as the stitching interface (the Resource Orchestrator (RO) interfaces and their description), are compiled in a Sub-Slice Deployment (SSD) document.

3.3.1.3 Sub-Slice Deployment

Once the resources are negotiated and transcribed in SLAs, which are established, on one hand, between the SP and V and on the other hand, between the SP and each RP. Details on how the SLA is established using a trust architecture will be described in section 3.3.2.
Once everything is settled, the Slice Orchestrator (SO) handling the Life Life Cycle Management (LCM) of the deployed network slices, will use the RO interfaces indicated in each SSD to: (*i*) instantiate and create the sub-slice; (*ii*) stitch the sub-slice with the other sub-slices to build the end-to-end network slice. Note that each RP uses its domain RO to manage and orchestrate its resources. RP exposes interfaces to allow other ROs or the SO to interact with the local RO.

3.3.1.4 Resource Provider reputation

The Resource Provider Trust module relies on the precedent experiences with RP, particularly on the respect of the SLA signed with a SP for a served TD resource. Indeed, a RP manages and provides resources for more than one TD. To this aim, the Resource Provider Trust module monitors all SLA signed between a SP

and a RP for sub-slices deployment. According to the monitored information on SLA, a $Rep(i, j)$ of RP_i for serving TD_j is maintained and updated. The $Rep(i, j)$, and hence the trust of RP when providing a resource of TD_j, is a function of the respected SLA by RP_i. The reputation of a RP increases if a signed SLA by RP is respected and decreases otherwise.

$$Rep(i, j) = min(\Re p_{max}, Rep(i, j) + \triangle R) \text{ if the SLA is respected.} \tag{3.1}$$

$$Rep(i, j) = max(\Re p_{min}, Rep(j, i) + \alpha.\triangle R) \text{ if the SLA is not respected.} \tag{3.2}$$

Where $\triangle R$ is the reputation increase step, $\alpha.\triangle R$ is the reputation decrease step, $\Re p_{max}$ is the maximum value of the reputation, and $\Re p_{min}$ is the minimum value of the reputation. Details on the SLA monitoring will be introduced in the coming sections.

3.3.1.5 Algorithm of the resource selection

As stated earlier, the SP relies on a resource broker to select RPs that will serve the different sub-slices for each TD. In this chapter, the resource broker will implement a local algorithm that selects the best offer from the received SSDC. The proposed algorithm uses two criteria for the selection, namely, cost and Resource Provider reputation. The cost is included in the SSDC, while the reputation is obtained from the Resource Provider trust entity (i.e., $Rep(i)$).

The problem of selecting the RP for the resource of TD is to find an optimal trade-off between reducing the cost of deployment while increasing the Resource Provider's reputation. We formulate this problem using a Linear Program (LP). First, we denote by $C_{i,j}$ the cost of TD_j resource of $RP_{(i)}$, and $Rep(i, j)$ is the reputation of $RP_{(i)}$ when serving resource of TD_j. Second, we assume that the network slice is composed of K sub-slices that need to be deployed using K TD resources. In addition, we assume that L RPs have responded to the SSCs published by the resource broker. We define a binary decision variable $x_{(i,j)}$ that indicates whether TD_i is deployed on resources provider j.

Knowing that the objective of the resource broker is to minimize the deployment cost and maximize the reputation of the user RP, we can formulate the problem as follows:

$$Minimize \sum_{i=1}^{K} \sum_{j=1}^{L} [\alpha(C_{(i,j)} x_{(i,j)} / C_{max}) - (1 - \alpha)(Rep_{(i,j)} x_{(i,j)} / Rep_{max})] \tag{3.3}$$

Where the constraints are:

$$\sum_{j=1}^{L} x_{(i,j)} = 1 \tag{3.4}$$

$$x_{(i,j)} \in \{0, 1\} \tag{3.5}$$

$$\alpha \in \{0, 1\} \tag{3.6}$$

Equation 3.3 reflects the objective function that aims to minimize the cost and maximize the reputation. A weight (α) is used in order to balance between the two objectives. A higher value of α means that the targeted solution prioritizes the reduction of the cost, while a low value of α indicates that the solution is giving more priority for maximizing the reputation. Rep_{max} and C_{max} are the maximum values of $Rep(i, j)$ and Ci_j. They are used to normalize the reputation and cost values to be used in the same objective function.
Constraint 3.4 ensures that a sub-slice resource of TD_j is deployed only once. Constraints 3.5 and 3.6 guarantee that $X(i, j)$ variables are binary and α is between 0 and 1, respectively. By solving the ILP, the resource broker can select the optimal RP in which a sub-slice should be deployed. We will study the solutions obtained by solving the ILP with Gurobi solver.

3.3.2 A Trust architecture for SLA management

3.3.2.1 SLA establishment

Once the resources are negotiated and the SP has selected RPs in which each sub-slice will be deployed, SLAs have to be established. On one hand, between the SP and V and on the other hand, between SP and each RP.
Generally speaking, SLAs are contracts between consumers and a service provider. An SLA specifies what customers can expect from a service provider. In 5G, SLA should reflect the QoS as related to the pre-defined type of slices, i.e. mMTC, eMBB, and uRLLC. The established SLA needs to be managed and monitored in order to ensure that the service is well functioning, and hence build the reputation of RPs.

The SLA management can be divided into eight phases, as shown in Fig. 3.2. The first SLA establishment is between V and SP, while the second establishment is between SP and RPs. In the first step, V identifies which SP can deliver the necessary resources that meet a network slice's needs.

The second step is the negotiation between SP and V. The two actors should agree on the terms, target performance level for the end-to-end network slice, and the trust monitoring system. When one party does not agree, then the process has

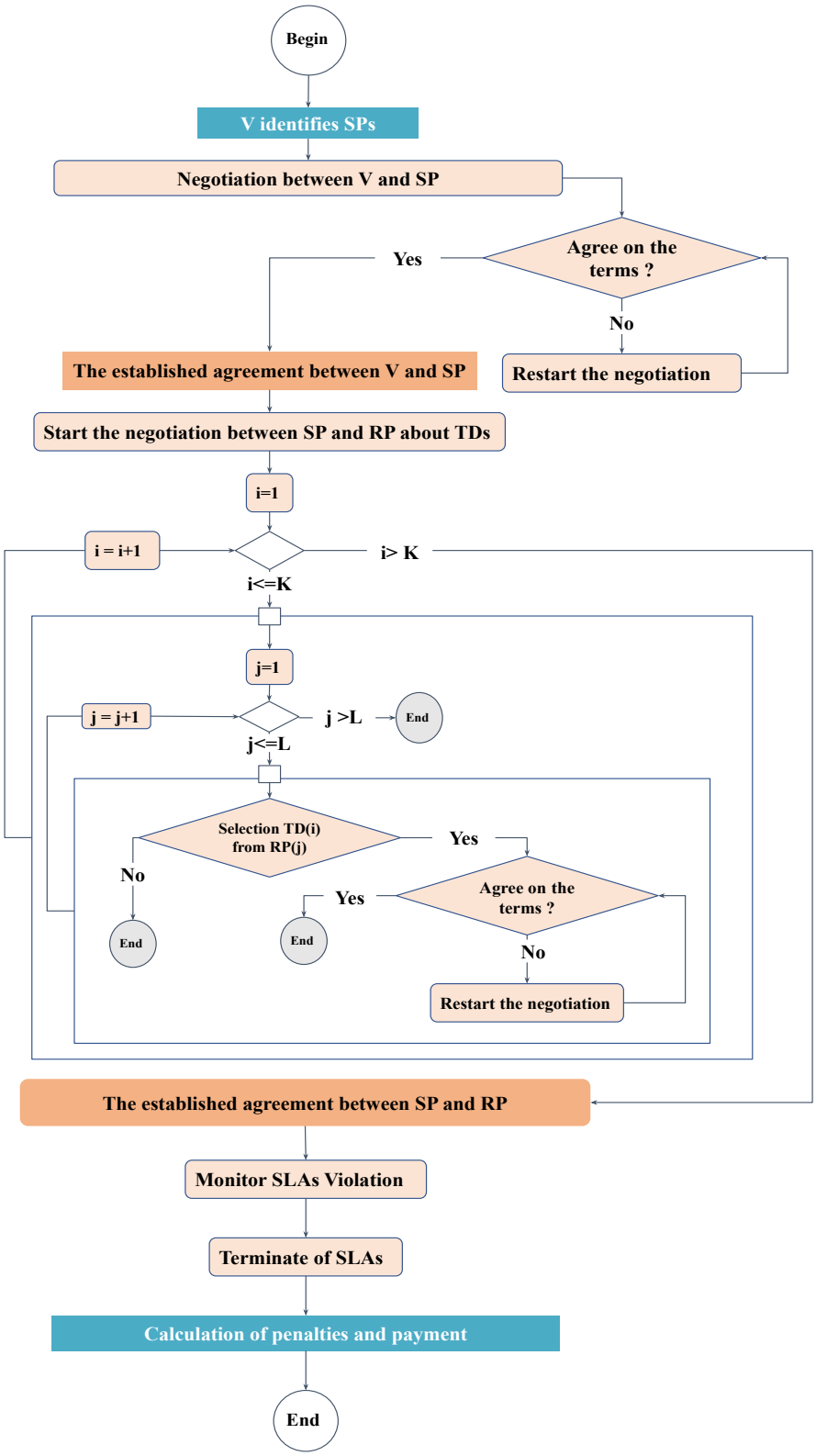


Fig. 3.2 The life cycle of SLAs management.

to start again until a consensus is reached between the different actors. The third step is the established agreement between V and SP. It involves the structure and

definition of the SLA template that is going to be placed. In this phase, the two parties (V, SP) sign the contract and agree about the obligations and penalties. The fourth step is the negotiation between SP and RPs. SP should select RPs to serve sub-slices, hence to deploy the end-to-end slice. SP should select K TDs from L RPs. They should agree on the terms, target performance level for the sub-slice, and the trust monitoring system.

The fifth step is the established agreement between SP and RP. It involves the structure and definition of the SLA template that is going to be placed. In this phase, the parties (SP, RP) sign the contract and agree about the obligations and penalties. It is worth noting that steps four and five are repeated for each sub-slice to deploy.

The sixth step is the SLAs violation monitoring, which consists of monitoring the slice and sub-slice performance metrics. This step is important to detect if SP or RPs are meeting the defined performance level signed in SLAs. When the service is not as expected, V must be compensated by the SP, and the latter must be compensated by RP that did not respect the SLAs. All parties, i.e., V, RPs, and SP, should actively monitor the metric of the slice and the sub-slices using a trust monitoring system, which should provide reliable measurements. The last step is the SLA termination. Based on the terms defined in the definition of SLA, the termination of an SLA happens when the SLA validity expires or when an SLA violation is detected. Also, they end when V wants to finish this service.
On the other hand, a calculation of the compensation in the case of SLA violation should be calculated automatically. Finally, in the phase of the penalties for SLA violation, the payment of the compensation, calculated based on the penalties defined in the SLA will be transferred to the account of V and the SP automatically.

3.3.2.2 SLA Structure

Similarly to any contract, SLA is a structured form of elements set (mandatory and optional), namely the period of validity, the parties who signed the SLA, the services type, guarantees in terms of objectives, penalties, suspension, or termination of the contract and compensation.

The period of validity: As shown in Fig. 3.3, the period of validity of a SLA is defined by the date of beginning and the end of the network slice lifecycle. The latter can be modified in accordance with the conditions of termination.

The parties: This section describes the parties involved in the contract, i.e., the three signatory parties: V, SP, and RPs. It includes the signatory parties as

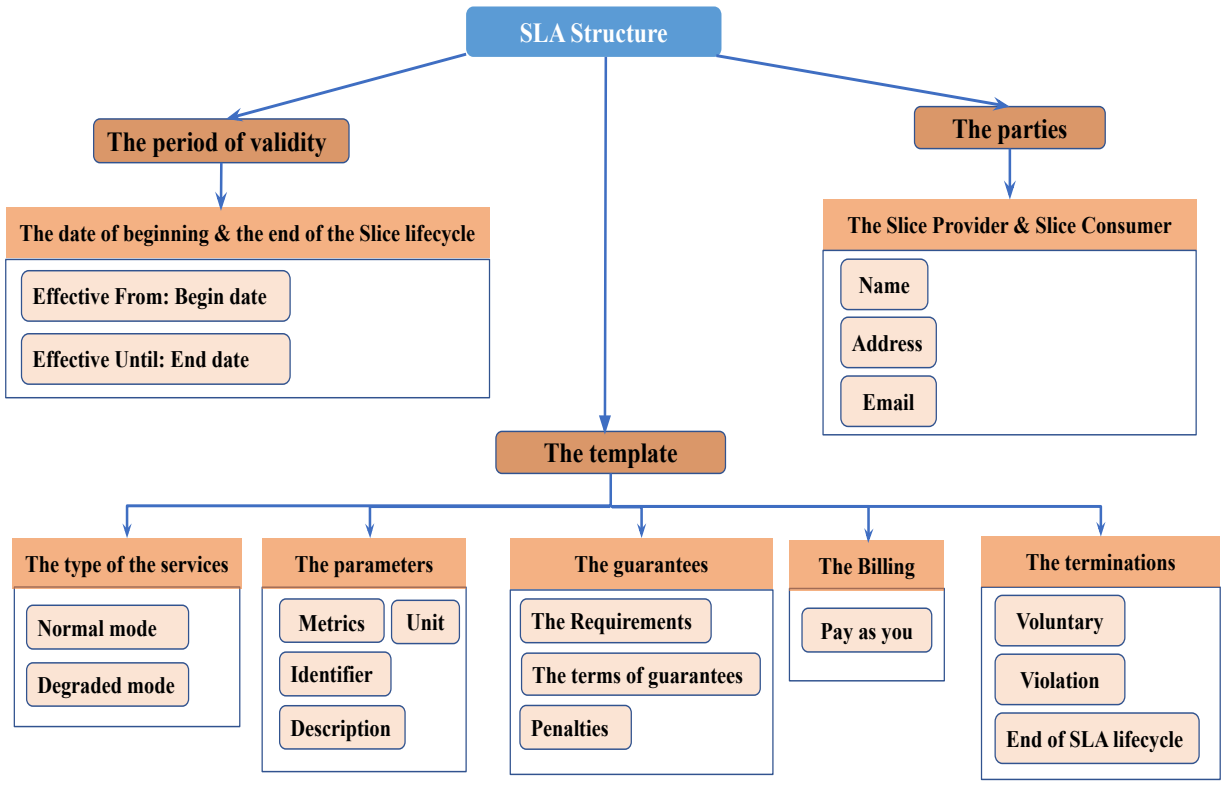


Fig. 3.3 Service Level Agreement Structure.

well as the trusted fourth party, i.e., the trust monitoring system, which is used to prove that the wanted service worked as expected, or was unavailable or less than the performance level signed in SLA. This part presents the properties of the parties, such as, the name, and the contact elements. In the studied case of Network Slicing in 5G, the first SLA is established between V and SPs. Later on, other SLAs are established between the SP and the selected RPs.

The template: It presents an essential part of the SLA. The template defines the parameters to ensure that the QoS offered are realistic and achievable. This part includes five elements: the services type, the parameters, the guarantees, the billing, and the termination condition.

The type of services: In 5G, there are different types of SLA signed. In this new generation, we have different types of slices with distinctive prioritization. Actually, 5G presents three network slice types, based on the typical characteristics required for use cases and Vs. The first type is eMBB slice which is basically an extension of the 4G mobile broadband service. The second is uRLLC which provides low-latency and reliable communication. The third type is mMTC which supports massive IoT devices with narrow bandwidth requirements.
Also, Slices are independent from each other and provide unique services without

interference with other slices. The core network slicing technology will facilitate the diversity of the new services offered. The variation of each slice priority needs dissimilar SLAs with different services. Moreover, the service provided can have one or more operating modes. For example, SP can offer a specific service with two modes: "*normal mode*" and "*degraded mode*". The advantage of degrading functionality is that SP can better plan its resources in short-term load peaks, without bothering V. Also, the service price will be adjusted according to the modes and is used to be attractive to V and profitable for SP.

The parameters: The parameters define the variables used in other contract sections by presenting particular elements such as the metric and the monitoring types. The metric is defined according to an identifier, a description (e.g., latency, throughput, reliability), and a corresponding unit (e.g., ms, bps, %). The monitoring specifies the nature of the evaluated metric such as average, maximum, and minimum. It also presents the evaluation window and the frequency of the values collection (period of the monitoring). As an example, for uRLLC slice, it is important to control the maximum value of latency.

The guarantees: This part presents three elements: requirements, terms, and penalties.

- *The Requirements:*

It presents the essential specifications for the slice's good functionality; a specification may be mandatory or optional. Moreover, it describes one or more preliminary technical elements to be completed, such as the version of a sensor used to connect to the mMTC slice.

- *The terms of guarantees:*

The terms of guarantees detail the Service Level Objectives (SLO) of the contract. A guarantee is a composition of terms (e.g., reliability, latency, bandwidth, throughput) and objectives via operators such as "**Or**" and "**And**". As an example, for the good performance of an uRLLC slice, V needs 4 objectives: the bandwidth of the slice should be between 20 Mbps (as a minimum threshold) and 50 Mbps (as a maximum threshold) "**and**" also the slice reliability should be equal to 99.9999 %, "**and**", End-to-End latency should be between 20 ms and 100 ms, from RPs until the end-user "**and**" finally, the throughput of eMBB slice be required to 100Mbps and 1Gbps.

- *Penalties:*

In case of a degradation of the end-to-end network slice performance (i.e., SLA violation), SP must pay compensation to V. In turn, the RPs involved in the SLA violation due to sub-slice resource degradation must compensate SP. The amount

of reimbursement of each term is agreed upon among the parties and described in the penalties section of the SLA. As an example, when the latency of the end to end slice exceeds the threshold, it means that one or several RPs are not respecting the performance signed in the SLA, which leads that the end-to-end SLA is not respected. Accordingly, SP firstly pays compensation to V. Then, RPs responsible for this bad performance of latency will pay compensation to SP.

The Billing: In general, billing follows two types: "*a pay package*" or "*pay as you go*". In our case, we use the billing "*pay as you go*" which means that the service's price depends on the operating modes, performance and penalties.

The terminations: A contract can be terminated automatically at the end date of SLA. The type of termination in this chapter could be as follows:

- *Voluntary*: when the V wants to finish this service.
- *Violation*: when the monitoring system detects a high level of violation.
- *End of SLA lifecycle*: when the validity of SLA ends.

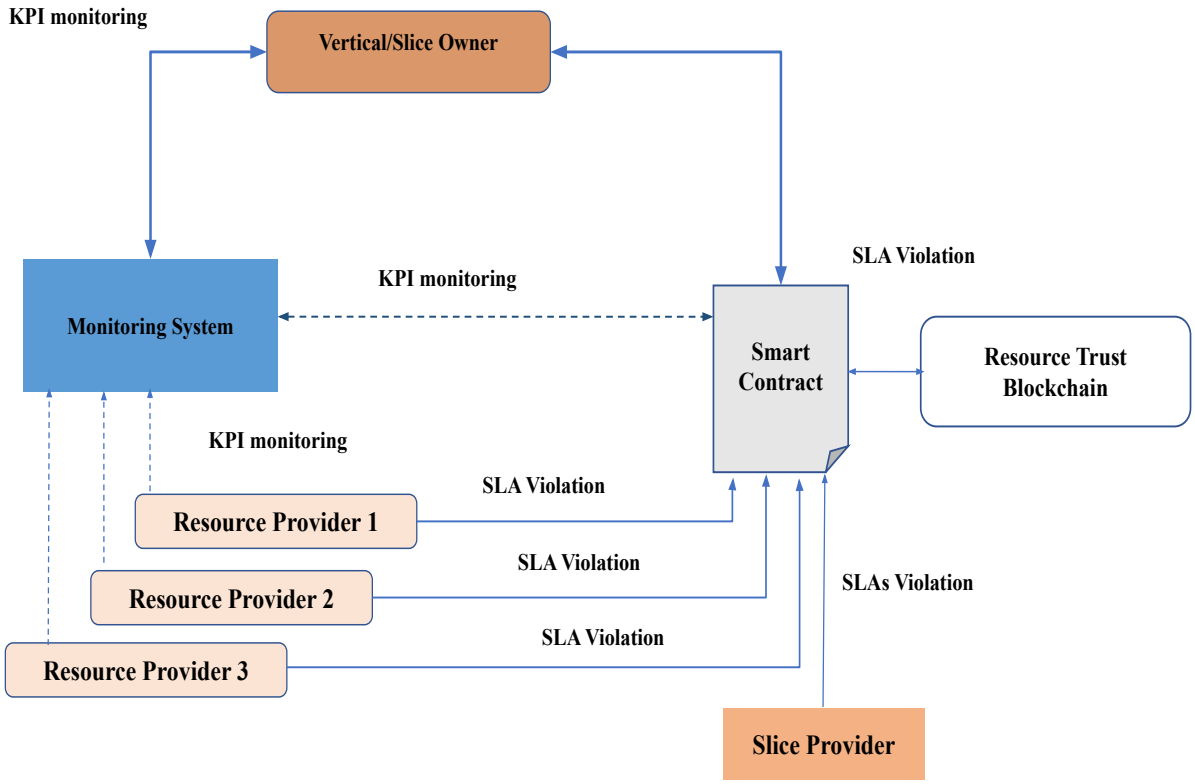


Fig. 3.4 Trust architecture for end-to-end network slice creation.

3.3.2.3 SLA management

In this section, we will introduce the trust management architecture to manage the SLAs among V, SP, and RPs. The proposed architecture is illustrated in Fig. 3.4. It complements Fig. 3.1 with two new entities in respect to SLA management, namely the monitoring system and the Smart Contract. The monitoring system is a third-tier trusted entity that monitors the KPI as specified in the SLA established between V and SP and also between SP and RPs. The monitoring information will be used to check if an SLA is violated. The Smart Contract will contain all the signed SLAs and related information such as validity period, target performance level, price, compensation value, and relevant information. It stores the addresses of SP, V, RPs, monitoring system, and Resource Provider trust. These addresses are used to check the account balance, transfer funds, report SLA violations to the Resource Provider Trust entity, and allow only authorized addresses to interact with the Smart Contract.

Smart Contract: In the proposed architecture, the Smart Contract is managed by the SP. According to the monitoring system's information, it (Smart Contract) checks if one of the involved entities is violating the SLA. Moreover, it automatically gives compensation to V if the service is not satisfactory, and finds which RP has failed to support the performance defined in SLA. The concerned RPs will be charged automatically to pay penalties to the service provider in this case. It is worth noting that the compensation and penalties to pay are committed by all parties when the SLA is signed. To this aim, the Smart Contract includes functions that calculate the number of detected violations in an interval. Also, it is used to verify if the compensation interval has ended. At the end of the SLA life cycle, the compensation is paid based on the number of violations. Finally, a compensation function is called to transfer the compensation value for both V address and the SP address.

In our case, Smart Contract uses algorithm. 1 to allow the dynamic compensation for V and SP. Here, the Smart Contract is used for an automatic subscription payment without the need of a Third Party Trust (TTP). First, the Smart Contract contains SLA information about the validity period, threshold of terms such as latency and throughput, initial SLA price, and compensation value per term. The Smart Contract also contains addresses of SPs, V, and RPs, as well as the monitoring system, in order to send the price and the compensation to the right parties. Second, the Smart Contract uses two special functions; a first function to check the account balance and to calculate the compensation, and a second one to transfer funds to the address.

The first function works as follows: when the measured value sent by the monitoring system is above the target performance level defined in the SLA, and the SLA life-cycle is not over yet, the number of violations will increase by one. These functions rely on time and use the block timestamp as a reference for the current time. As the Smart Contract is not able to automatically execute the function by itself, the monitoring solution must periodically call the Smart Contract to verify if the SLA is still valid or not. If the current block timestamp is above the SLA end time, the Smart Contract verifies if there is compensation to be paid. If not, it transfers the remaining Smart Contract balance to SP and RPs. Else, when the terms exceed the threshold during the life-cycle of SLA, a violation compensation value will be calculated. Then, the final price will be sent to both V and SP addresses stored by the Smart Contract, using the second function.

$$SumViolation(metric_k) = \sum SLAviolated(metric_k) \tag{3.7}$$

$$Comp(metric_k) = CompensationPerUnit(metric_k) * SumViolation(metric_k) \tag{3.8}$$

$$Compensation = \sum_{n=1}^{Number\_of\_metrics} Comp(metric_k) \tag{3.9}$$

Inside the Smart Contract, the dynamic calculation of the compensation value assumes that the monitoring solution performs measurements every second. During the SLA life-cycle, if the end-to-end slice's measured performance is different from the agreed performance level (threshold of latency or throughput in SLA signed between V and SP), then one or more RPs are not delivering the defined performance level. Thus, it is crucial to check which RP is responsible for this issue. Let's suppose that the slice is deployed using three TDs, where each TD is provided by one RP (RP_1, RP_2, RP_3). In addition, our Smart Contract contains the SLA information, such as:

- ***"Initial_SLA_price_V"*** that is the initial price supposed to be paid to SP by V as defined in the SLA,
- ***"Initial_SLA_price_*RP_1*"*** that is the initial price supposed to be paid RP_1 by SP as defined in the SLA,
- ***"T_Latency/Throughput_V"*** that is the threshold of metrics (KPIs) (Latency/
  Throughput) defined in SLA signed between V and SP.

For the sake of clarity, the proposed algorithm is not repeated for the two considered metrics. Hence, we will consider the following notation for the metrics (KPIs): "Latency/Throughput". For the eMBB slice, the metric to control is

the "Throughput" while for the uRLLC slice, the latency is to control. Therefore, "Latency/Throughput" corresponds to throughput and "Latency", respectively. If we aim to control more than one metric (latency and throughput) per algorithm, then we have to add another function similar to *"CalculateCompensation"* with different inputs.
It should be noted that:

- ***"CompensationPerUnitV"*** is the compensation value to be paid by SP to V in the case of one violation,
- ***"CompensationPerUnit*RP_1*"***, *"CompensationPerUnit****RP***$_2$*"*, and *"CompensationPerUnit****RP***$_3$*"* correspond to the value of compensation which will be paid to SP in case of one SLA violation by RP_1, RP_2 and RP_3, respectively.
- ***"addressVertical, addressSP, address*RP_1***, address*RP_2***, address*RP_3***"*** are the corresponding addresses of the actors' accounts.

We use the function *"CalculateCompensation"* to verify the performance of the latency or the throughput in the RP_1, RP_2, and RP_3 (see Algorithm. 1). When the latency level or throughput level is different from the threshold defined in SLAs, then the number of violation will increase by one, and we update the total number as in the equation 3.7. After that, we have to calculate the compensation value (*"Compensation"*), which is derived using equations 3.8 and 3.9. The compensation is then sent to the right address using the "*SendPrice*" function. If metrics' values (latency or throughput) did not reach a threshold of SLA (e.g., 30 ms for latency), meaning no SLA violation, then the V will pay the entire price defined in the SLA to the SP.
However, if the metrics' values exceed the threshold, then V will pay only the difference between the initial price and the paid compensation. Similarly, the SP will send only the difference between the defined price and the paid compensation to RPs. At the end, V receives a portion of the SLA price corresponding to the compensation from the SP, and the latter receives compensation from RPs.
On the other hand, the Smart Contract will write the SLA violation in the Resource Trust Blockchain, indicating which RP has violated the SLA for which TD resource. The information will be used by Resource Provider Trust to update the reputation of the RPs, as described in the previous section.

3.4 Performance Evaluation

In this section, we use computer simulation to evaluate the performances of the trust architecture framework in terms of (1) end-to-end network slice negotiation

Algorithm 1 Algorithm of the Smart Contract

Input: V_L/T_RP_1, V_L/T_RP_2, V_L/T_RP_3, V_L/T_V
▷ List of value send by the MS
Data: $CompensationPerUnitV, CompensationPerUnitRP_1, CompensationPerUnitRP_2, CompensationPerUnitRP_3$
$Initial\_SLA\_price\_V, Initial\_SLA\_price\_RP_1, Initial\_SLA\_price\_RP_2, Initial\_SLA\_price\_RP_3$, $T_{Latency/Throughput\_V}, T_{Latency/Throughput\_RP_1}, T_{Latency/Throughput\_RP_2}, T_{Latency/Throughput\_RP_3}$,
$addressV, addressSP, addressRP_1, addressRP_2, addressRP_3$
▷ Initialization
Output: $SLA\_Price\_V, SLA\_Price\_RP_1, SLA\_Price\_RP_2, SLA\_Price\_RP_3$
▷ Final SLA price will send from SP to V and Final SLA Price send from RP to SP
function CalculateCompensation($V\_L/T\_RP_1, V\_L/T\_RP_2, V\_L/T\_RP_3, V\_L/T\_V$)
  while EndofSLA = false **do**
    if $v(t, metric_{Latency/Throughput\_V}) \geq / \leq T_{Latency/Throughput\_V}$ **then**
      Number_SLA_violation_V =+ 1
      if $v(t, metric_{Latency/Throughput\_RP_1}) \geq / \leq T_{Latency/Throughput\_RP_1}$ **then**
        Number_SLA_violation_RP_1 =+ 1
      end if
      if $v(t, metric_{Latency/Throughput\_RP_2}) \geq / \leq T_{Latency/Throughput\_RP_2}$ **then**
        Number_SLA_violation_RP_2 =+ 1
      end if
      if $v(t, metric_{Latency/Throughput\_RP_3}) \geq / \leq T_{Latency/Throughput\_RP_3}$ **then**
        Number_SLA_violation_RP_3 =+ 1
      end if
    end if
    SLA_Price_V = Initial_SLA_price_V - (Number_SLA_violation_V * CompensationPerUnitV);
    SLA_Price_V_RP_1 = Initial_SLA_price_RP_1 - (Number_SLA_violation_RP_1 * CompensationPerUnitRP_1);
    SLA_Price_RP_2 = Initial_SLA_price_RP_2 - (Number_SLA_violation_RP_2 * CompensationPerUnitRP_2);
    SLA_Price_RP_3= Initial_SLA_price_RP_3 - (Number_SLA_violation_RP_3 * CompensationPerUnitRP_3);
  end while
end function

function SendPrice($SLA\_Price\_V, SLA\_Price\_RP_1, SLA\_Price\_RP_2, SLA\_Price\_RP_3$)
  if EndofSLA = True **then**
    send (SLA_Price_V) to ($addressSP$) ;
    send ($SLA\_Price\_RP_1$) to ($addressRP_1$) ;
    send ($SLA\_Price\_RP_2$) to ($addressRP_2$) ;
    send ($SLA\_Price\_RP_3$) to ($addressRP_3$) ;
  end if
end function

and deployment; (2) SLA management.

3.4.1 End-to-end network slice resource selection

For the simulations, we consider that four RPs respond to the SP's SSC. We assume that each RP provides an offer to run sub-slices for three TDs. We have changed the value of α (weight) for each run to observe the impact on the trade-off (between cost and reputation) when selecting the RPs to host sub-slices.

We use the Gurobi Jupyter Notebook to solve the ILP that allows selecting the RPs for each TD. Gurobi is useful for a wide variety of industries including manufacturing, financial services, energy and telecommunications as well as marketing campaign optimization and supply network design. As explained in the previous section, our objective function aims to find the resources based on their cost and reputation.

Table 3.1 summarizes the simulation results. First, we randomly generate cost and reputation for each couple (RP_i,TD_j). For each couple, the 2nd column indicates the reputation of RP_i when serving resources of TD_j, while the 3rd column corresponds to the cost of RP_i to run a sub-slice in TD_j. For each value of α (from column 4th to 9th), 1 indicates that RP_i has been selected to host a sub-slice of type TD_j; 0 otherwise.

To illustrate better the results of Table 3.1, we draw three Figures: Fig. 3.5, Fig. 3.6 and Fig. 3.7. Each figure shows, per TD, the reputation and cost of the fourth RP and the selected RP for each value of α.

For TD_1 (Fig. 3.5), we remark that RP_2 is selected for low values of α (between 0 and 0.6), while for high values, RP_4 is selected. We argue this by the fact that RP_2 has the highest reputation and the second lowest cost. Hence, it is selected when the reputation is privileged over cost. Nevertheless, when α is higher than 0.6, i.e., the cost is prioritized, RP_4 offering the lowest cost is selected. For TD_2 (Fig. 3.6), RP_1 is always selected as it has the highest reputation and proposes the lowest cost.

Regarding TD_3(Fig. 3.7), three different RP are selected. When α is low (between 0 and 0.2), i.e., reputation is privileged, RP_1 is selected as it has the highest reputation to serve TD_3 sub-slices. In contrast, when α is high (equal to 1), i.e., a lower-cost solution is privileged, RP_2 is selected. The latter offers the lowest price. Finally, when a trade-off is sought (α between 0.4 and 0.8), RP_4 is selected as it provides the better trade-off between reputation and cost.

Clearly, from these results, we deduce that the proposed selection algorithm to be run by the resource broker is able to find the best trade-off between maximizing the reputation and reducing the cost by well fixing α.

Table 3.1 Results of simulation for the algorithm of resource selection while varying α.

(RP, TD)	Reputation	Cost	$\alpha = 0$	$\alpha = 0.2$	$\alpha = 0.4$	$\alpha = 0.6$	$\alpha = 0.8$	$\alpha = 1.0$
(1, 1)	123.0292121	1303.276056	0	0	0	0	0	0
(1, 2)	415.0614412	848.8607873	1	1	1	1	1	1
(1, 3)	389.0051414	942.9657908	1	1	0	0	0	0
(2, 1)	174.4518733	1153.495538	1	1	1	1	0	0
(2, 2)	302.9867984	1837.715308	0	0	0	0	0	0
(2, 3)	266.1766331	542.1692793	0	0	0	0	0	1
(3, 1)	113.8639498	1224.255528	0	0	0	0	0	0
(3, 2)	361.1993909	1047.50991	0	0	0	0	0	0
(3, 3)	330.9289954	830.3358363	0	0	0	0	0	0
(4, 1)	114.0687306	1050.376848	0	0	0	0	1	1
(4, 2)	379.1780584	1241.622399	0	0	0	0	0	0
(4, 3)	350.4639782	548.1868774	0	0	1	1	1	0

3.4.2 Performance evolution of the Smart Contract

To evaluate the proposed SLA trust management framework, we choose two different metrics: the latency and throughput. In fact, these quantitative metrics are defined in SLAs, and it is possible to be controlled by the Smart Contract. We simulate a network slice deployed on three different TD; a different RP provides each one. V application is represented by a client and a server. We focused on measuring both the latency and bandwidth as key performances, which are periodically sent to Smart Contract by the monitoring system.

The deployment of the Smart Contract was performed using Ethereum [48], a Blockchain-based distributed computing platform. This program is executed by nodes on the Blockchain in the Ethereum Virtual Machine. Also, we use Ganache [49], a local Blockchain designed for developing and testing. It stimulates a real Ethereum network, including the availability of accounts number funded with 100 Ether by default. It presents an interface that can be accessed on a port of the localhost in the same way one would connect to a real Ethereum node. Also, we use truffle [50], which is a development environment for the compilation and deployment of Smart Contracts. Solidity [51] is the programming language used to write the contract code, and truffle looks for ".sol" files to compile and migrate to the Blockchain.

Moreover, we use Postman, which is an API testing tool. It can send the value of the metrics (throughput or latency) to the Smart Contract. The evaluation scenario stared from the monitoring system performing periodic requests. It sends the current values of the two metrics to the Smart Contract, every defined period. During a 100 seconds interval, the evolution of the latency performed in V

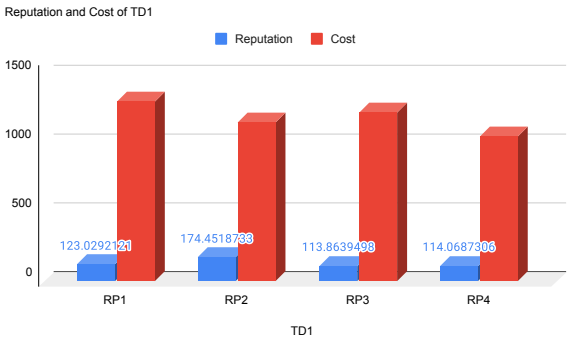
Table 3.2 Parameters of the Smart Contract.

Parameters	**Value**
Price SLA (V - SP)	100 ETH
Price1 SLA (SP - RP_1)	20 ETH
Price2 SLA (SP - RP_2)	20 ETH
Price3 SLA (SP - RP_3)	20 ETH
Compensation	1 ETH
Compensation1	0.2 ETH
Compensation2	0.2 ETH
Compensation3	0.2 ETH
Threshold of latency defined in SLA_SP_V	30 ms
Threshold of latency defined in SLA_RP_1	10 ms
Threshold of latency defined in SLA_RP_2	10 ms
Threshold of latency defined in SLA_RP_3	10 ms
Threshold of throughput defined in SLA_SP_V	100 Mbps
Threshold of throughput defined in SLA_RP_1	100 Mbps
Threshold of throughput defined in SLA_RP_2	100 Mbps
Threshold of throughput defined in SLA_RP_3	100 Mbps
Validity	100 s

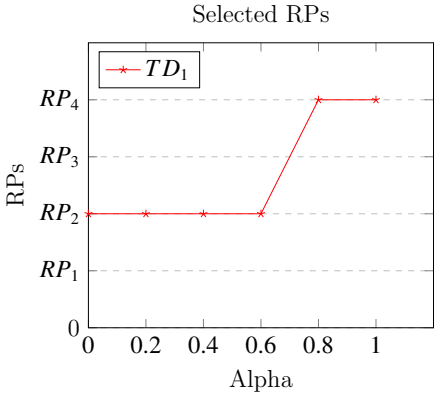
is depicted in Fig. 3.8. The test works as follows, to evaluate the proposed solution. Firstly, the Smart Contract is deployed, and a new SLA is created following the parameters presented in Table 2. Secondly, the measurements of latency or throughput are considered the output of the trusted monitoring agent that performs periodic calls to the Smart Contract informing about SLA violations. If the monitored latency or throughput is above the threshold defined in the SLA; then the Smart Contract increases the number of SLA violations for the period until the end of the lifecycle of SLA. We have considered two use cases: when V requests uRLLC slice or requests eMBB slice.

3.4.2.1 uRLLC Slice

As specified by many uRLLC services in 5G, the needed latency is required to be between 5ms and 30ms. Based on these values, we fixed the violation threshold of latency for V to 30 ms, as depicted in Fig. 3.8d with a red line. In addition, we suppose that the violation threshold of the latency in the RPs 1, 2, and 3 should be 10 ms, as depicted in Fig. 3.8a, Fig. 3.8b, Fig. 3.8c with a red line. We argue this by the fact that we measure the end-to-end latency, which is composed of the

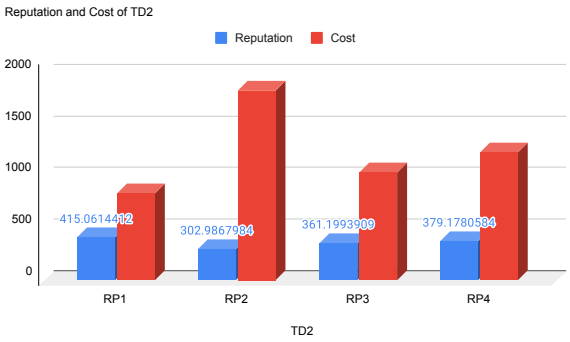


(a) Reputation and cost of TD_1.

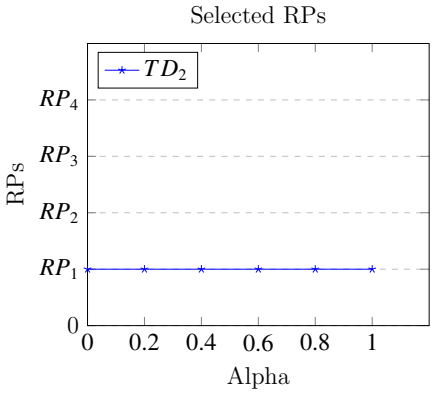


(b) The selected RPs.

Fig. 3.5 Resource provider selection for TD_1.

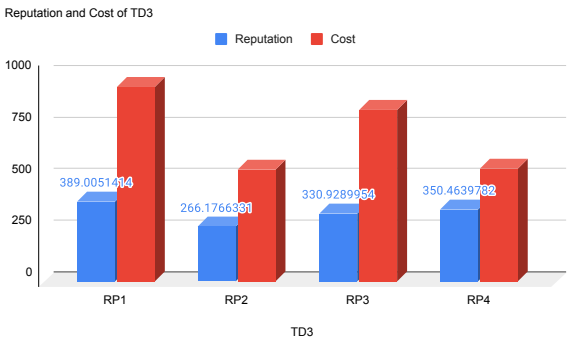


(a) Reputation and cost of TD_2.

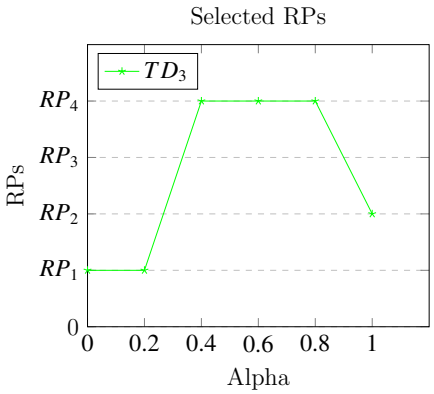


(b) The selected RPs.

Fig. 3.6 Resource provider selection for TD_2.

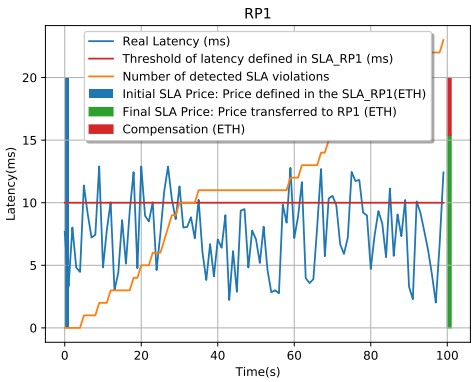


(a) Reputation and cost of TD_3.

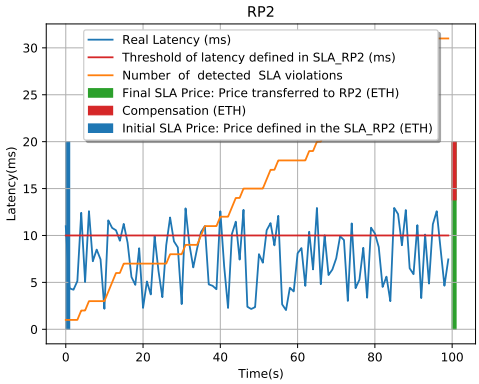


(b) The selected RPs.

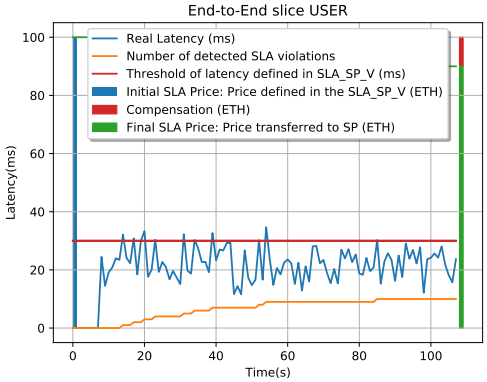
Fig. 3.7 Resource provider selection for TD_3.



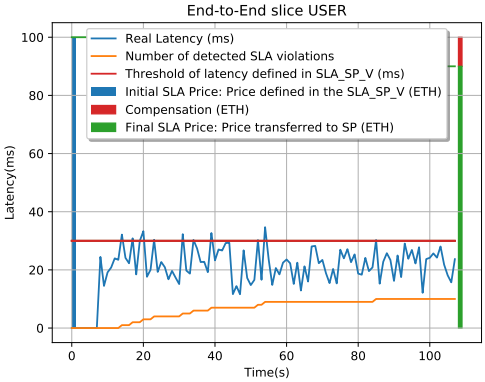
(a) Evolution of latency and balance of RP_1



(b) Evolution of latency and balance of RP_2



(c) Evolution of latency and balance of RP_3



(d) Evolution of latency and balance of V

Fig. 3.8 Evolution of latency and balance of RPs and V.

latency experienced in each TD. During the 100 seconds interval, the evolution of the latency performed in the V and RPs is depicted in Fig. 3.8; represented by a blue line. After that, the latency values in the end-to-end slice, RP_1, RP_2 and RP_3, are iterated to verify each one against the target performance level defined in the SLA (i.e., 30 ms for V (Fig. 3.8d) and 10 ms for RPs 1, 2 and 3 (Fig. 3.8a, Fig. 3.8b, Fig. 3.8c). If the value is above the threshold, then the number of detected SLA violations will increase by one, which is depicted using an orange line in Fig. 3.8. As a result, the initial price that is supposed to be sent to SP (Fig. 3.8) will be decreased.

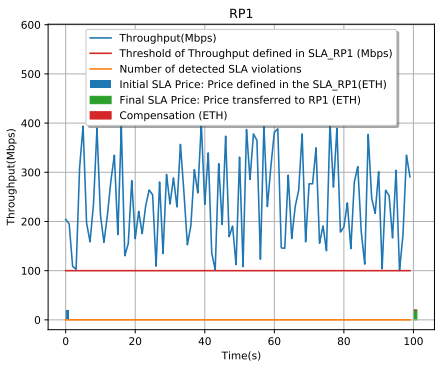
After the end of SLA, the billing model allows the dynamic compensation to V and SP and automatic payment, without need to TTP using the Smart Contract to realize these transfers. First, it is expected that V sends the price of the network slice before the start of the slice. The blue rectangle, in Fig.8, presents the initial price defined in SLA signed between V and SP. V deposits 100 ETH in the beginning. This fee is locked in the Smart Contract until the end of the SLA lifecycle. Once the SLA is finished, the funds are transferred to SP and RPs. The green rectangle presents the real price of SLA to be paid by V after the calculation process, and the red rectangle value depicts the compensation that will be paid to V. At the end of the SLA lifecycle, the compensation will be calculated based on the counter during the slice usage. Actually, the real price that will be paid at the end is the difference between the initial price and the compensation. In our case, the initial price is 100 ETH, and the compensation value per unit is 1 ETH. So, the final price to pay is 100 - 1 x 12 (number of violations) = 88 ETH. Consequently, the final price will be 88 ETH. Meanwhile, the monitoring system also sends the measured values of latency at RPs 1, 2 and 3, which are participating in the running of uRLLC slice. As shown in Fig. 3.8a, Fig. 3.8b, Fig. 3.8c, when the SLA is starting, RP_1 should receive 20 ETH at the end of the SLA. But when the performance of latency in RP_1, RP_2, or RP_3, exceeds the threshold (10 ms), their counters of the SLA violation will increase by one. At the end, RP_1 will not receive 20 ETH, but it will receive the difference between the initial price and the compensation. The latter is calculated in the Smart Contract. Similarly, the same process will happen to RP_2 and RP_3, as shown in Fig. 3.8 b and c, respectively.

3.4.2.2 eMBB Slice

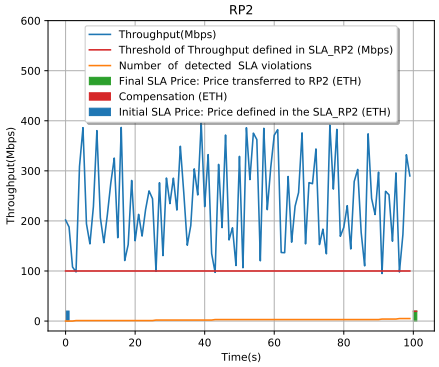
Another example we considered is the eMBB slice, which needs a high throughput between 100 Mbps and 1Gbps. Based on these values, the violation threshold was fixed to 100 Mbps, which is depicted using a red line in Fig. 3.9d. It is worth noting

that unlike latency, the end-to-end throughput is not composed of the throughput observed in each domain; each RP should support the same throughput. The Smart Contract will be used to ensure that the throughput observed at V and provided by RPs is higher than this threshold.

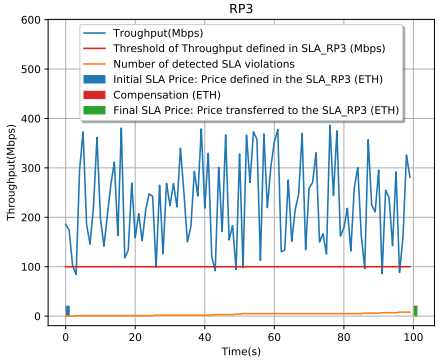
It can be seen in Fig. 3.9a, Fig. 3.9b, Fig. 3.9c that when the throughput is lower than the threshold, the counter of SLA violations will be increased by one, which is depicted using an orange line. Consequently, the initial price supposed to be sent to SP will be decreased, which is depicted using a green dashed line in the figures. The blue rectangle presents the initial price defined in the SLA signed between V and SP. At the end of the SLA lifecycle, the compensation will be calculated based on the counter during the slice use.



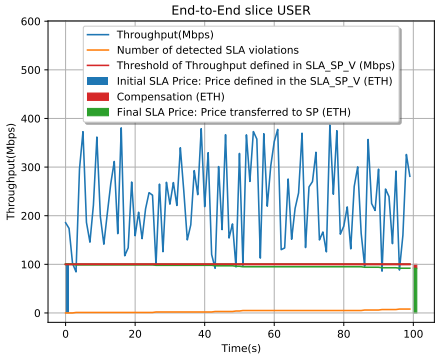
(a) Evolution of throughput and balance of RP_1



(b) Evolution of throughput and balance of RP_2



(c) Evolution of throughput and balance of RP_3



(d) Evolution of throughput and balance of V

Fig. 3.9 Evolution of throughput and balance of RPs and V.

Actually, the real price that will be paid at the end is the difference between the initial price and the compensation. In our case, the initial price is 100 ETH, and the compensation value per unit is 1 ETH. So, the final price to pay is 100 - 1 x 8 (number of violations) = 92 ETH. Therefore, the final price will be 92 ETH. On the other hand, the monitoring system also sends the value of throughput

provided by RPs 1, 2 and 3, participating in deploying the eMBB slice. When the SLA is starting, RPs 1, 2 and 3 should receive 20 ETH at the end of the SLA, as signed in SLAs. However, when throughput in RP_1, RP_2 or RP_3 are less than the threshold (100 mbps), their counters of the SLA violations will increase by one. At the end, RP_1, RP_2 and RP_3 will not receive 20 ETH, but they will receive the difference between the initial price and the compensation, which is calculated in the Smart Contract.

3.5 Conclusion

This chapter has introduced a trust management architecture that allows negotiating and deploying end-to-end network slices and managing automatically established SLA among involved actors. The proposed framework relies, on one hand, on Blockchain and Smart Contracts to ensure security and anonymous transactions, and on the other hand, on an ILP-based optimization model to select the most adequate resource provides.

Through simulations, we proved the ability of the proposed framework to select the optimal RP for each sub-slice. In addition, results showed that the SLA management process was successfully automated using a decentralized solution and removing a third tiers player's dependency to handle the billing process.

CHAPTER 4

Zero-touch security management for mMTC network slices: DDoS attack detection and mitigation

4.1 Introduction

In recent years, computing has evolved to support the growing market of IoT, which involves pieces of hardware, such as sensors, smart door locks, smartwatches, etc., having the ability to connect to the network, transmit and receive data. A recent forecast estimates the number of IoT devices that should be connected to the network will exceed 25.4 billion in 2030 [52].

In terms of connectivity to the network, several technologies are candidates to connect these devices. We can mention LORA, SigFox, IEEE 802.14.5, and 5G. The latter is seen as an excellent alternative as it allows continuous connectivity of the IoT devices, supporting even mobile devices (i.e., embedded in cars or drones). 5G is the latest generation of mobile networks promising the support of several new emerging services, including those relying on IoT.

5G envisions the running of network services relying on IoT as a network slices of mMTC type. As its name indicates, this type of network slice is intended and optimized to connect a massive number of IoT devices to a service or an application. 5G supports multi-tenancy, as network slices run in parallel, and a high degree of isolation is ensured among them; a network slice component (control or data plane) cannot access other network slice components. In this chapter, we focus on mMTC network slices since they are used to deploy IoT applications and services using MTC devices.

Meanwhile, the growing market of IoT devices has increased cyber security threats and widened attack surfaces; hence securing modern networks is a challenging task. Indeed, IoT devices are often weak when it comes to security, considering: (1) their usual low-cost and the lack of the necessary built-in security controls to defend against threats; (2) their constrained environment and limited computational capacity. Besides, they are a high-value asset to attackers. Indeed, finding a vulnerability of a type of device allows attackers to infect more devices of that type and, hence, conduct attacks from the infected equipment. For example, in 2016, a Malware called Mirai infected between 800,000 and 2.5 million devices through default passwords and used them to conduct DDoS attacks against some public web applications [52].

5G was designed with built-in security controls to address many of the threats found in 4G/3G/2G networks, such as enforcing mutual authentication to prevent fake base-station attacks, encrypting subscriber identities, and enforcing more secure cipher suites. However, more functionality always comes with new risks and attack vectors when opening the network to IoT devices, particularly. Some of these risks can affect the performances of 5G network functions (mainly the service availability). By embracing network slicing, 5G networks can mitigate inter-slice

attacks as isolation is one of the key features. Indeed, the isolation guaranteed by network slicing offers performance guarantees to the applications and ensures isolation, such that attacks (e.g., leakage, breach, DDoS remain contained and do not propagate to the network. However, an in-slice attack may correspond to a subset of the UEs attached to a specific network slice issuing malicious traffic towards the infrastructure services. A typical example is compromised MTC devices (i.e., IoT devices) generating a massive number of network attachment requests. These attacks need to be quickly identified and mitigated to avoid system failure. It is worth noting that 3GPP stated clearly in [53] that 5G networks should be protected from DDoS attacks, where mechanisms detecting and mitigating this type of attack are needed.

In this chapter, we propose ZSM solution that addresses the challenges of in-slice DDoS attack detection and mitigation, considering the case of mMTC network slices. Generally, this type of attack targets the 5G CN elements shared among the network slices. The proposed ZSM solution relies on a closed-control loop composed of a triplet (MS, AE, and DE), that interacts with the 5G CN in order to detect attacks and automatically react by mitigating the attacks. The critical challenge addressed in this chapter is how to detect a DDoS attack initiated by a compromised set of MTC devices inside a network slice. Indeed, there is no available traces or dataset that reproduce abnormal traffic in 5G, unlike other types of networks where many datasets are available. Besides, it is very challenging to detect during an event if there is an attack and what are the involved devices in the attack. To overcome these limitations, we followed 3GPP traffic models [54, 55] for MTC to identify normal traffic and train an ML model to recognize the normal traffic, and hence abnormal traffic. Our solution waits until the end of an event to analyze the traffic (normal or abnormal), and deduce if an attack happened and what are the involved devices. Then, all the devices will be detached and banned from future access to the network. We believe our solution is pertinent to mitigating a DDoS attack as the latter loses its intensity, if the number of involved devices is reduced by detaching and blocking them.

The main chapter's contributions are:

- A novel closed-control loop featuring AI/ML to achieve ZSM objective in 5G and beyond networks,
- A novel approach that detects MTC attach events over a time duration,
- A novel ML algorithm that leverages Gradient Boosting to learn and predict an upper bound interval for normal MTC traffic (following a $\beta(3,4)$ as per the recommendation of 3GPP),

- A novel DDoS detection algorithm that defines a detection rate of all MTC devices involved in an attack,
- A novel mitigation algorithm that detaches and blocks all MTC devices that have been suspected to be part of an attack,
- An implementation of the ZSM concept (i.e., closed-control loop) has been done on a 5G facility to prove the concept and evaluate the performances of the attack detection and mitigation algorithms.

The rest of this chapter is organized as follows. In section 4.2, we present a review of related works. We provide general background about the used mechanisms to detect and mitigate DDoS attacks. Section 4.3 details the contributions related to attach events and attack detection, including the devised ML method. Section 4.4 presents our attack mitigation algorithm. We introduce the proof of concept implementation and extensive performance results of the proposed detection algorithm, comparing it with a statistical-based solution. Finally, section 4.7 concludes the chapter.

4.2 Background and related work

In this section, we provide the necessary background and related work to understand the concepts presented in this chapter. First, we start by highlighting the 5G thread landscape. Second, we present how ML is used to detect different threats. Then, we end the section by focusing on DDoS attacks in 5G that rely on IoT devices.

4.2.1 5G threat Landscape

A 5G network comprises four major technological domains: the RAN, CN, transport network, and inter-connect network [56]. From a security perspective, some threats affect all of these planes, and others affect only specific ones. [57]. Threats in 5G can be classified according to the parts (or the technological domains) of the network they are impacting [57]:

- UE threats: Mobile botnets can launch DDoS attacks on multiple network layers, impacting the 5G infrastructure, web servers, and other UEs. The goal is to disturb and shut-down services.
- RAN threats: Rogue base station threats for conducting Man in the Middle (MitM) attacks, this class of attacks can compromise user information, break privacy and track users, and cause a Denial of Service.

- CN threats: These attacks relate to elements of the Core Network, including SDN and VNF components, as well as Network Slicing. These attacks can lead to Denial of Service, eavesdropping, interception, or hijacking [58].
- Network slicing threats: attacks that can break the isolation between network slices [57].
- SDN threats: Separation of control and user (data) plane that allows a malicious user to attack the link between the control and the user planes. For instance, a DDoS attack could be performed, or control could be gained over network devices (ex., Topology Poisoning attacks) [59].

4.2.2 Machine Learning for threat detection

ML has gained a broad interest in many applications and fields of study, particularly cybersecurity. With hardware and computing power becoming more accessible, ML methods can be used to analyze and classify bad actors from a huge set of available data. There are hundreds of ML algorithms and approaches, broadly categorized into supervised and unsupervised learning [60]. Supervised learning approaches are made in the context of classification, where input matches to an output, or regression, where input is mapped to continuous output. Unsupervised learning is mostly accomplished through clustering and has been applied to exploratory analysis and dimension reduction. Both clustering and regression methods can be applied in cybersecurity for analyzing malware in near real-time, thus eliminating the weaknesses of traditional detection methods [60]. ML techniques have been used to detect and mitigate DDoS attacks on networks effectively. Successful approaches have always used a variety of features to achieve great results, and have targeted DDoS attacks that are either very popular (that have public datasets available), or that are easy to reproduce (for which creating a dataset similar to real-world behavior is possible). Most of these approaches target common application protocols like Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). Table 4.1 summarizes some of the existing research in this area.

However, the difference in the used datasets complicates the comparison among the different solutions. Moreover, accuracy levels can be misleading since these solutions consider that a packet is a sample in the dataset instead of a malicious host. Therefore, with the flooding nature of the DDoS attacks, even a naive decision of considering that all the traffic is malicious would yield an accuracy higher than 80%. Finally, the lack of publicly-available data and datasets for attack detection on 5G networks makes it hard to understand, reproduce and mitigate 5G attacks.

Table 4.1 Comparison of some DDoS attack detection solutions.

Reference	The mitigated attacks	ML algorithms used
João Henrique Corrêa et al. 2019 [61]	TCP SYN flood	KNN and Decision Trees
Rohan DDoShi et al. 2018 [62]	TCP SYN flood, UDP flood, HTTP GET flood	KNN, Support Vector Machines (SVM), Decision Trees, Random Forests and Neural Networks
Faisal Hussain et al. 2020 [63]	Various attacks on application protocols	RESNET
Xiaoyong Yuan et al. 2017 [64]	Various attacks on application protocols	CNN and RNN
Maryam Ghanbari et al. 2018 [65]	Various attacks on application protocols	CNN after SVM pre-processing
Roberto Doriguzzi-Corin et al. 2020 [66]	Various attacks on application protocols	CNN

4.2.3 Detection of DDoS attacks in IoT and 5G networks

One of the main threats in 5G networks is DDoS. Hence, many works relying on ML have designed algorithms and tools to detect and mitigate this type of attacks. In [67], a ML algorithm is proposed to circumvent attacks through an automated solution in 5G networks. It has been integrated with the Deep Packet Inspection (DPI) function to detect unfamiliar attacks in the traffic of IoT devices. A solution named "Corero" has been introduced in [68]. It aims to help organizations to comprehend 5G attacks and specifically prevent DDoS attacks, using automatic and real-time detection algorithms. However, the presented approach needs to use a very high-cost infrastructure.

In [69], the authors present an IoT security framework, including Azure, AWS IoT, and SmartThings. The AWS framework supports mutual authentication of IoT devices, which can be done using certificates, groups, roles, and AWS Cognito identities. The used authorization is policy-based, with the rules mapped to each certificate, allowing the owners of IoT devices to define their rules. All IoT traffic is encrypted to secure communications. In [70], the authors propose using

security monitoring methods to detect and prevent massive IoT devices from being attacked or controlled. The proposed solution prevents IoT devices from being used maliciously to launch, for instance, DDoS attacks. The detection mechanism of the proposed solution is based on a randomization technique to defend against DDoS signaling attacks that emerged in the new 5G RRC three-state model.

The work in [71] proposes a modified Network Intrusion Detection System (NIDS) that defends against DDoS attacks on the mobile edge of multi-tenant 5G networks. The proposed approach solves the problem by combining traditional NIDSs with a classifier based on SVM to obtain the needed information on the device's traffic. Two filters are implemented: the first filter is placed at the mobile edge computing servers to prevent spoofing attacks close to the source, and the second filter is located in cloud servers to classify the complete network traffic.

Unfortunately, the few works mentioned in this section propose solutions that need new software or specific protocols, which are not standard and need high-cost implementation. Besides, they do not consider the context of network slicing, where most of the addressed challenges are solved thanks to network slice isolation. Therefore, the main remaining security threats and challenges come from compromised devices running inside the slice. In this case, shared network components are vulnerable to in-slice attacks since the devices are under the tenant's control and have the green light to access the network. To the best of the authors' knowledge, our work is the first one that addresses the challenge of in-slice attack detection and mitigation in 5G considering mMTC network slices.

4.3 Zero-touch security management for in-slice attack: Assumptions and System architecture

Our work aims to provide a solution featuring zero-touch security management of in-slice attack detection and mitigation considering mMTC slices in 5G. The proposed solution relies on machine learning to detect abnormal traffic of MTC devices that could cause DDoS on the control plane of the 5G core network (by flooding the AMF with signaling messages). Hence, it will be possible to mitigate the attack by making efficient decisions to prevent flooding of the AMF with traffic and causing DDoS or deteriorating performance for legitimate users. This type of attack can be more effective on mMTC than other 5G services, assuming the very high number of MTC devices supposed to support. In this chapter, we assume that a mMTC slice is composed of a shared sub-slice with other existing network slices, which runs the 5G CN (including the AMF) and gNB, and a specific sub-slice to run the application that collects data from the MTC devices.

It is well accepted that MTC devices generate two main types of traffic [72].

The first one is "Periodic", where the devices emit data periodically, which may correspond to the case of meteorological data. The second type is "Event-Driven". In this case, the devices emit data when a specific event occurs; for example, smoke (the signal of a possible fire) is detected. Detecting anomalous traffic in the first type is simple, as most of the anomaly detection algorithms can directly be applied to the problem. However, for the second type of traffic, predicting when an event will occur consistently is very challenging. While there exist models for the traffic during activity periods, it is difficult to solve the problem of finding anomalies just by trying to learn the function the data distribution follows; if we consider time-series data. To detect malicious traffic of MTC devices, we will consider the traffic model proposed by 3GPP, which suggests that traffic of MTC devices' connection after detecting an event follows the $\beta(3,4)$ probability distribution [73]. Accordingly, we assume that (1) normal traffic follows the $\beta(3,4)$ probability distribution; (2) the detection events do not overlap (i.e., each event is independent of the other). We argue the latter assumption by the fact that most of the devices run only one type of application, which monitors a single event.

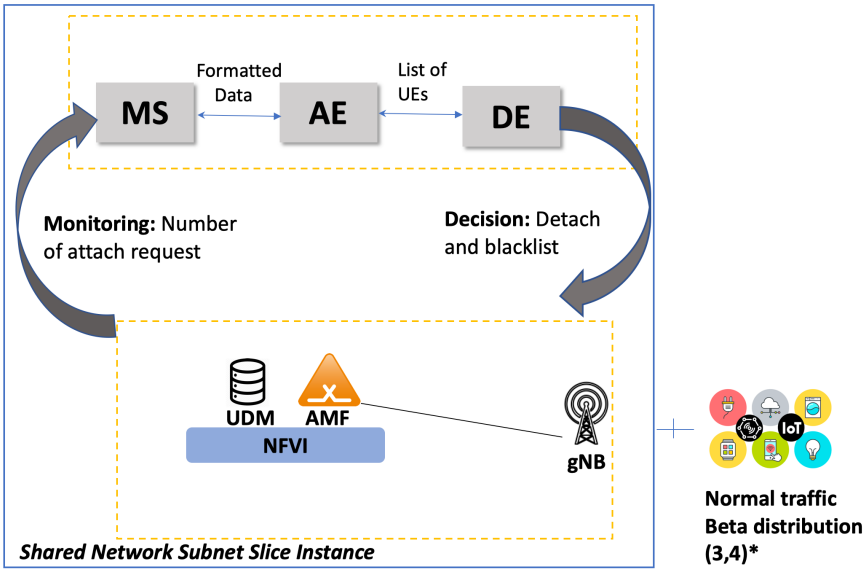


Fig. 4.1 The high-level vision of envisioned system architecture.

Fig. 4.1 illustrates the high-level vision of the system, which is composed of the closed-control loop components (MS, AE, and DE) that interact and protect the shared sub-slice components (5G CN and gNBs) against DDoS attacks. Here, we focus on protecting the AMF as it is the entry point of the 5G CN and treats all the Attach Requests coming from the different gNB under its control.

The closed-control loop is composed of three entities: MS, which collects information from the AMF, AE, which uses ML to predict attacks, and DE, which

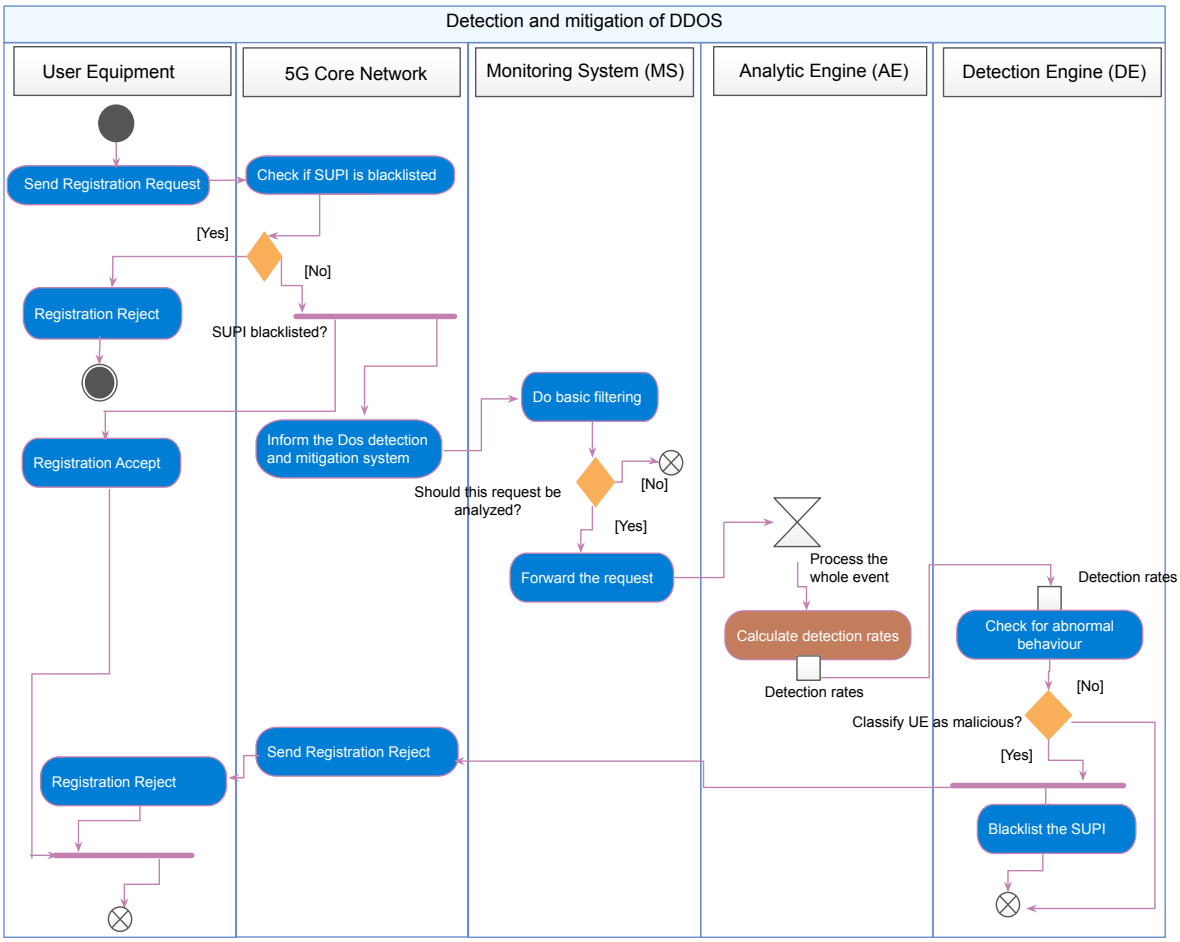


Fig. 4.2 Interaction of the mMTC network slice and the closed-control loop to detect and mitigate attacks.

reacts to the alert sent by AE by acting on AMF (block and blacklist UE). The control-control loop runs as software and can be co-located with the orchestrator managing the life cycle of the shared sub-slice [74]. It should be noted that AMF, via an Element Manager (EM), exposes API for an orchestrator or a manager to extract and monitor information on the AMF's functioning or to change the configuration of the latter. In the proposed framework, MS monitors the Attach Request received by the AMF, while DE requests AMF to send Registration Reject to suspected devices.

Fig. 4.2 highlights the interaction among the different actors involved in detecting and mitigating DDoS attacks: the mMTC network slice components (UEs and 5G CN) and the closed-control loop elements (MS, AE, and DE). It is worth noting that the closed-control loop runs in parallel to the mMTC network slice elements and only monitors Attach Requests to detect and mitigate attacks. In the considered scenario, the MTC devices (or UEs), when detecting an event or

participating in an attack, first send a Attach Request to AMF. The latter must first authenticate the devices and then give them access to the network resources (register the device), mainly to the data plane, to send data to the remote application. During the authentication process, the AMF checks with the Unified Data Management (UDM) if a device is blacklisted or not. To recall, UDM is the 5G CN function, which stores subscribers' information (Subscriber Permanent Identifier -SUPI - Quality of Service -QoS- Policy, the key k, Operator key, etc.). A device is blacklisted if it has participated in an attack. Meanwhile, MS, via the EM/AMF API, monitors the Attach Requests received by AMF. MS filters the data to extract the needed information, such as timestamps and SUPI. This information is communicated to AE that processes the whole event (attach period that may correspond to an attack) in order to classify if the event corresponds to an attack or not. When the event finishes, AE communicates the list of involved UEs; for each UE, a probability of being part of the attack is included. DE then mitigates the attack by requesting AMF to send a Registration Reject message to UEs having a high probability of being in the attack while adding the concerned UEs to a blacklist maintained by the UDM. The following sections will detail each component of the closed-control loop with an important focus on the AE functions, as they represent the critical components of the framework.

4.4 Closed-control loop: Attacks detection

4.4.1 Monitoring System (MS)

MS collects data from AMF on every Attach Request received from the MTC devices. This data is accessible through the API exposed by EM of AMF. For each Attach Request, MS extracts the device identifier SUPI and a precise timestamp. Indeed, in the 5G protocol, each UE is identified with a unique identifier called SUPI. The latter is encrypted to provide better privacy and prevent the IMSI catcher attacks that were popular on previous-generation telecommunication protocols. The SUPI should not be transferred in clear text over the RAN except routing information, e.g., Mobile Country Code (MCC) and Mobile Network Code (MNC) [75]. For this reason, it is challenging to identify devices from the traffic received on the radio layer; hence our solution has to intervene at the 5G CN (i.e., AMF), which can decrypt the SUPI information. The extracted information is then forwarded to AE via a communication bus based on the Publish/Subscribe concept.

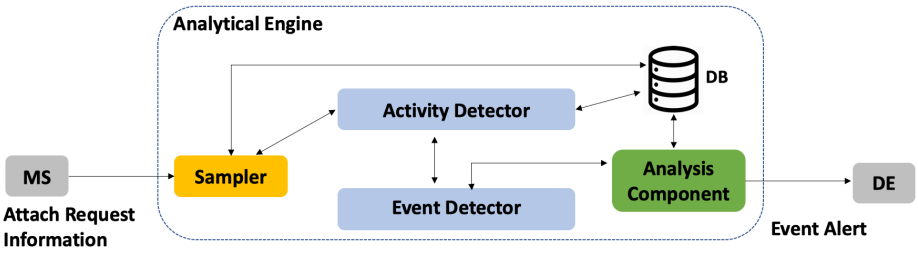


Fig. 4.3 AE's components.

4.4.2 Analytical Engine (AE)

Fig. 4.3 illustrates a detailed vision of the AE components, which are: Sampler, Activity Detector, DataBase (DB), Event Detector, and Analysis components. They interact together to: (1) detect when an event starts and ends. It can correspond to MTC devices report (normal traffic) or attacks; (2) analyze the event to detect if it is normal or abnormal traffic; (3) compute the detection rate for each device (probability that a device has participated in the attack) and send a report to DE. We decided to separate the event detection from event analysis to improve performances and consider all the relevant data when running the overall attack detection algorithm. Indeed, we decided to detect activity periods (i.e., events) in the network traffic and only feed data to the ML algorithm at the end of an event, which provides the advantage that the resource-intensive component (detection analysis) runs once every event. We also consider two corner cases: (1) after a duration clearly greater than the maximum length of an event; (2) when peak traffic exceeds a limit indicating that it is definitely an attack. For both cases, we tag the devices as malicious.

The only downside of separating event detection from the analysis is that UEs participating in an attack will not get banned right away when the attack starts. However, since the damage in DDoS attacks stems from their duration in time, the devices will get disconnected and blacklisted a few seconds or minutes after the event starts by DE. The detection algorithm does not need to run in real-time, and it can look at data of the whole event.

4.4.2.1 Event detection

To detect activity periods, we first calculate the rate of Attach Request. To this end, we devise a new component called the Sampler, which receives data that reaches AE from MS and emits data periodically by grouping the Attach Requests in time intervals of a fixed length. Fig. 4.4 illustrates how the Sampler works. The

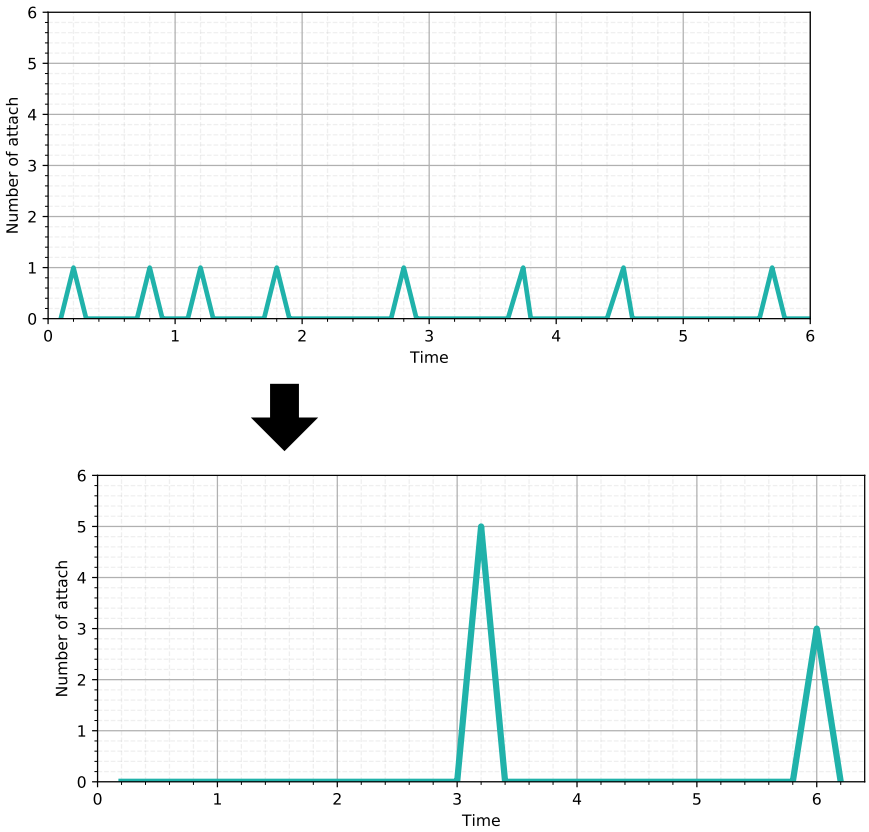


Fig. 4.4 Sampler: attach requests on time intervals of a fixed length.

upper part of the figure shows the real Attach Request timestamp reception, while the down part of the figure shows the output of the sampler that groups the Attach Request every 3 seconds. The Sampler outputs allow detecting the start and end of an event. If there is traffic, and we are not already in an event, we consider that an event has started. If we are in an event and detect that the system has not received traffic for a given duration, we assume that the event has ended. We use a DB for storing all the data relevant to the event. To have a modular system, we separate the Activity Detector (the part that detects whether there is traffic or not) and the Event Detector (the part that delimits an event's start and end timestamps).

As stated earlier, a DB is used to store information about the event, namely the number of Attach Requests received for each event, a list of SUPI values that identify the devices that emitted each Attach Request, and timestamps. The DB is accessible and used by: (1) the Sampler to store pairs (*timestamp*, *supi*), retrieve the largest timestamp stored, retrieve all the data stored, and delete all the data stored; (2) Activity and Event Detectors to store and edit a boolean value *is_event*. It is worth noting that a relational database is not an optimal choice in terms of the database model. A better choice is a Time Series DB, as we will store time events. After many considerations, we finally opted for a key-value database.

Table 4.2 The design of our key-value database.

Key	Type	Used By
devices	**sorted set**	Sampler
is_event	**string**	Activity and Event Detector
last_attach_requests	**sorted set**	Activity and Event Detector

Indeed, our processing of timestamps will still be efficient, as we will be using a sorted set, and we will be able to store the boolean value needed by the Activity Detector. This avoids the need for multiple DBs inside AE. The design of the database is highlighted in table 4.2.

4.4.2.2 Event analysis: ML Algorithm

The core component of AE is the Analysis Component, which receives data about one event, including the SUPI identifiers of all the devices that sent an Attach Request; then, it calculates a percentage for each device being part of an attack. The Analysis Component output (i.e., a percentage) should be zero ideally for normal traffic, as no abnormal behavior is present. When some devices misbehave and send Attach Requests that clearly do not correspond to normal traffic, the Analysis Component output should be higher than 0. The detection rate should increase as the traffic rate increases above the normal level. We also want high detection rates (preferably 100%) for traffic likely to cause a DDoS attack.

Many algorithms can be applied to solve this problem. However, not all of them can be used as we have two important considerations:

- We already have a model for our data, the $\beta(3,4)$ probability distribution, but the goal of ML algorithms is usually to estimate a function we do not know from its inputs and sometimes its outputs (in the case of supervised learning).
- New equipment can be introduced so that the event lengths can vary. Hence, the envisioned algorithm should not detect these changes as anomalies.

While malicious traffic that triggers DDoS is easily discernible from normal traffic, it is hard to evaluate the detection rates for anomalies that are not blatant attacks (i.e., when the traffic rate is just a bit above the prediction interval bound, for example).

In what follows, we enumerated a list of algorithms that can be used for this purpose.

- Statistical tests, checking the mean, median, extremums, variance, and more, and comparing them with the properties of the $\beta(3, 4)$ distribution. Note that we will compare our solution against a statistical-based solution.
- k-NN, SVM, Isolation Forests, Decision Trees, Gradient Boosting, etc.
- Deep Learning algorithms based on Neural Networks (Auto Encoders, Recurrent Neural Networks).

We discarded Deep Learning based algorithms as learning from data is not something we want. It would break our second consideration, and the change in the number of connected devices could trigger wrong predictions. Further, we believe that a neural network trained on normal traffic will not give a gradually higher detection probability on outliers because it does not consider outliers as elements that are more distant from normal data. It instead works with the combination of linear and non-linear mathematical operations. We considered the algorithms that can calculate prediction intervals, which are intervals that likely include our data. Indeed, if we can get an interval that includes our data, we can use the distance from its upper bound to calculate a prediction percentage. The most popular ML algorithm for calculating prediction intervals is Gradient Boosting. It is an extension of Boosting, where the additive generation of weak models is based on the gradient descent algorithm over an objective function [76]. The Gradient Boosting decision tree algorithm has demonstrated great performances on many machine learning tasks, including global contests [77]. It produces a prediction model in the form of an ensemble of weak prediction models, often decision or regression trees. It combines weak "learners" into a single strong learner in an iterative fashion, lowering the error estimated with the chosen loss function at each stage.

4.4.2.3 Event analysis - Data generation and model training

We generated data for training the ML algorithm in the same format as the data we run our detection on, i.e., timestamps of Attach Requests in a simulated event. We used a 5G testbed with emulated UE to emulate an event and generate data. The details on the 5G testbed are provided in section 4.5.

Algorithm 2 is used for generating data. The complexity of this algorithm is $O(n^2)$, where $n * n$ operations are required for inputs, while each outer loop runs n times. We consider that the average number of equipments that would take part in an event is $\frac{duration \times 3}{7}$. We argue this by the fact that the mean of the $\beta(a, b)$ distribution is $\frac{a}{a+b}$, where a=3 and b=4. We generate random numbers following

Algorithm 2 Data Generation

procedure GENERATE_EVEN_DATA (int duration, int sample duration)
▷ The average number of equipments that would connect during an event of length duration
    $num\_equipments = floor(\frac{duration \times 3}{7})$
    timestamps is an array of *num_equipments* reals
    for $i \leftarrow 1$ **To** *num_equipments* **do**
        $timestamps[i] \leftarrow random\_\beta_{3,4}() \times duration$
    end for
    Sort the timestamps array
▷ At this point, timestamps has real values in the range [0,duration] *sampled* is a list of integers
    $k \leftarrow 0.0$
    $i \leftarrow 0$
    $sampled\_index \leftarrow 0$
    while $k <= duration$ **do**
        $j \leftarrow 1$ ▷ Count the number of samples in the range $[k, k + sample\_duration]$
        $count \leftarrow 0$
        while $j < num\_equipments$ **and** $timstamps[j] < k + sample\_duration$ **do**
            $count \leftarrow count + 1$
            $j \leftarrow j + 1$
        end while
        end while
        $i \leftarrow j$
        $sampled[sampled\_index] \leftarrow (k, count)$
        $sampled\_index \leftarrow sampled\_index + 1$
        $k \leftarrow k + sample\_duration$
    end while
    end while
    Output sampled
end procedure
end procedure

the $\beta(3,4)$ probability distribution. These numbers represent the expected timestamps of Attach Requests. We send Attach Requests and sleep for a duration equal to the difference between the random numbers generated to simulate the event. When this step ends, the data is stored in a file, where each entry corresponds to a sample period. Let us note by S the sample vector defined as $\{s_1, s_2, s_3, \ldots, s_n\}$, where s_1 corresponds to the sample period 1, and noted in an entry of the file as (t_1 - 0) ($t_1 = \Delta$); s_i is ($t_{i+1} - 0$) ($t_{i+1} = i + 1 * \Delta$). The duration of the sample period is identical and obtained as follows: $\Delta = \frac{Total\_Duration\_Event}{n}$. For the training phase, we will use the data generated earlier to train a Gradient Boosting algorithm to predict for each sample s_i, the upper bound of the number of Attach Request expected during that sample period; we note it by $Predict_i$. Here we are interested in the upper bound value as it corresponds to the maximum intensity of normal traffic; hence exceeding this value means that we are most probably facing a DDoS attack. Once the training is done, we obtain a vector P that contains n predicted values corresponding to the maximum expected Attach Requests per sample period. The $Predicted_i$ values are also stored in the file with each corresponding entry. The file will be used later as an input to analyze an event and calculate the detection rate.

4.4.2.4 Event analysis - Detection

During the detection step, the event detected by the Sampler is stored in the DB. The event is organized in sample periods equal to n with a duration Δ (the same value used for the training phase). This will allow us to normalize the number of samples of an event since each event has a different duration period, and the $\beta(3,4)$ intensity depends on the duration. Thus, we do not need to train the model using different durations, as the normalization step will allow training on a single duration corresponding to $\beta(3,4)$ distribution. Since the event has been organized by sample periods with the total number of Attach received during the sample period as well as the SUPI of the UE, we use the ML model (mainly the file obtained in the precedent step) to extract the $Predict_i$ values for each sample period. Then, we derive another bound for each sample period as follows: $Predict_i$ $\times$ (AE_DETECTION_THRESHOLD - 1); a limit above which any traffic gets a 100% detection rate and gets classified as malicious. For each sample period, we calculate the detection rate as follows:

For x_i, the number of Attach Requests in the sample period $[s_i]$:

If $x_i \leq Predict_i$, then $detection_i = 0$.

Else,

$detection_i = min(1.0, \frac{x - Predict_i}{Predict_i \times (AE\_DETECTION\_THRESHOLD - 1)})$

where $0 < detection_i < 1$.

Let us suppose that during an interval (Δ), the number of Attach Requests is greater than the predicted one, meaning abnormal traffic. In this context, to estimate the other bounds, from which detection values are 100%, we use $predicted_i \times (AE\_DETECTION\_THRESHOLD - 1)$; the predicted value is multiplied by a constant, corresponding to the rate between the distance of x from the $Predict_i$. This is needed to reduce the ML errors impact and hence reduce the False Positives. Note that if $detection_i$ is higher than zero, all the involved UEs during that sample period are considered as part of a DDoS attack with rate $detection_i$.

4.5 Closed-control loop: attacks mitigation

DE is the decision-maker of the closed-control loop system. It receives data from AE and decides the actions to take for UEs that emit abnormal traffic. DE gets as input a list including the suspected UEs (SUPI) and their corresponding detection rate values (i.e., $detection_i$) belonging to the attacks. We devise two versions of DE. The first solution blacklists devices if their calculated prediction is higher than *DE_DETECTION_THRESHOLD* (i.e, a configurable parameter). The lower the threshold value, the higher the probability that devices are blacklisted. Therefore, the network operator would use lower values in order to be more strict, but in turn, it may increase the false positive. To avoid having high false-positive results, we introduce a second solution that relies on three thresholds. This solution considers the whole event and classifies the received list of UEs into three categories: F_1, F_2, and F_3. DE calculates how many UEs have obtained a detection rate ($detection_i$) higher than 0.8 and assigns it to the first category, namely F_1. The second category includes UE having a detection rate between 0.3 and 0.8. This category corresponds to F_2. The remaining UEs are included in F_3. Then, DE checks if, in the event, a significant part of the devices had higher than usual detection rates. As a result, different decisions are to be taken:

- First, if $F_1 \geq F_2$ and $F_1 > F_3$, DE blacklists all the devices by adding their SUPI values to a table of blacklisted values, and disconnect them from the network.

- Second, if $F_2 \geq F_1$ and $F_2 \geq F_3$ and $(\forall x \in detection, x > 0)$, DE adds the

SUPI values to a table named "non-trusted devices". Each UE belonging to this table has a counter named T_{imsi}. The counter is increased by 1 each time the UE is involved in abnormal traffic that is not blatantly an attack. The counter is incremented until it reaches a value that yields to blacklist the device.

- Third, DE ignores the alert sent by AE, and it will do nothing.

The reason behind using 3 categories is based on the AE analysis and the results accuracy of the employed ML algorithm. Indeed, we notice that when the detection values associated with the connected devices sent by AE are high, the devices' generated traffic does not follow the Beta distribution. Hence, they should be immediately blacklisted as they present abnormal behavior, justifying the need for the first category. The latter is considered a red alert, and the associated devices SUPI are blacklisted. However, when the values are neither too high nor too low, it means that the traffic is almost close to the Beta distribution. In this case, the detected devices have malicious behavior, or the values are due to technical failure. So we introduced the second category, which means that the devices are not blacklisted, but DE will memorize the associated SUPI for future events. If the devices are classified in the second category more than 2 times, they will be considered malicious and moved to the first category to be blacklisted. The last category corresponds to the detected traffic being very close to the Beta distribution. This case can be either an error in the ML calculation or a delay in sending the Attach Request.

4.6 Performance Evaluation

4.6.1 The test platform

To validate the proposed zero-touch security management system, we have used a 5G testbed deployed at EURECOM[1]. The testbed has been developed and used in many 5G European projects such as 5GEve[2] and 5G!Drones[3]. We have implemented the closed-control loop components (i.e., MS, AE, and DE) and a Element Manager (EM) on top of the AMF. The latter exposes API to (1) MS to monitor the Attach Request message; (2) DE to detach and blacklist UEs involved in an attack. Fig. 4.5 illustrates the different technologies used to implement the above-mentioned components. As a quick reminder, the roles of the different components are:

[1]https://www.youtube.com/watch?v=90SRV9ZpPVot
[2]https://www.5g-eve.eu/
[3]https://5gdrones.eu/

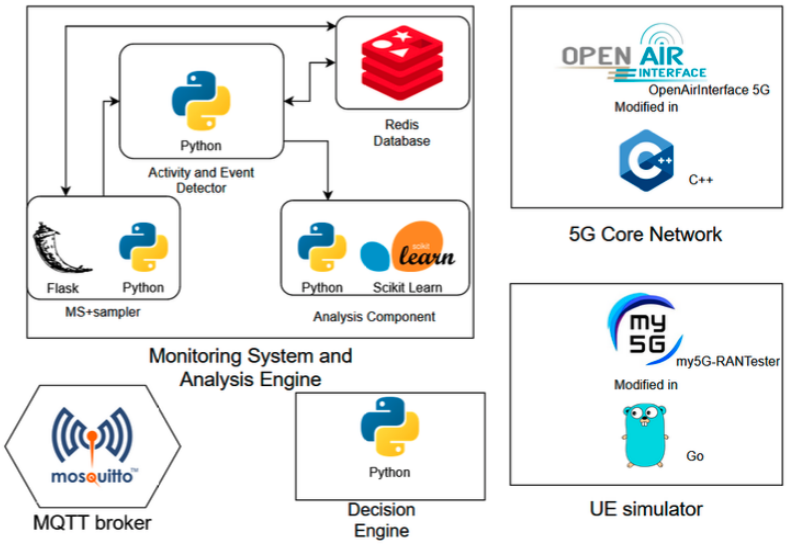


Fig. 4.5 Test platform and technological components

- MS and sampler: MS is the first component to receive traffic from the 5G CN. It performs basic filtering on it. While the sampler does sampling of the input data, it receives information on Attach Requests as they are received (with no guaranteed periodicity) and emits periodic data, with the number of Attach Requests received in time intervals of a given length.

- Activity and Event Detectors: These components receive the sampled data and should detect an event. For each event, these components only emit data at its end, with the number of requests on each time-slice and all the UEs that emitted traffic during the event.

- Analysis Component: This component runs the ML algorithm on the given data, calculating a detection rate for each time-slice (for all devices in the time-slice).

- DE: This component receives data from the Analysis Component and decides which devices should be disconnected from the network and then blacklisted.

- MQ Telemetry Transport (MQTT) Broker: is used to implementing the communication bus between the different components of the closed-loop control system, and between the closed-loop control system and the AMF.

Regarding the UE, we used and updated a 5G UE emulator, my5G-RANTester[4], to be able to send a high number of UE Attach Request messages in parallel to

[4]https://github.com/my5G/my5G-RANTester

Table 4.3 The configurable parameters in our system.

Parameter Name	Default Value	Component
INTERVAL_LENGTH	**6.0**	Sampler
UNCHANGED_INTERVALS_COUNT	**4**	Sampler
REQUEST_THRESHOLD	**1**	Detector
MAX_EVENT_DURATION	**600**	Detector
DE_DETECTION_THRESHOLD	**0.35**	DE

simulate an attack or normal traffic. Indeed, my5G-RANTester is a tool for emulating control and data planes of the UEs and gNB. my5G-RANTester follows the 3GPP Release 15 standard for RAN. By using my5G-RANTester, it is possible to generate different workloads and test several functionalities of a 5G CN, including its compliance with the 3GPP standards. Scalability is also a relevant feature of the my5GRANTester, which can mimic the behavior of a large number of UEs and gNBs simultaneously accessing a 5G CN. Currently, the wireless channel is not implemented in the tool. The AMF and 5G CN components are based on OpenAirInterface (OAI) [5].

As described earlier, AE and DE use several parameters that need to be tuned to optimize the different steps of the event analysis. These parameters are summarized in Table 4.3 and defined as follows:

Δ(Sec): Length of the sampling interval. A message is sent from the sampler every Δ seconds, including the number of UEs that connected in the last interval of time of this length.

***UNCHANGED_INTERVALS_COUNT*:** Number of intervals without activity after which an event is marked as finished. This parameter is used to detect the end of an event. For instance, if we assume its value is equal to 4 and *INTERVAL_LENGTH* is 6, then we can consider an event has ended if there is very little to no activity for 24 seconds (or four intervals) while an event was ongoing.

***REQUEST_THRESHOLD*:** If there are fewer requests than this value in an interval of time, we assume no activity.

***MAX_EVENT_DURATION (Sec)*:** If the end of the event is not detected after *MAX_EVENT_DURATION*, we consider it is an attack.

***DE_DETECTION_THRESHOLD*:** a threshold used by DE to determine if a device should be banned from the network or not. Any detection rate higher than this value leads the system to ban the device.

[5]https://openairinterface.org/

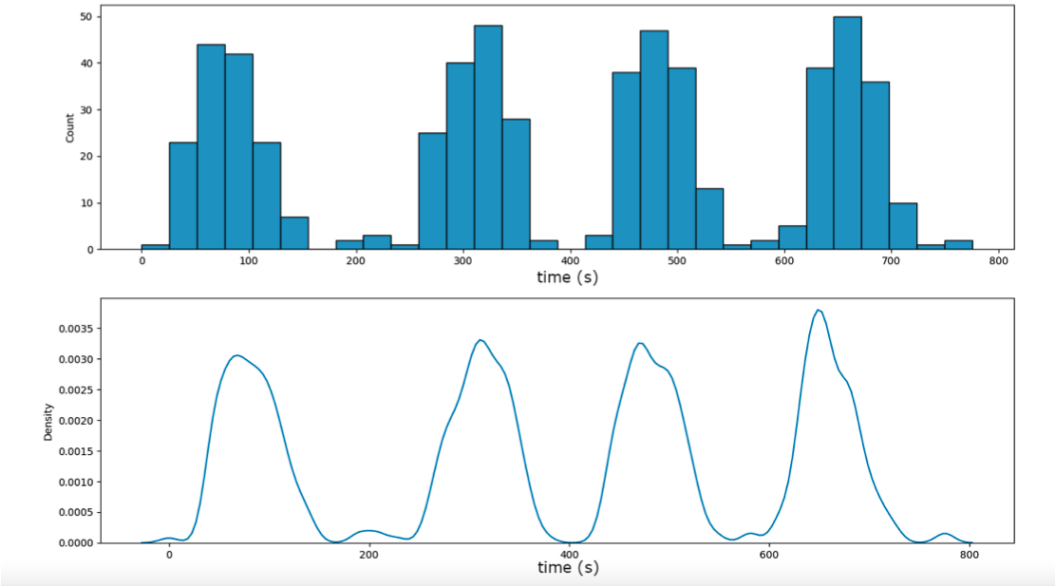


Fig. 4.6 Normal traffic - four events.

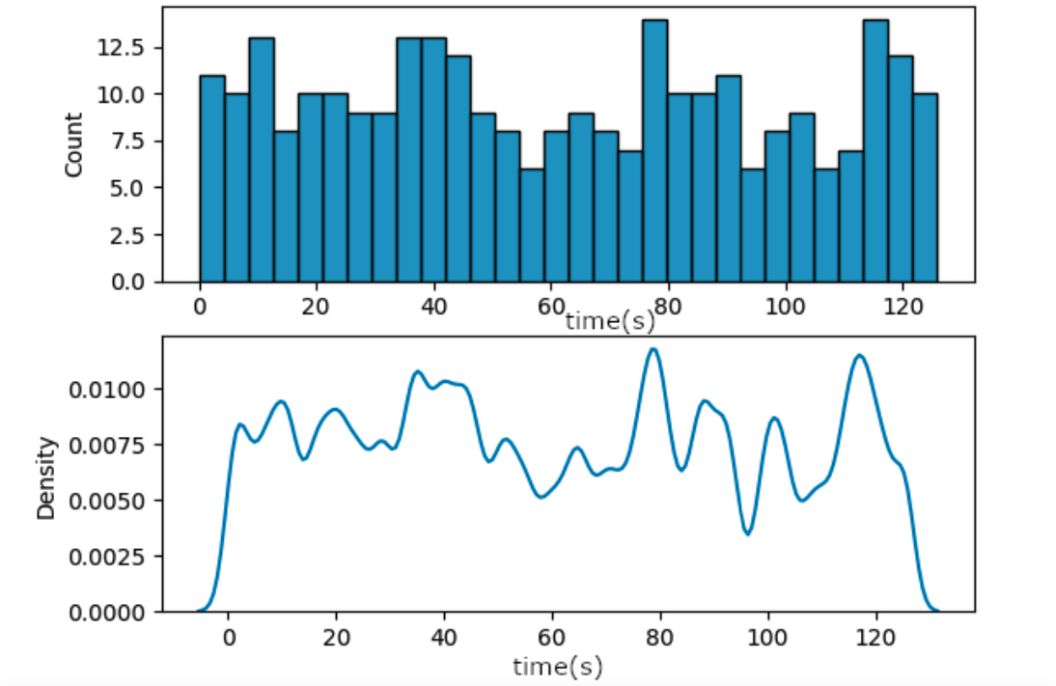


Fig. 4.7 Malicious traffic.

Algorithm 3 Event simulation

procedure SIMULATE_EVENT(int duration)
    $num\_equipments = floor(\frac{duration \times 3}{7})$
    $timestamps$ is an array of $num\_equipments$ reals
    for $i \leftarrow 1$ **To** $num\_equipments$ **do**
        $timestamps[i] \leftarrow random\_\beta_{3,4}() \times duration$
    end for
    $currSUPI \leftarrow MCC + MNC +$ "0000000001"
    $Sleep(timestamps[0])$
    for $i \leftarrow 1$ **To** $num\_equipments - 1$ **do**
    $SendAttachRequest\_ASYNC(currSUPI)$
    $currSUPI \leftarrow str(int(currSUPI) + 1)$
    Sleep(timestamps[i])
    end for
    $SendAttachRequest\_ASYNC(currSUPI)$
end procedure
end procedure

Regarding the testing conditions, as we do not have datasets that include both positives and negatives, we generate traffic corresponding to non-malicious using the $\beta(3,4)$ probability distribution, while for abnormal traffic, any other distribution can be used. Besides, we are considering timestamps as our only feature; therefore, we cannot differentiate between two Attach Requests received at a moment when traffic is dense.

To test our system under realistic circumstances, we leveraged the my5G-RANTester with a script that emulates real UE traffic (control plane) to communicate with 5G Core Network components. We used the emulator to simulate an event with different chosen SUPI values. The generated traffic is similar to what real UEs would generate. It follows the 3GPP specifications 15. Algorithm 3 generates the traffic featuring:

- Simulating an event: Sending Attach Requests that follow the Beta-distribution
- Simulating an attack: Sending Attach Requests that follow the uniform distribution
- Simulating a Attach Request of a single UE

Algorithm 3 is used for simulating an event. It is very similar to the algorithm that generates data. The complexity of this algorithm is $O(n)$, where n operations are required for input, and the outer loop runs n times. Here, n corresponds to the number of devices involved in an event. Figs. 4.6 and 4.7 show a visualization of traffic likely corresponding to a normal (four events and attack) and malicious one, respectively.

Last but not least, readers may see a video[6] showing a demonstration of all the components, i.e., closed-control loop as well as AMF and UEs, working together to detect DDoS attacks. The following section will provide some results on the model accuracy obtained via the experiments.

4.6.2 Test results

To evaluate the performance of the proposed framework, we focus mainly on the performance of the attack detection algorithm, which is the key function of the closed-control loop. To this end, we evaluate the Gradient Boosting algorithm to detect DDoS attacks accurately and compare its performance with a statistical method. Like the Gradient Boosting solution, the statistical method is applied at the end of the event. Based on the event duration, the statistical method defines a limit function using the mathematical function of $\beta(3, 4)$ to compare the different points by the report to this limit; all the points exceeding this limit are considered as a possible attack. The main differences compared to Gradient Boosting are: (i) it needs the exact duration of the event; (ii) it uses the $\beta(3, 4)$ function to deduce the limit.

Gradient Boost We measure the accuracy of the Gradient Boosting algorithm in front of normal and abnormal traffic. On normal traffic, the accuracy denotes how often the system yields a detection rate of UEs that is greater than zero. This does not mean that these devices will get banned, but ideally, a value of 0 should be returned for all the devices emitting normal traffic. To evaluate our model on normal traffic (True Positive (FP) = False Negative (FN) = 0), we generate data for 500 normal events and run our detection algorithm on each of them. We then counted the number of UEs for which we obtained a greater-than-zero detection rate versus the total number of UEs in all the events. We generate event durations randomly (between 30 and 250 seconds). Regarding malicious traffic (True Negative (TN) = False Positive (FP) = 0), we also ran 500 tests, but this time, between 7 and 15 Attach Requests are received every 6 seconds, for a total duration that is random between 30 and 250 seconds.

Fig. 4.8 allows visualizing the results for normal traffic. The points correspond to the event data, while the green line is the anomaly interval. If a point is outside the limit (in green), it will be assumed as an attack. For normal traffic, the accuracy is computed as $1 - \frac{FP}{TN+FP}$. Hence, the results show an Accuracy on normal traffic of 96.76859478052322%. We expected this result as the interval used in our training data includes around 95% of the data in the training dataset,

[6]https://www.youtube.com/watch?v=QzCmGfwtDLAt=7s

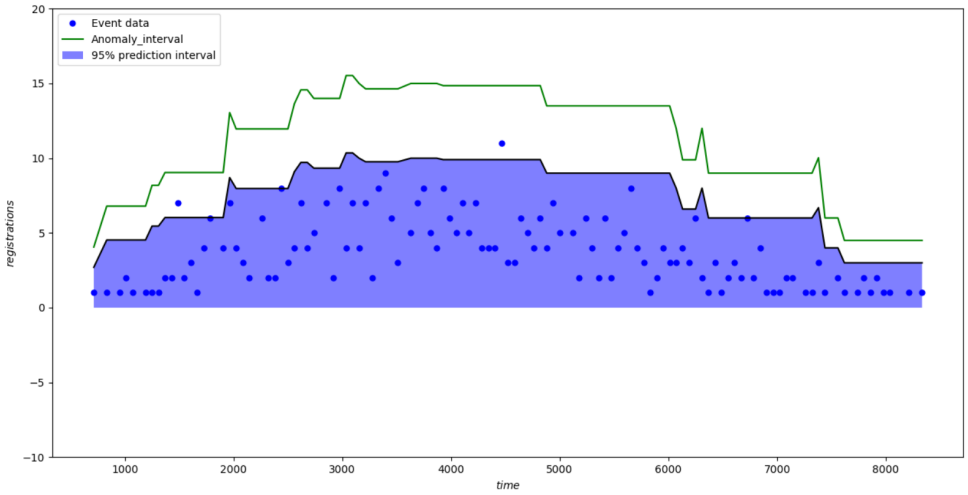


Fig. 4.8 The result of the detection algorithm over normal traffic.

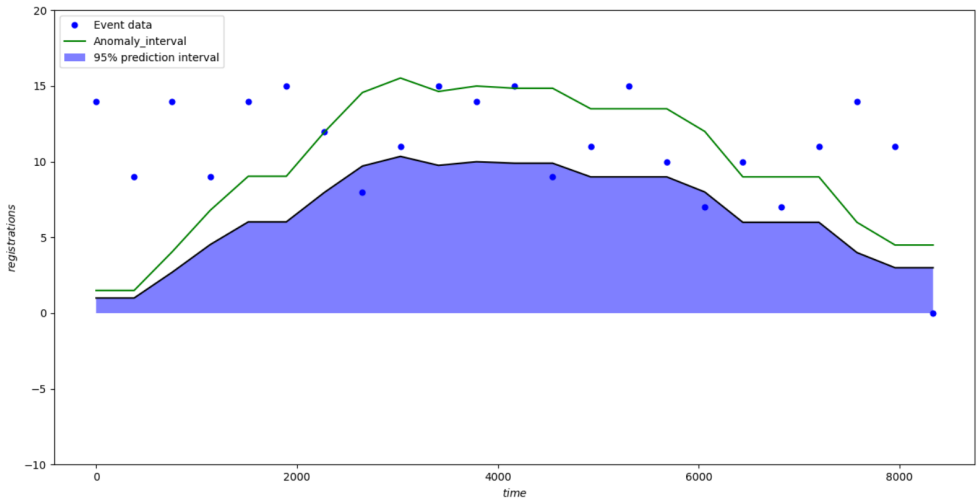


Fig. 4.9 The result of the detection algorithm over abnormal traffic.

as depicted in Fig. 5.15.

Fig. 4.9 illustrates the results for malicious attacks. For this case, the accuracy is computed as $1-\frac{FN}{FN+TP}$. Hence, the results show an accuracy of 83.63319140762557%. This represents an excellent result as banning a relevant part of the devices taking part in a DDoS attack is enough to mitigate it. We recall that this is just the detection rate calculated by the AE component, the final decision regarding the devices that should be banned is taken by the DE component.

Table 4.4 shows the performance of the Gradient Boosting-based solution when modifying the *AE_DETECTION_THRESHOLD* value. It is worth recalling that this value is used to derive the detection rate and corresponds to a protecting gap to reduce the impact of the ML prediction and hence reduce the FP value. We remark that the value allowing to reduce both FP and FN is 2.0. Also, when

Table 4.4 Impact of the AE detection threshold on the Gradient Boosting accuracy

AE_DETECTION_THR.	1.2	2.0	3.0
Normal event	82.98654	96.76859	97.64853
Abnormal event	81.91305	83.6331914	61.86956

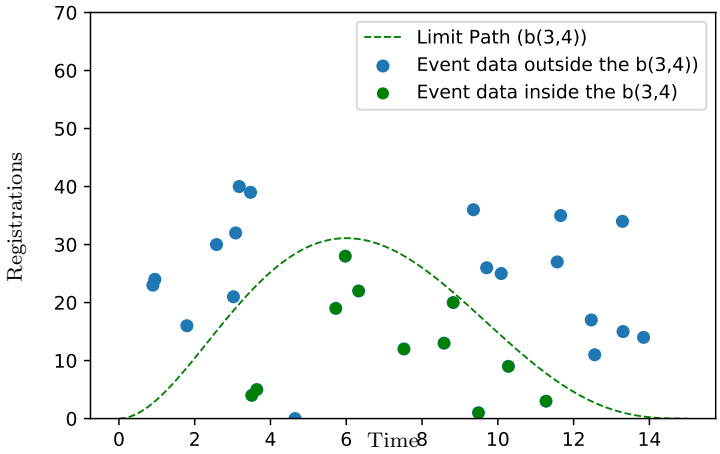


Fig. 4.10 The result of the statics method over abnormal traffic.

the *AE_DETECTION_THRESHOLD* value increases, FP is reduced as the FN increases, whereas when it is reduced, both FP and FN increase.

Statistical Method For the sake of comparison, we used the same scenarios as for Gradient Boosting to generate normal and abnormal traffic. Then, we applied the statistical method and verified its accuracy in detecting attacks. Fig. 4.10 illustrates the usage of the statistical method in case of an abnormal event. The discontinue green line shows the $\beta(3,4)$ curve obtained according to the event duration. The $\beta(3,4)$ curve allows us to have a limited path from which all the outside points are considered anomalies, hence potential attacks. The statistical method's results show that 36.0% of the Attach requests are not following the β(3,4) distribution (they are out of the limit path). Therefore, they can be considered potential attacks. On the other hand, Fig. 4.11 presents a test of a normal event. The results show that 90.0% of the Attach Requests follow the β(3,4) distribution. The accuracy of the statistical method to detect anomaly are : $((1-\frac{FP}{TN+FP}$ = 84.21052 % (normal traffic), $1-\frac{FN}{FN+TP}$ = 57.142857 % (abnormal traffic))). We remark that these values are weaker than the ones obtained with the Gradient Boosting algorithm. We argue these differences by the fact that the duration estimation has a strong impact on the statistical solution. The shape of the $\beta(3,4)$ curve changes drastically according to the duration (noted D), which seriously im-

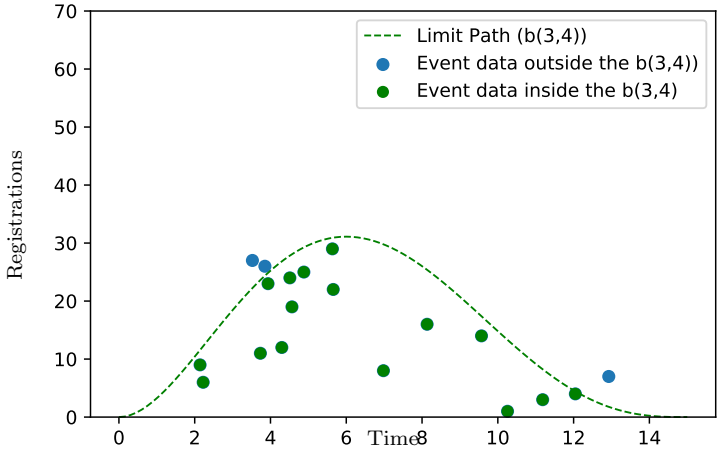


Fig. 4.11 The result of the statics method over normal traffic.

Table 4.5 The accuracy of the statistical model considering different Duration values.

Method	GB	Static		
		D - Δt	D	D + Δt
Normal traffic	96.76859	45.18924	84.21052	80.24568
Abnormal traffic	83.633191	30.4156	57.142857	51.86854

pacts the accuracy. For instance, we change the duration by +/- $\epsilon = 2sec$, and the obtained results are summarized in Table 4.5. We see clearly from this table the impact of the duration on the accuracy as a small error on the duration drastically yields a drop in the accuracy. Particularly, if the duration is less than the real one, many points will be out of the curve. In the Gradient Boosting algorithm, we do not have this concern, as the latter normalizes the sample period duration and uses the trained model to detect the interval.

Decision Engine Regarding DE performances, we evaluated the first version that relies on a single threshold *DE_DETECTION_THRESHOLD*. Accordingly, in this section, we evaluate the *DE_DETECTION_THRESHOLD* impact on the number of blocked devices. Table 4.6 shows the number of banned UEs for three values of the *DE_DETECTION_THRESHOLD*. As expected, we remark that lower threshold values (ex. 0.1) are very conservative, which allows blocking more UE.

Table 4.6 Impact of the DE Detection threshold.

DE_DETECTION_THR.	0.1	0.35	0.8
Nb (Normal event)	3	1	0
Nb (Abnormal event)	21	16	10

While a higher threshold value (ex. 0.8) may be less strict and reduce the number of banned UE. We advise two solutions to fix the *DE_DETECTION_THRESHOLD* value. The first one considers the performance limit of the element to protect against DDoS attacks. In our case, we computed the response time of the AMF to Attach Requests while increasing their number. After a certain number of Attach Requests, we noticed that the AMF started to be very slow, which can be caused by a DDoS attack. Therefore, after some tests, we found that the value of *DE_DETECTION_THRESHOLD* equal to 0.35, which avoids reaching the number of Attach Request that yields bad AMF performances. Another solution is to use a dynamic threshold value that decreases or increases over time according to the number of consecutive events classified as an attack.

4.7 Conclusion

In this chapter, we introduced a zero-touch security management framework that aims to protect mMTC network slices from in-slice DDoS attacks. The proposed framework relies on a closed-control loop that tracks Attach Requests generated by mMTC devices to detect abnormal traffic and mitigate possible DDoS attacks. The mitigation process is enforced through disconnecting and banning suspected devices. The attack detection algorithm uses a ML technique, specifically Gradient Boost, to create a prediction interval that is likely to include normal traffic in our system.

It then calculates, for every sample that is outside the interval, a metric that depends on its distance from the bound of the interval; this metric is called the detection rate. The decision engine uses this metric to take action to mitigate attacks, which consists in banning and disconnecting devices from the network to prevent them from conducting similar attacks in the future. The proposed framework has been implemented using a 5G testbed. The obtained results demonstrate the closed-control loop's ability to predict and mitigate DDoS attacks efficiently.

CHAPTER 5

Toward Securing Federated Learning against Poisoning Attacks in Zero Touch B5G networks

5.1 Introduction

Fifth-generation and Beyond (B5G) networks are exponentially growing as key enablers of various applications related to multiple vertical industries [78]. These emerging applications are characterised by heterogeneous requirements, including ultra−low latency, high bandwidth and communication reliability, support of massive device density, etc. The ZSM architecture emerges to automate the management and orchestration of running network slices [79]. This requires a heavy usage of advanced Deep Learning (DL) techniques in a closed-loop way to auto build the suitable decisions, enabling it to meet network slices' requirements. Specifically, the traffic/data generated by network slices is first monitored and then used to build analytic functions (using DL mechanisms).

In general, the analytic functions aim to monitor network slices performances or SLA to predict/detect any degradation or violations. Noting that KPI of a network slice can be computation-oriented, network-oriented, or service-oriented [80]. While computation- and network-oriented KPIs can easily be monitored, through the NFVI manager and SDN controller, to build analytic functions.

However, service-oriented KPIs are difficult to share due to their confidentiality and privacy nature, since they are directly linked to a running vertical industry's application and service, such as the Internet Protocol (IP) address allocation, response/processing time of a particular VNF, and statistics on handled packets by a router.

In this context, FL is playing a vital role to train deep learning model in a collaborative way among thousands of network slice participants, while ensuring their privacy, and hence network slices isolation. In addition, FL allows to avoid transfer large amounts of monitoring data across the network. This is especially interesting in the ZSM context, in which some autonomic loops are installed on the edge, and would be likely to cause perturbations if they report all of their monitoring to a remote location. Specifically, rather than uploading their data to train their models, running network slices share only their local model parameters, during several rounds, with a central entity, e.g. Inter Domain Slice Manager, to aggregate them and build a global model. Thus, the central entity does not have direct access to the training data. Therefore, FL is more than required to create analytic functions about service-oriented KPIs of running network slices, while ensuring their confidentiality and privacy.

However, FL is vulnerable against poisoning attacks [81], where an insider participant, a malicious network slice, may upload poisoning updates to the central entity, so that it can cause a construction failure of the global model. For instance, a malicious participant may consider a poisoned latency values in building its local

model, in such way that the aggregated global model cannot then detect/predict latency-related SLA violations. Another example for an analytic function that implements an intrusion detection system, building poisoning models by one or several participant(s) can prevent the detection of system intrusions, even with presence. Thus, poisoning attacks may affect the global performance of FL-based models for running network slices as well as the whole system. Therefore, it is crucial to design security means to detect and mitigate such threats.

In this chapter, we design a novel framework to automatically detect malicious participants in FL process. First of all, based on real test bed, we generate a realistic dataset that focuses on the latency as service-oriented KPI of running network slices. This dataset is then exploited to build an analytic function about latency prediction, in federated way. In addition, the basic idea of our framework is to select dynamically one participant as a trusted entity, by leveraging deep reinforcement learning. The trusted participant will be in charge of identifying poisoning model updates, using unsupervised machine learning. The main contributions of this chapter are summarized as follows:

- We use OAI to generate a real dataset about the latency KPI of the AMF, as a VNF. This service-oriented KPI corresponds to the response time for handling UEs attach requests, when different configurations, such as available Random Access Memory (RAM) memory and number of CPU, are taken into account.

- Exploiting our dataset, we build a DL-based model in federated way between several running network slices. Our model enables to predict the latency of the AMF function and hence anticipating any latency-related SLA violations.

- We also build an online Deep Reinforcement Learning (DRL) model that selects dynamically a network slice as a trust participant, at each federated learning round, i.e. when participants send their local models towards the central node, based on several metrics such as their reputations (participants).

- At each FL round, the trusted participant applies dimensionality reduction scheme and unsupervised machine/deep learning to detect the poisoned model(s) and hence malicious participant (s).

This chapter is organized as follows. In Section 5.2, we present a review of related works. We provide a general background about the used mechanisms

to detect/mitigate poisoning attacks in FL. Section 5.3 describes the design and specification of the proposed framework. In Section 5.4, we evaluate and validate our framework. Finally, section 5.5 concludes the chapter.

5.2 Related Work

A few solutions have been designed to deal with poisoning attacks, when building learning models in FL way. These works can be classified into two main categories: works dealing with poisoned local models [82, 83] and those to deal with data poisoning of malicious participants [84, 85].

5.2.1 Local Model Poisoning

The objective of model poisoning attacks is to poison the local model updates of the FL clients, before sending them to the FL server or inserting hidden backdoors into the FL global model. This attack impacts the performance of the global model by giving mis-classifications or wrong predictions.
In [82], the authors demonstrated how FL model can be poisoned. They showed that any malicious client can introduce hidden backdoor functionality into the joint global model, e.g., to ensure that an image classifier model can predict labels, for some input data, which were introduced (labels) by the malicious client. The authors designed a new model-poisoning attack based on model replacement, and evaluate it on top of several assumption on the standard FL.
Another scheme is proposed in [83], to study the resilience of of distributed implementations of Stochastic Gradient Descent (SGD) against Byzantine failures, including network asynchrony, software bugs, attackers aiming to compromise the whole system as well as biases in local datasets. The authors first showed that current approaches do not tolerate Byzantine failures. Then, they proposed a resilience property of the aggregation rule that can ensure model convergence despite Byzantine participants.

A novel blockchain-based FL scheme was designed in [86], to deal with model poisoning attacks. This scheme is based on blockchain to create smart contracts and hence prevent malicious clients from being involved in FL process. Therefore, the central server may easily identify unreliable clients by executing smart contracts to defend against poisoning attacks. The numerical results showed that this scheme can efficiently detect and mitigate poisoning attacks, and thus securing FL in B5G networks.

Table 5.1 Comparison of poisoning attack detection solutions.

Works	Poisoning Attack type		Securing side		B5G	Used Technique
	Model Poisoning Attack	Data Poisoning Attack	Client	Server		
Yi Liu, *et al.* [86]	✓		✓		✓	Smart Contract and Blockchain
Peva Blanchard, *et al.* [83]	✓			✓		A resilience property of the aggregation rule
Phillip Rieger, *et al.* [87]		✓		✓		Analyzes the parameter updates of the model's output layer
Mustafa S.Ozday, *et al.* [88]		✓		✓		Adjusting the aggregation server's learning rate
M. Jagielski, et al. [89]		✓	✓			Model Robustifying: Trimmed versions of the loss function
Y. Zhao, *et al.* [90]	✓			✓		Generative adversarial network (GAN)
M. Subedar, et al. [91]		✓	✓			Probabilistic modeling of deep features
J. Steinhardt, et al. [92]		✓	✓			Training Data Filtering Input Manipulation Detection
Our Solution	✓	✓	✓		✓	**Supervised/Unsupervised Learning: Dimensional Reduction Algorithms , Deep Reinforcement Learning**

5.2.2 Data Poisoning

This type of attack is known as contamination of the training data, it takes place during the FL training phase of machine learning model. Generally, a malicious node tries to poison the training data by injecting carefully designed samples to compromise the whole FL learning process. This attack is also called, "dirty-label poisoning attack".
In [84], the authors first studied the threats that may target federated learning, in terms of sybil-based poisoning attacks. Then, they designed a new detection and mitigation scheme that enables to identify poisoning sybils based on clients' updates. The basic idea of this scheme is to use an adaptive learning rate at each client level, based on the similarity between inter-client contribution.
The authors also showed that their scheme may deal with the existing poisoning attacks, such as backdoor poisoning attacks and sybil-based label-flipping. Noting that label-flipping attack is a special case of data poisoning, where the labels of two input observations are flipped, while data features (inputs) are kept unchanged.
Another example of poisoning attack is backdoor data poisoning [85], where an adversary can modify individual features or small regions of the original training dataset (some pixels of images) to embed backdoors into the FL model. As an example, in the images processing, the attacker creates a stamp on an input image, so that the FL model behaves according to the adversary's objective, if the input contains the backdoor feature.

5.2.3 Discussion and comparison

In Table. 5.1, we compare between existing works according to several criteria such as, targeted poisoning attack, the used techniques, secure side (client or central node), and B5G context or not. Even there are some works addressing the challenge of how to deal with poisoning attacks in FL context. However, most of them focused on how to adjust learning parameters at the clients side, such as learning rate [84], or designing new rules for local models aggregation at the central node side [83], in order to be able in detecting malicious participants. In addition, these works dealt with either local model poisoning, or data poisoning attacks, and not with both.
In B5G networks context, few works have been proposed where are mainly based on blockchain, to prevent the participation of malicious clients [86]. However, blockchain will require more computing and storage resources in addition to those needed by B5G networks for their management and orchestration.
Additionally, the proposed framework is a costly solution.

5.3 Our Trust federated deep learning Framework

In this section, we describe our framework enabling to secure federated learning in B5G networks, against poisoning attacks, named TQFL for Trust deep Q-learning Federated Learning. The design of our framework comprises four main steps, starting from generating a realistic dataset to design a detection scheme for poisoning attacks: (*i*) The generation of realistic dataset about the AMF function's latency of running network slices, and its (latency) related parameters. (*ii*) Building a deep learning model to predict the AMF function's latency of each running network slice in federated way, in order to prevent any latency-related SLA violation. (*iii*) Building an online DRL model that selects dynamically a network slice as a trust participant, at each federated learning round. (*iv*) At each FL round, the trusted participant applies dimensionality reduction scheme and unsupervised machine/deep learning to detect the malicious participant (s). Before we proceed, we first give an overview of our framework in the next subsection.

5.3.1 Overview of TQFL Framework

As depicted in Fig. 5.1, we consider n running network slices that may be initiated by different vertical industries, such as intelligent transportation, Industrial IoT, and eHealth verticals. The running network slices are interconnected to an Inter-Domain Slice Manager (IDSM), which is in charge for the management and orchestration of network slices. To enable ZSM, the IDSM side includes an AE for building learning models and a DE, to make suitable decisions based on AE's outputs. On the other side, each running network slice is managed locally by a Domain Slice Manager (DSM), which also includes a MS for monitoring data and in-slice traffic, and an AE for building learning models.

5.3.2 Generation of Realistic Dataset

Machine/Deep learning algorithms require data to create learning models. However, more the datasets are realistic and large, more the deep learning models are accurate and adaptable for various situations. Hence, the first critical step toward developing accurate learning models is the data set collection (data acquisition or monitoring).

With the lack of real dataset, we conducted a real testbed using the Eurecom

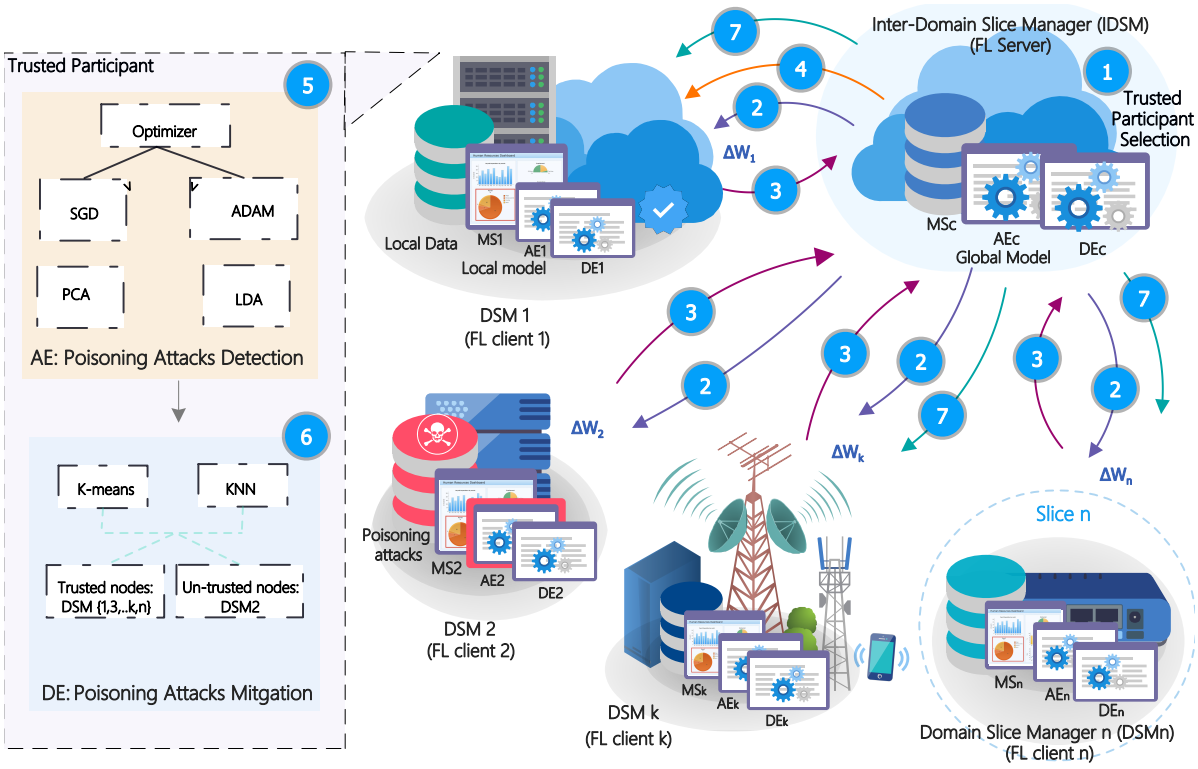


Fig. 5.1 Overview of TQFL Architecture.

OAI platform, to generate a realistic dataset, called EARCD for Eurecom AMF Resource Consumption Dataset. OAI implements 5G radio access and core networks, as open source software. We emulated ten instances of AMF, running as VNFs inside ten isolated network slices. The network slices differ from each other, in terms of their AMFs' configurations, for instance: the AMF of network slice 1 has 1GB of memory and 1 CPU, the AMF of network slice 2 has 2GB of memory and 2 CPUs, etc.

In addition, we used my5G-RANTester tool to emulate user equipments (UEs), one gNB and to generate attach requests, that will be then handled by the different network slices' AMFs. By increasing the number of UEs, we are able to generate up to 560 attach request per second, covering different traffic density.

Besides, it is clear that the latency values increase as the number of attach request increases. However, the latency values depend greatly to the configurations of network slices' AMFs. Fig. 5.2 depicts the average latency values according to the received number of attach requests, for two different configurations of AMFs' VNFs. We clearly observe that AMF with 2048 MByte of memory and 2 CPUs succeeds to decrease the latency, when compared to AMF with 1024 MByte and 1 CPU.

Therefore, ten local datasets (ten network slices) were generated, by varying the number of handled attach request/s, where each dataset contains 2813 samples

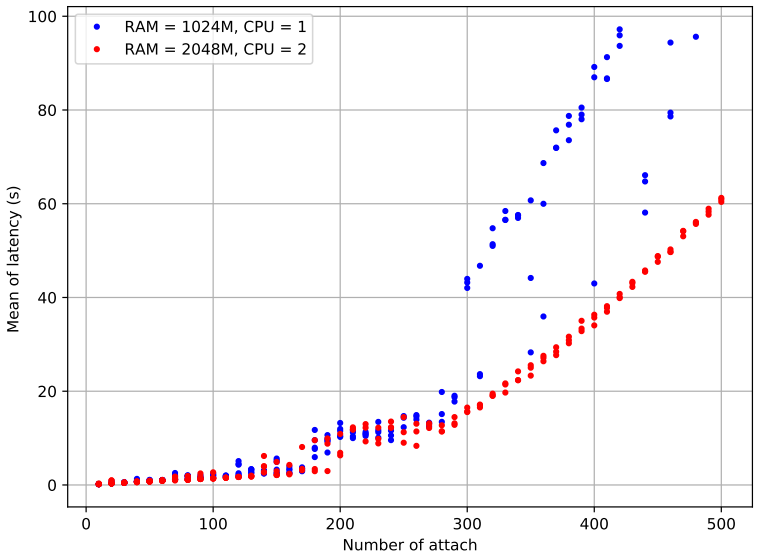


Fig. 5.2 Latency of multiple attach with different configurations of CPU and RAM.

(rows).

In addition, each local dataset presents five features, where the feature "*ram −limit*" presents the memory allocated to the container in megabytes, and the feature "*cpu −limit*" shows the CPU allocated to the container. The feature "*ram −usage*" presents the amount of memory used by the container at the time of the metrics collection in byte. And the feature "*cpu −usage*" presents the amount of CPU used by the container at the time of the metrics collection. Finally, the feature *n* shows the number of parallel registration requests sent to the AMF. Also, the dataset presents the label *mean* that represents the mean registration time for all the registration requests in microseconds.

5.3.3 Latency KPI Prediction in Federated Way

To enable zero touch management of the latency KPI related to the UEs attach requests, we built a deep learning model, that enables each domain slice manager (DSM) to predict the average needed latency for UEs attach requests, in federated way. As depicted in Fig. 5.1, we recall that each running network slice includes MS, AE, and DE, which allow it to monitor needed information (used CPU, used memory, etc.), build local learning models, and make the suitable decisions based on made predictions, respectively. Thus, we create a deep learning model in federated manner, that comprises four main steps:

1. **Data Pre-Processing**: it is important to preprocess the collected data, before feeding it to the deep learning model, since the output of this step may directly affect the performance of the learning model. This step consists to check if the data is on the same format and scale, does not include null and redundant values, and includes all needed features for the training step.

2. **Learning Initialization**: In this step, the Inter-Domain Slice Manager (IDSM), as the central node, creates an initial global model and defines the learning hyper-parameters, in terms of batch sizes, number of epochs, learning rate, neural network architecture, etc. These parameters are then sent to the network slices' DSMs, as clients. Noting that the network slices implement an artificial neural network (ANN), which comprises one input layer of 5 neurons (the five input features), seven hidden layers of 20 neurons, and one layer of 1 neuron (Latency value). In addition, the activation function of the neuron nodes is rectified with a linear function, and two different optimizers are used: SGD and ADAptive Moment estimation (ADAM) optimizers (see subsection 5.4.1 for more details).

3. **Training of Local Models**: each network slice's DSM, through its AE, updates parameters of its local model. Indeed, each DSM gathers needed data through its MS and splits it into many batches. Then, the DSM's AE performs the average gradient on each batch, with respect to the current model, during several epochs and with a particular learning rate. Once done, the network slices' DSMs send their local models back to the central IDSM.

4. **Building of Global model**: the central IDSM aggregates the DSMs' local models, using *FedAvg* algorithm proposed in [93], which is based on distributed SGD, as weights optimizer. The aggregated global model is then sent back to the DSM nodes.

5.3.4 Poisoning Attack Detection

In this section, we present our poisoning attack detection scheme, which begins with the election of a network slice participant as a trusted entity. Then, the latter will be in charge of detecting malicious clients, by leveraging unsupervised learning.

5.3.4.1 Trust Participant Selection using Deep Reinforcement learning

In each FL round, the central IDSM of running network slices selects a running network slice (DSM), as a trusted node. To do so, we design a new deep reinforcement learning-based model, to derive an optimal policy about trust node selection,

while considering several criteria related to such nodes, such as their reputation, detection rate of malicious nodes, and its accuracy in building learning models.
Deep Reinforcement learning is a process that enables one or set of agents to learn how to make suitable decisions, through error and trial, and based on their (agents) previous experiences. Specifically, each agent interacts with the environment to receive either penalties or rewards, for the actions it makes. Hence, the main objective of deep reinforcement learning is to derive an optimal policy about agents' actions, that maximizes agents' cumulative reward.
In our study, the central IDSM is the agent that interacts with running network slices' DSM (environment), in discrete time steps, as shown in Fig.5.1. In what follow, we first formalize our problem using Markov Decision Process (MDP). Then, we apply Deep Q-Network algorithm (DQN), to predict the best trusted participant to select, at each federated learning stage.

a. Markov Decision Process Model

The problem is often modeled using a MDP, where, at every timestamp t, the central IDSM manager, is in a state s_t, takes an action a_t. Then, it (IDSM) will receive a scalar reward from the network slices' DSM (environment). In addition, the system passes from the state s_t to a state s_{t+1}, according to environment dynamics $p(s_{t+1}|s_t, a_t)$. Therefore, the central IDSM manager attempts to learn a policy $\pi(a|s)$, that helps to map from observations to actions, and maximizing its rewards.
In our study, a state $s \in S$ is a three-sized tuple $(MC; TC; GMA)$, where:

- MC_i is the number of malicious models (participants), that are detected by network slice's DSM_i;
- TC_i is the number of trust models, that are detected by network slice's DSM_i;
- GMA_i is the accuracy of the global model, after detecting and removing malicious models, by network slice's DSM_i;

At every timestep t and when aiming to take an action a, the central IDSM can select a network slice i, among the n running slices, that has the highest reputation Rep. Then, the system may transit to a new state s_{t+1}, that corresponds to the number of detected malicious clients MC_i, the number of trust clients as well as the new accuracy of the new global model.
Furthermore, when the system moves to a new state s_{t+1}, a reward $r_{t+1} = R(s, a)$ is associated with the transition $p(s_{t+1}|s_t, a_t)$. For this end, we model the reward that the central IDSM expects to get from the selected network slices' DSM, as follows:

$$r_{t+1} = \begin{cases} \mathrm{GMA}_{t+1} & \text{if } MC_i >= 1 \text{ or } GMA_{t+1} >= GMA_t \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

It is clear that the received reward from a participant i affects directly its reputation in the federated learning process, and as the reward increases, as the reputation of the network slice i increases as follows:

$$Rep_i = Max(0, r_{t+1}) \tag{5.2}$$

Based on equation 5.1, the central IDSM's reward will be affected not only by the number of observed malicious participants, but also by the accuracy of the new global model. Therefore, the IDSM manager aims to derive the optimal policy in selecting a trust participant, which optimizes the accuracy of the learning model as well as its detection of poisoning attacks.

b. Deep Reinforcement Learning

As we mentioned before, our reinforcement scheme is based on DQN, which combines Q-learning and deep neural network, to learn an optimal policy from input data. Q-learning is a reinforcement learning algorithm that aims to identify the optimal policy of action-selection, maximizing the total reward, called Q-value, for any finite MDP. The Q-value of each action is calculated and stored in a table, named Q-table. In addition, at every timestamp (learning episode), the Q-values are updated using the following formula:

$$Q(s_t, a_t) = Q(s_t, a_t) + \Theta\left(r_{t+1} + \lambda \max_{a \in A} Q(s_{t+1}, a)\right) \tag{5.3}$$

With Θ is the learning rate and λ is the discount factor, indicating the future rewards importance.

Besides, the basic idea of DQN is to use neural network to predict the Q-value of all possible actions, based on current state. In particular,DQN comprises two neural networks: target network $Q'(s'; a; \theta')$ and a prediction network $Q(s; a; \theta)$. The prediction network is updated after each learning epoch, while the target network is directly updated from the prediction network, after every several iterations. Hence, DQN aims to reduce the loss function between both neural networks, as follows:

$$L = \left(r + \lambda \max_{a' \in A} Q'(s', a', \theta') - Q(s, a, \theta)\right)^2 \tag{5.4}$$

Where θ is the learning weights of the Q-network, that is updated using gradient

back propagation optimizer.

In our study, we implement a fully connected neural network. It comprises one input layer with three neurons, that correspond to the three states (***MC***, ***TC***, ***GMA***), two hidden layers with 24 neurons each, and one output layer to predict the Q-values of the running network slices. Noting that we tried several configurations, in terms of the number of intermediate layers and their number of neurons. We selected the best configuration that provided better performance.

5.3.4.2 Dimensionality Reduction of model updates

Once a trust client is selected, it will be in charge of detecting whether the received updates include a malicious model or not.
First of all, the selected trust client receives the n model updates of network slices from the central IDSM, and then applies a dimensionality reduction techniques to be able in presenting the model updates in 2D dimensions. Indeed, the dimension reduction is the transformation of dataset from a high-dimensional space into a low-dimensional space, in such a way, the low-dimensional representation retains the meaningful properties of the original data.
Fig. 5.3 shows a dimensionality reduction of our FL model updates in 3D, when applying linear discriminant analysis reduction technique. The different color of the points, which are defined by 3 axes (x:0, y:1, z:2), present the updates of different nodes (10 clients). However, as we can notice, the reduction to 3D will not provide a clear visualization, to interrupt and classify the model updates. That is why we decided to apply dimensionality reduction to 2dimensions (2D) (Fig. 5.4).
In our study, to design an efficient detection scheme, we are based on two different techniques: Principal Component Analysis (PCA) as unsupervised technique (ignores class labels), and Linear Discriminant Analysis (LDA) as a supervised technique.

※ Principal Component Analysis (PCA)

PCA is a statistical technique, which helps to explain data of high-dimensions, by extracting only some principal components of such data. The process of PCA comprises four main steps:

1. **Standardization of model updates' values**: the trust client receives n model updates, where each one contains five values that correspond to the five input features, including RAM capacity, CPU capacity, RAM used, CPU used, and number of attach request. The first step of standardization consists to put the feature values in the same range and format, in order to make sure that they will equally contribute to the final analysis, using the following

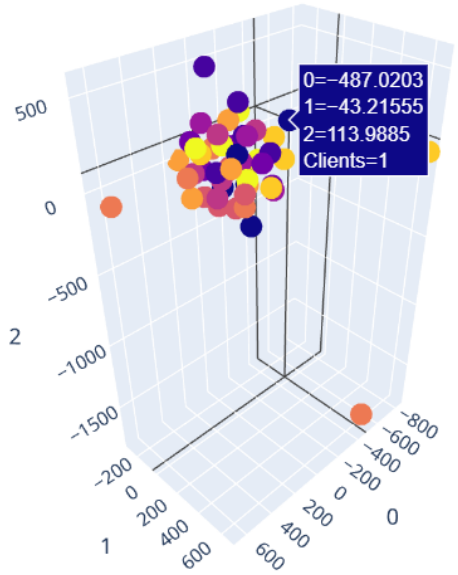


Fig. 5.3 LDA visualization with 3 dimensional applied on the nodes update.

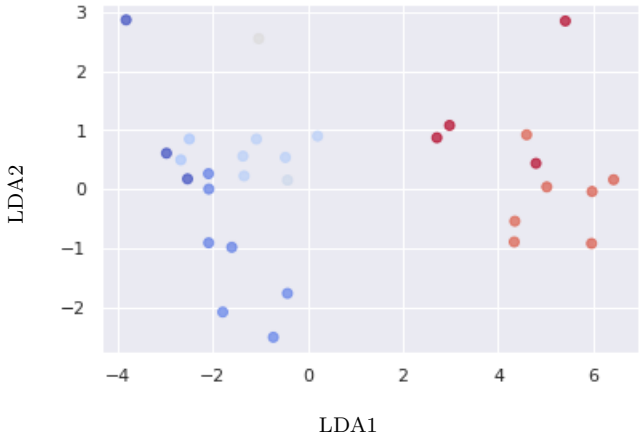


Fig. 5.4 LDA visualization with 2 dimensional applied on the nodes updates.

equation:

$$Str(value) = \frac{(value - mean)}{standarddeviation} \tag{5.5}$$

2. **Covariance Matrix**: The second step is the calculation of the "Covariance Matrix" between the input features of our dataset. Notably, the aim is to understand how these input features vary from the mean, with respect to each other. The covariance matrix is a $m \times m$ symmetric matrix (where m is the number of input features, 5 in our case). In fact, the covariance matrix describes the correlations between all the possible pairs of our input features, as follows:

$$\text{Cov(input features)} = \begin{pmatrix} Cov(x_1, x_1) & Cov(x_1, x_2) & \dots & Cov(x_1, x_m) \\ Cov(x_2, x_1) & Cov(x_2, x_2) & \dots & Cov(x_2, x_m) \\ Cov(x_3, x_1) & Cov(x_3, x_2) & \dots & Cov(x_3, x_m) \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ Cov(x_m, x_1) & Cov(x_m, x_2) & \dots & Cov(x_m, x_m) \end{pmatrix}$$

The correlation between two features depends mainly on the the covariance sign. If it is positive, both features increase or decrease together, which means that they are strongly correlated. Nevertheless, if it is negative, one increases whereas the other decreases, which signifies that they are inversely correlated.

3. **eigenvectors and eigenvalues of the covariance matrix**: eigenvectors and eigenvalues are the core of PCA, enabling us to identify the principal components. These parameters are new variables, that are constructed as mixtures or linear combinations of the initial features. Indeed, the eigenvectors (principal components) describe the directions of the new feature space, while the eigenvalues shows the variance of the data along the new feature axes. Both parameters are calculated using the following formula:

$$\begin{aligned} &Cov(input features) * v = \lambda * v \\ &Where: \\ &\quad \text{- } Cov(input\ features)\ is\ the\ covariance\ matrix \\ &\quad \text{- } v\ is\ the\ eigen\ vectors \\ &\quad \text{- } \lambda\ is\ the\ eigen\ value \end{aligned} \tag{5.6}$$

Therefore, eigenvectors and eigenvalues enable to create new g-dimensional data that gives g principal components.

4. **Computation of feature vector**: This step aims to select which principal components to keep and which ones to remove (those have lesser significance or lower eigenvalues). The output of this step is a matrix of vectors named "Feature vector", that has the selected components as columns. Finally, the initial data are aligned with the new principal component, using equation 5.7. In addition, the feature vector with EigenVectors are used to align/reorient the data from original axes to the new principal component axes.

$$FinalData = FeatureVector^T * StandardizedData^T \tag{5.7}$$

※ **Linear Discriminant Analysis (LDA)**

LDA is also a dimensionality reduction technique that aims to find not only the component axes, maximizing data variance (PCA), but also a feature subspace that maximizes class separability. Thus, LDA provides the ability to project a feature space (a dataset with nb dimensional samples), into a smaller subspace k, while maintaining the class-discriminatory information.

The LDA process, also, comprises different steps. First, we consider a matrix of n classes of model updates, that correspond to the n clients (network slices), as follows:

$$\mathrm{C} = \begin{pmatrix} C_1 \\ C_2 \\ \dots \\ C_n \end{pmatrix}$$

And each class comprises five input features, as follows:

$$F = \begin{pmatrix} F(1,c_1) & F(2,c_1) & \dots & F(5,c_1) \\ F(1,c_2) & F(2,c_2) & \dots & F(5,c_2) \\ F(1,c_3) & F(2,c_3) & \dots & F(5,c_3) \\ \dots & \dots & \dots & \dots \\ F(1,c_n) & F(2,c_n) & \dots & F(5,c_n) \end{pmatrix}$$

1. **Computing the d-dimensional mean vectors:** The first step of LDA is to compute a d-dimensional mean vector $\mathrm{M}(C_i)$ for the different classes n.

2. **Compute the Scatter matrix (in between class and within the class scatter matrix):** The second step is to compute the "Scatter Matrices" which are equivalent to the variance. In fact, we have two matrices of E x N dimensions to calculate: "The within-class" and the "between-class scatter matrix". The objective of these matrices is to determine the variability within a class (Intra class scatter) as well as between different classes (inter class Scatter). Both consist to calculate the distance between different points(inter/intra classes). The first matrix (within-class scatter) Sca is calculated as follows:

$$
\begin{aligned}
&S\,ca = \sum_{j=1}^{n} S\,s_j \\
&Where: \\
&S\,s_j = \sum_{x \in Di}^{(number\_of\_samples)} (x - m_i)(x - m_i)^T, \\
&S\,s_j \text{ is a scatter matrix for every class,} \\
&m \text{ is the mean vector,} \\
&m_i = \frac{1}{n_i} \sum_{x \in Di}^{(number\_of\_samples)} x_k
\end{aligned}
\tag{5.8}
$$

Likewise, the class-covariance matrices is computed by adding the scaling factor $\frac{1}{N-1}$ to the within-class scatter matrix, so that the equation 5.8 becomes:

$$
\begin{aligned}
&\xi_i = \frac{1}{N-1} \sum_{x \in Di}^{(number\_of\_samples)} (x - m_i)(x - m_i)^T \\
&S\,c_j = \sum_{i=1}^{n} (N_i - 1)\xi_i
\end{aligned}
\tag{5.9}
$$

The between-class scatter matrix SB is computed by the following equation:

$$
S_B = \sum_{i=1}^{n} N_i (m_i - m)(m_i - m)^T \tag{5.10}
$$

where m is the overall mean, and m_i and N_i are the sample mean and sizes of the respective classes.

3. **Computation of the eigenvectors and eigenvalues for the scatter matrices:** After that, the next step is the computation of the eigenvectors and corresponding eigenvalues for the scatter matrices, as we did for PCA.

4. **Selecting linear discriminant for the new feature subspace:** Following, the sort of eigenvectors should be applied by decreasing eigenvalues and choose k eigenvectors with the largest eigenvalues to form a $d \times k$ dimensional matrix (where every column represents an eigenvector).

5. **Transforming the samples onto the new subspace:** Next, the selected $d \times k$ eigenvector matrix will be used to transform the samples onto the new subspace dimension.

5.3.5 Poisoning Attacks Mitigation

$$\Delta w^2 = DR_{(d:2)}(\Delta w^n) \tag{5.11}$$

Once applying Dimensionality Reduction (DR) techniques and depicting network slices' updates (Weight matrix: Δw^n) in a 2d plan Δw^2 (see eq.5.11), the last step consists of grouping the received updates into several clusters, in order to determine malicious updates/models. Actually, we apply the clustering algorithms in order to classify the FL nodes into 2 different clusters/groups: the first group present the trusted nodes, which present similar updates to those of the node selected by DRL, and the second group present the un-trsuted nodes, which present dissimilar updates to those of the node selected by DRL. Then the FL server (IDSM) will continue the training using only the list of trusted nodes. Algorithm 4 describes the main instructions performed by the central IDSM.

Algorithm 4 TQFL Framework.

Require: Iterations k, Network Slices N, FL Round R, List of FL nodes L_{Nodes}, Dimensions of the updates d, Reputation of FL Nodes $Rep(N)$.
Ensure: Aggregated Model M^{i+1}.
The IDSM selects the trusted node:
$Trusted\_Node = DQL\ (Rep(N \in L_{Nodes}))$
$r = 0$,
The IDSM create the initial model:
Initialize M_0
$i = 1$
while $i \leq R$ **do**
  $j = |1|$
  while $j \leq |N|$ **do**
    $R_j^{i+1} \leftarrow DSMUpdate(j, R^i)$
    $Send\ R_j^{i+1}\ to\ Trusted\_Node$
    The Trusted_Node :
    $\{R_j{}^{i+1}\}^{d=2} = Dimesinalty\ Reduction(\{R_j{}^{i+1}\}^{d=n})$
    $L_{Trusted\_Nodes} = Clustering\ Algorithm(\{R_j{}^{i+1}\}^{d=2})$
    $L_{Malicious\_Nodes} = Clustering\ Algorithm(\{R_j{}^{i+1}\}^{d=2})$
  end while
  $M^{i+1} \leftarrow \frac{1}{|L_{Trusted\_Nodes}|} \sum_{t=1}^{t=|L_{Trusted\_Nodes}|} R_t^{i+1}$
end while
Return M^{i+1} to Network slices' DSMs.

To ensure an effective detection of malicious updates, we chose to apply two

different clustering algorithms (unsupervised and supervised). The first is k-means which is an unsupervised learning algorithm. It considers no labelled update models' data. The second is k-nearest neighbors (KNN), a supervised learning algorithm. It considers labelled data about model updates. Both algorithms aim to divide the received update models, at each FL round, into k clusters, that share similarities and are dissimilar to the model updates belonging to another cluster. We note that we leverage the model update of the trusted participant as a reference that helps us to make the difference between malicious and trust models.

5.4 Performance Evaluation

In this section, we evaluate the performance of our TQFL framework in order to validate it.

5.4.1 Experiment Setting

We developed the main components of our framework, in terms of FL-based latency model, DQN-based participants selection model, clustering and dimensionality reduction-based attack detection model, using the Tensorflow Python library and leveraging our EARCD dataset. We evaluate our FL-based model to predict the latency KPI, in terms of Mean Squared Error (MSE) on top of two different weight optimizers (SGD and ADAM), in order to determine a suitable optimizer that improves the prediction accuracy of our both FL-based and attack detection models.

We note that we evaluate our FL-based model with and without the presence of malicious nodes to show the impact of poisoning attacks on the performance of our FL model. The malicious nodes (network slices) generate poisoning attacks by introducing new malicious/incorrect data samples and build their local models on top of such data. For example, the malicious nodes may consider high latency values, even when receiving a low number of attach requests and vice versa.

In addition, we trained our DQN model during more than 5000 learning episodes. Once converged, we deployed our DQN-based model at the inter-domain slice manager, which is in charge to selecting a trusted participant at each FL round. Moreover, to evaluate our attack detection scheme, after a given number of FL training rounds (r < R'), the ten network slices' DSMs send their model updates to the inter-domain network slice manager. The latter first selects one network slice as a trusted party before sharing with it the model updates. Table. 5.2 gives more details about the parameter settings used in our simulation.

Table 5.2 The parameter settings.

The parameter settings	**Value**
Federated Learning (FL)	
Number of layers	4
Number of neurons	20
Optimizer 1	SGD
Learning rate	0.0001
Optimizer 2	ADAM
Activation function	ReLU
Loss function	MSE
Reinforcement Learning (DQN)	
Number of layers	2
Number of neurons	24
Optimizer	ADAM
Learning rate	1e-2
episodes	5000
Dimensional Reduction Algorithm (DRA)	
Dimension of LDA	2D
Dimension of PCA	2D
Clustering Algorithm (CA)	
Number of cluster (k)	1, 2

5.4.2 Evaluation of Latency prediction in Federated way

Following, we present the results without malicious nodes, for two different optimizer: ADAM and SGD.

Fig. 5.5 depicts the MSE metric of our FL model during several FL rounds, with and without presence of malicious network slices. On top of the ADAM optimizer, we clearly observe that MSE of our model without malicious nodes is lower than with malicious nodes, around 0.1035681. However, it increases as we add more malicious nodes, that will inject more malicious/incorrect data in terms of latency KPI. Therefore, poisoning data attack affects highly not only the performance of our FL model in terms of MSE, but also the model convergence towards an accurate global model.

Similarly, Fig. 5.6 shows the MSE metric of our FL model on top of the SGD optimizer. We also see that the MSE increases as the number of malicious network slices is increased as well, and the FL model without malicious nodes outperforms the other models (with malicious nodes), with a MSE of 0.173584. Hence, these results confirm the results of Fig. 5.5, and show that data poisoning attack can have a negative impact on the performance of our FL model, especially in terms of global model convergence, whatever the used learning optimizer (ADAM or SGD).

Table 5.3 The Selection results.

$t:t1$									
RepDSM1	RepDSM2	RepDSM3	**RepDSM4**	RepDSM5	RepDSM6	RepDSM7	RepDSM8	RepDSM9	RepDSM10
0.2	0.4	0.3	**0.7**	0.61	0.48	0.37	0.68	0.54	0.62
Selected DSM			✓						
$t:t2$									
RepDSM1	RepDSM2	RepDSM3	RepDSM4	RepDSM5	RepDSM6	RepDSM7	**RepDSM8**	RepDSM9	RepDSM10
0.1	0.22	0.15	0.74	0.58	0.52	0.31	**0.8**	0.60	0.71
Selected DSM							✓		

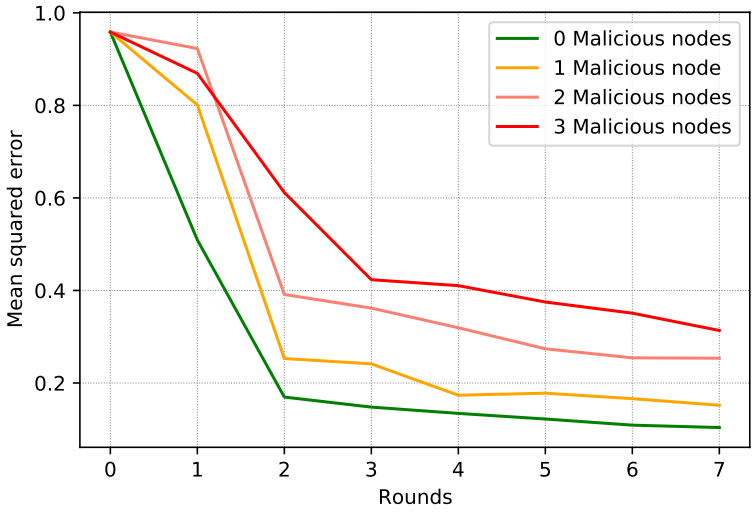


Fig. 5.5 Mean Squared error of our FL model when using ADAM Optimizer.

5.4.3 Evaluation of Trust Participant Selection

TABLE. 5.3 shows the results of the selected DSM based on our DQN-based scheme, for two different time instances $t1$, and $t2$. These results are obtained based on the nodes' reputations ($Rep_i, i \in [1, 10]$). As we observe, when $t = t1$, the $DSM4$ was selected as trust node, since it maximizes the reputation value. Similarly, when $t = t2$, the network slices' DSMs have different reputation values, however, $DSM8$ is selected, because it presents the highest value. In other words, at each FL round, our DQN-based algorithm enables to select the network slice maximizing the reward, since the reputation value is determined based on the reward using equation 5.2.

5.4.4 Evaluation of Data Poisoning Attack Detection

After showing the negative effect of the poisoning attacks on the performance of the global FL model, in this subsection, we evaluate the performance of our combined dimensionality reduction and unsupervised clustering scheme against data poisoning attacks, and on top of the two different optimizers: ADAM and SGD. Noting that for data poisoning attacks, the malicious node tries to inject incorrect latency values, e.g. high latency value, even when the node has high resource capacity, in terms of memory and computing.

Combining LDA with k-means Fig. 5.7 and Fig. 5.9 depict the detection results when combining LDA and K-means, on top of ADAM and SGD optimizers, respectively. We also vary the percentage of malicious network slices' DSM and

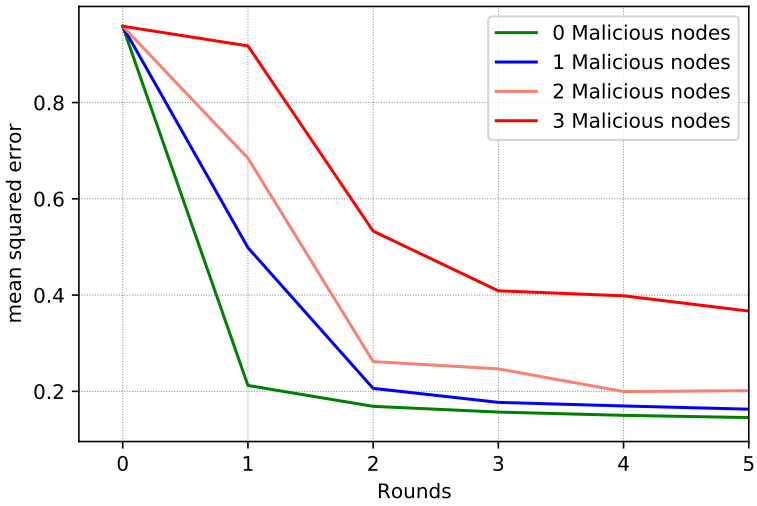
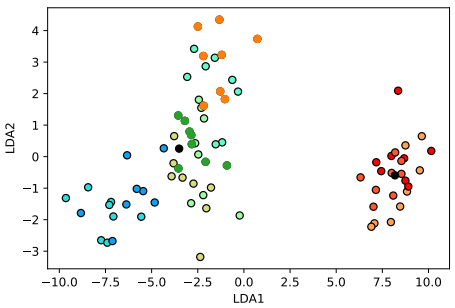
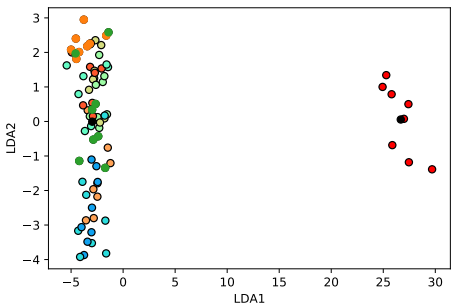


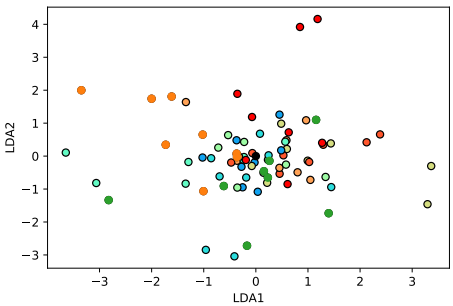
Fig. 5.6 Mean Squared error of our FL model when using SGD Optimizer.



a: Nb of malicious nodes = 3.

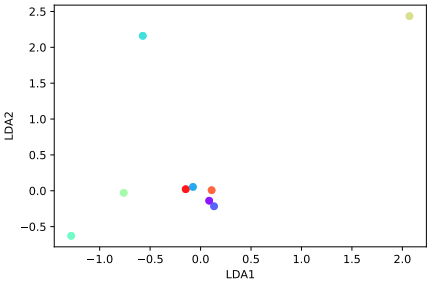


b: Nb of malicious nodes = 1.

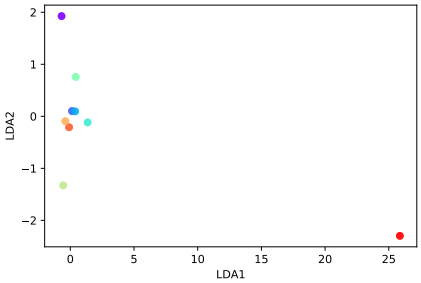


c: Nb of malicious nodes = 0.

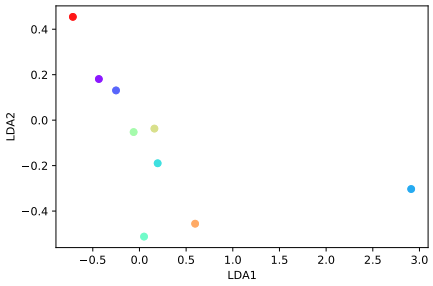
Fig. 5.7 LDA + K-means for different number of malicious nodes (ADAM optimizer).



a: Nb of malicious nodes = 3.

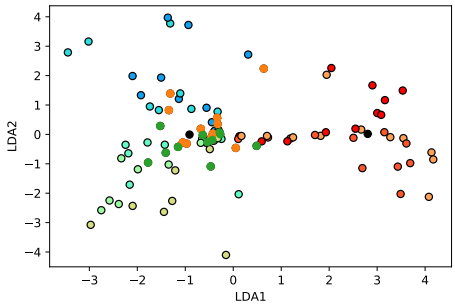


b: Nb of malicious nodes = 1.

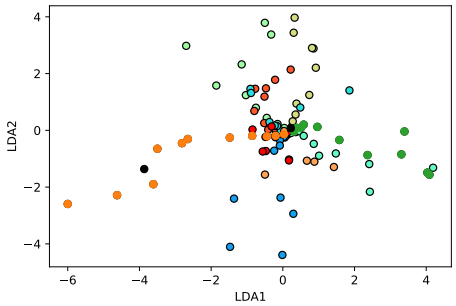


c: Nb of malicious nodes = 0.

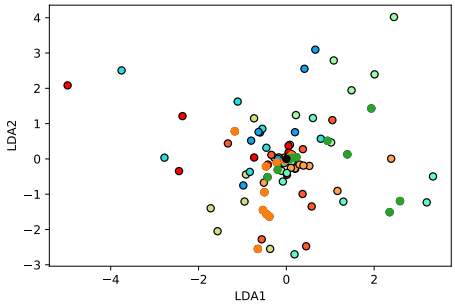
Fig. 5.8 LDA + KNN for different number of malicious nodes (ADAM optimizer).



a: Nb of malicious nodes = 3.

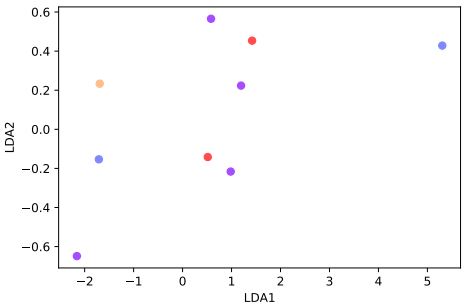


b: Nb of malicious nodes = 1.

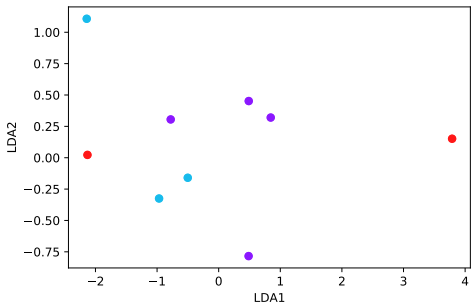


c: Nb of malicious nodes = 0.

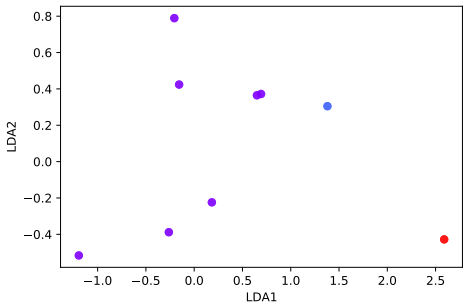
Fig. 5.9 LDA + K-means for different number of malicious nodes (SGD optimizer).



a: Nb of malicious nodes = 3.

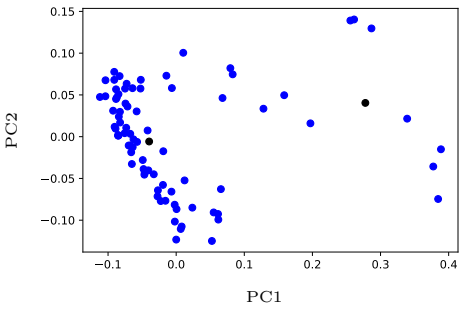


b: Nb of malicious nodes = 1.

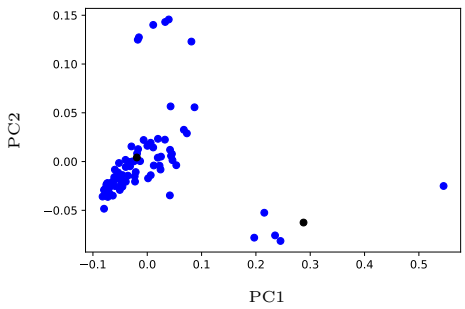


c: Nb of malicious nodes = 0.

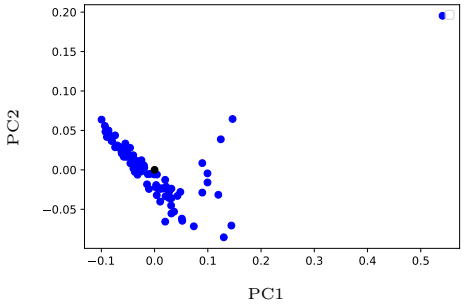
Fig. 5.10 LDA + KNN for different number of malicious nodes (SGD optimizer).



a: Nb of malicious nodes = 3.

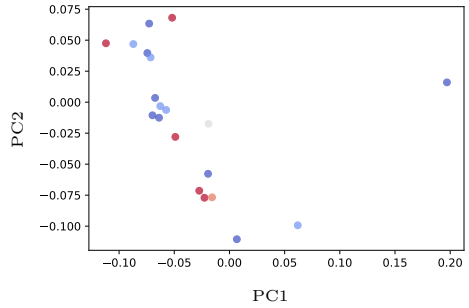


b: Nb of malicious nodes = 1.

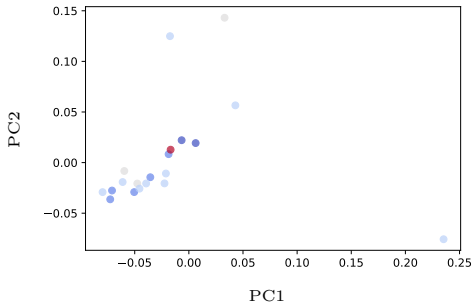


c: Nb of malicious nodes = 0

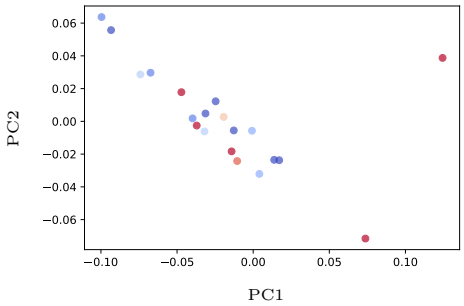
Fig. 5.11 PCA + k-means for different number of malicious nodes (ADAM optimizer).



a: Nb of malicious nodes = 3.

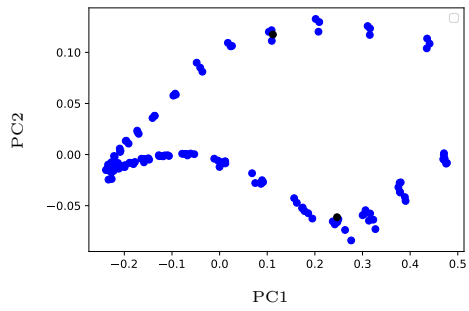


b: Nb of malicious nodes = 1.

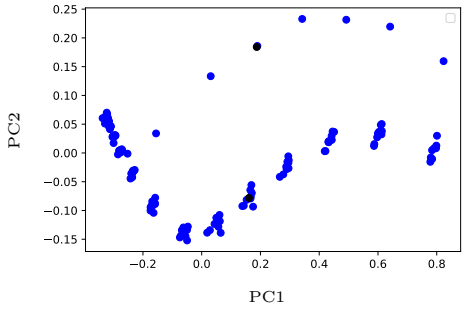


c: Nb of malicious nodes = 0.

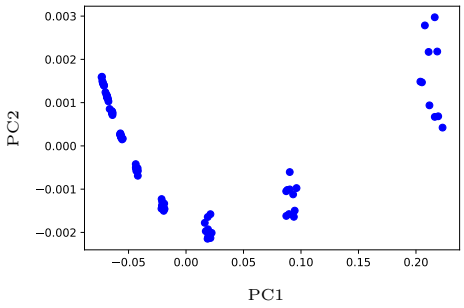
Fig. 5.12 PCA + KNN for different number of malicious nodes (ADAM optimizer).



a: Nb of malicious nodes = 3.

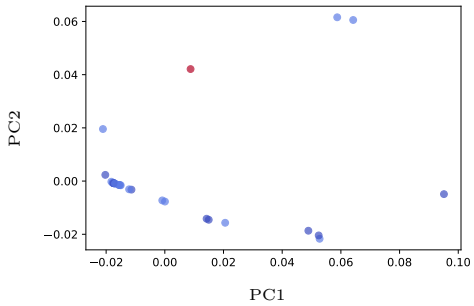


b: Nb of malicious nodes = 1.

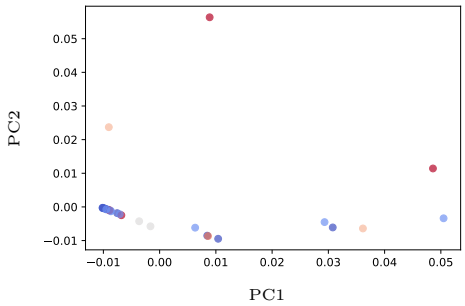


c: Nb of malicious nodes = 0.

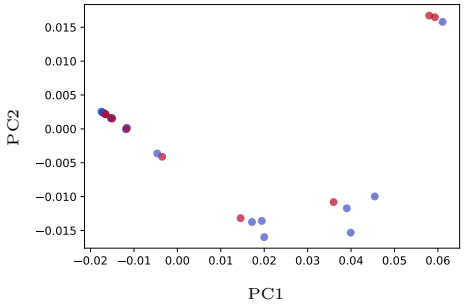
Fig. 5.13 PCA + k-means for different number of malicious nodes (SGD optimizer).



a: Nb of malicious nodes = 3.



b: Nb of malicious nodes = 1.



c: Nb of malicious nodes = 0.

Fig. 5.14 PCA + kNN for different number of malicious nodes (SGD optimizer).

show the trusted nodes that are selected at each FL round (nodes in green colour). As we see, our scheme can clearly detect the malicious nodes, even with only one of malicious node. Specifically, the trusted node applies both LDA and K-means, and then all nodes that are in the same cluster with it, are considered as correct updates, while the nodes (models) that are in the other (s) cluster (s) will be considered as malicious. Therefore, our trust participant selection algorithm helps us not only to select a trusted node, but also to determine malicious nodes when performing the dimensionality reduction and unsupervised clustering. Moreover, determining the trusted cluster of nodes will also help the FL server (IDSM) to select a trusted participant for the next FL round.

Combining LDA with KNN Fig. 5.8 and Fig. 5.10 also show the clustering of local models when applying both LDA and KNN, on top of ADAM and SGD optimizers, respectively. Whatever the number of malicious nodes, we also observe that there are always some isolated points that represent the infected models sent by the malicious DSMs. However, for the LDA technique, we see that the isolated models (infected) are identified better with the ADAM optimizer than with the SGD optimizer (Figs. 5.7 and 5.8). Hence, the LDA (with KNN or k-means) technique give better detection on top of the ADAM optimizer. In fact, these last combinations show the clearest clustering (two separate groups) compared to

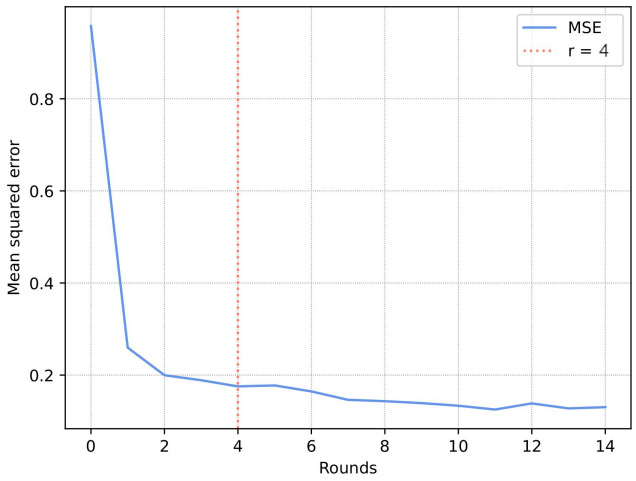
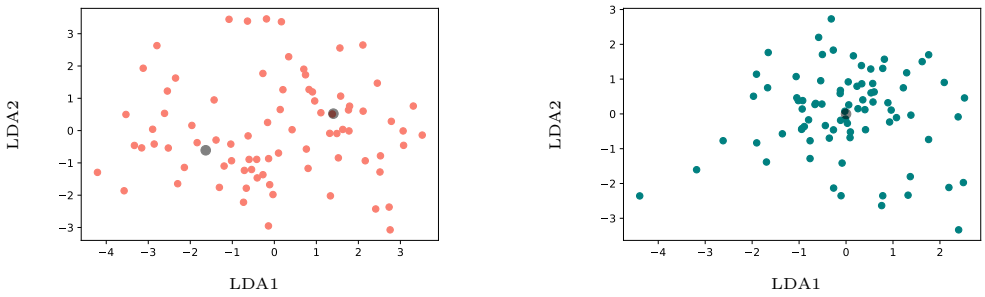


Fig. 5.15 Mean Squared error of the FL global model (ADAM optimizer).

other algorithms.

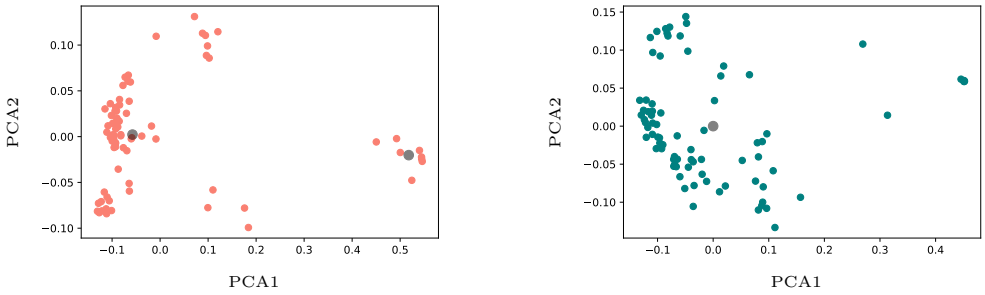
Combining PCA with K-means As we did for LDA, we also evaluated the performance of PCA technique when combined with clustering unsupervised algorithms. Figs. 5.11 and 5.13 show the clustering detection when combining PCA with K-means, on top of the ADAM and SGD optimizers, respectively. We remark that both optimizers succeed in separating and identifying infected models by incorrect data. However, infected models are better identified on top of the SGD optimizer, as compared to the ADAM optimizer. Thus, PCA with K-means gives better performance in detecting infected models on top of the SGD optimizer. Indeed, this last combination shows the clearest clustering (two separate groups) compared to other algorithms.

Combining PCA with KNN Similarly, Figs. 5.12 and 5.14 depict the detection when combining PCA with KNN on top of the ADAM and SGD optimizers, respectively. As in Figs. 5.11 and 5.13, PCA with KNN on top of both optimizers clearly separate correct local models from infected ones and thus enables to detect/identify malicious DSMs. We also see that infected models are better identified when leveraging the SGD optimizer than the ADAM optimizer. Therefore, the PCA technique with either K-means or KNN gives better detection of malicious models on top of the SGD optimizer, which is confirmed in Figs. 5.11, 5.13, 5.12, 5.14. In fact, these last combinations show the clearest clustering (two separate groups) compared to other algorithms.



a) With the malicious nodes. b) Without the malicious nodes.

Fig. 5.16 LDA + K-means on top of the ADAM optimizer.



a) With the malicious nodes. b) Without the malicious nodes.

Fig. 5.17 PCA + K-means on top of the SGD optimizer.

Impact of malicious nodes detection on the FL accuracy In Fig. 5.5 and 5.6, we showed that poisoning data attack affects highly not only the performance of our FL model in terms of MSE, but also the model convergence towards an accurate global model. Moreover, the observation confirms that even a single malicious node can influence the global model accuracy as well as open the possibility of conducting two types of attack. The first one is a Byzantine attack, in which the malicious node aims to prevent the model from converging. Whereas the second is the poisoning attack in which the attacker intends to make the FL global model misinterpreting some of the inputs in its future use.

Fig. 5.15 depicts the MSE metric of our global FL model, when applying our combined dimensionality reduction and clustering scheme, to detect and remove infected models. Noting that we apply our scheme after a given number of FL rounds $r = R' = 4$, i.e. after collecting some updates from the network slices' DSMs. We clearly observe that our scheme improves MSE of the global FL model, which decreases as we increase the number of FL rounds, even when injecting some infected models from the fourth round. Therefore, our detection scheme enables not only to identify malicious DSMs, but also improving the accuracy of the global FL model.

5.4.5 Evaluation of Model Poisoning Attack Detection

In this subsection, we evaluate our scheme against model poisoning attacks, where malicious DSMs try to send incorrect learning models, which are generated randomly for instance. Fig. 5.16 shows the results when combining LDA and K-means on top of the ADAM optimizer. We remark that LDA with K-means does not clearly succeed to identify the malicious models Fig. 5.16-b. However, when combining PCA and K-means on top of the SGD optimizer, the malicious models are better detected, as depicted in Fig. 5.17-b. Therefore, we can deduce that model poisoning attacks are better detected and identified, when using PCA with K-means techniques, as compared to LDA and K-means combination.

In general, we can deduce that our scheme succeeds to provide stable performance in dealing with both data and model poisoning attacks that can target federated learning-based models. In particular, combining dimensionality reduction and clustering unsupervised learning helps us to not only detect/identify infected learning models, but also to improve the global accuracy of the FL model.

5.5 Conclusion

In this chapter, we designed a novel secure federated learning framework, which leverages reinforcement deep learning, dimensionality reduction, and clustering unsupervised learning to deal with both data and model poisoning attacks that can target federated learning-based models, in 5G and beyond networks. First of all, we generate a realistic dataset related to network slicing KPIs, in order to build a federated learning model for latency KPI prediction. We then generate realistic data and model poisoning attacks on top of our federated learning-based model.

To deal with them, a dynamic selection algorithm of a trusted node was first proposed, leveraging reinforcement deep learning. The latter is in charge of detecting and identifying malicious learning models, and hence network slices. The numerical results show the efficiency of our framework in dealing with model/data poisoning attacks, while preventing the Byzantine attacks, and thus in mitigating such attacks and ensuring a stable performance of the federated learning model.

CHAPTER 6

A Trust and Explainable Federated Deep Learning Framework in Zero Touch B5G Networks

6.1 Introduction

Beyond 5G networks (B5G), or so-called "6G", are considered as key enabler of a wide range of new pervasive services related to different vertical industries, including eHealth, automotive, energy, and manufacturing [94]. This is possible by the support of massive number of coexisting network slices with different requirements and functionality. However, such new services and paradigms introduce more issues on the management and orchestration of the massive network slices, during their life-cycles, that traditional network infrastructures fail to cope with. To achieve this vision, ZSM calls for a large usage of advanced deep learning algorithms, in order to dynamically build efficient decisions at different levels, including radio resource allocation, energy-efficiency management, computing/memory resource provisioning, etc. In this context, FL proved their efficiency in building collaborative deep learning models, among several network slices. FL allows running network slices to create deep learning models, without sharing their data, and thus ensuring the privacy and isolation of such network slices. Indeed, FL-based solutions give "machine-centric" decisions about running network slices and their performance, which will be then executed/applied by managers, i.e., slice manager staff/module.FL-enabled solutions do not provide any details about why and how such decisions were made, and thus such decisions cannot be properly trusted/understood by slice managers.

It is a very critical decision, because the FL model is collaboratively trained in disturbed nodes, and without access and verification to their local dataset. In other words, the main issue of FL-based mechanisms are the black-box decisions, whose internal functioning of the FL model is not understood and hidden. To alleviate this issue, we leverage XAI paradigm that aims to improve the transparency of black-box FL decision-making process. In particular, XAI helps to explain the FL-based decisions to make them interpretable/trustable by network slices managers.

In this chapter, we design a novel XAI-powered framework to explain FL-based decisions. Our framework studies the use of linear and non-linear XAI techniques, to identify the most informative features and investigate their impact on the final FL model predictions.

Therefore, we first build a deep learning model in federated way, to predict KPI of network slices. Specifically, our FL-based model predicts the latency KPI of network slices, in order to help in anticipating any violation of SLA, related to the latency KPI. Then, we develop several XAI models on the top of our FL-based model, such as SHAP, LIME, RuleFit, and PDP to enhance the level of trustiness (in the local data/model and the FL global model), transparency, and explana-

tion of the FL-based decisions, to different B5G network stakeholders, such as slice managers. Noting that our framework leverages a real dataset about latency KPI that, we generate using OpenAirInterface platform.
The rest of this chapter is organized as follows. In section 6.2, we present a review of related works. Section 6.3 describes the design and specification of the proposed framework. In Section 6.4, we evaluate and validate our XAI-empowered FL framework. Finally, section 6.5 concludes the chapter.

6.2 Related work

In this section, we present the few works, that addressed the explainability of FL-based models. In [95], the authors proposed a novel scheme to interpret FL-based models. They leveraged shapley value to compromise the data-privacy protection and model explainability in FL. This scheme enables each FL participant to get feature importance for its own features (data), and a unified feature importance for the features of the other participants.
In [96], the author proposed an explainable horizontal federated learning approach, leveraging integrated gradients explainability method. The authors compared between centralized and federated models, and applied integrated gradients to show the score of each feature related to each prediction for both models.
We observe that few studies have been proposed to deal with the explanation issue of FL-based models. These studies addressed this issue either in general way, i.e. whatever the use case [95], or focusing on a specific and simple use case, such as Taxi travel time prediction [96], which is very simple and different from B5G-enabled use cases.
In this chapter, we designed a novel framework that leverages RuleFit, LIME, SHAPE and PDP as XAI approaches, to explain and interpret an FL-based model of latency prediction. We note that our framework enables not only to deduce the most relevant features conducting to each FL-based latency prediction, but also providing both local and global explanations related to latency predictions of running network slices.

6.3 Proposed architecture of the XAI-Empowered FL Framework

In this section, we present our XAI-empowered FL framework for the B5G networks. First, we give an overview about the architecture of our proposed FL framework. Then, we describe our deep neural architecture we build to predict the latency related to the B5G networks, and our XAI-FL approaches we applied to interpret and explain the outputs of our deep learning model, trained on local

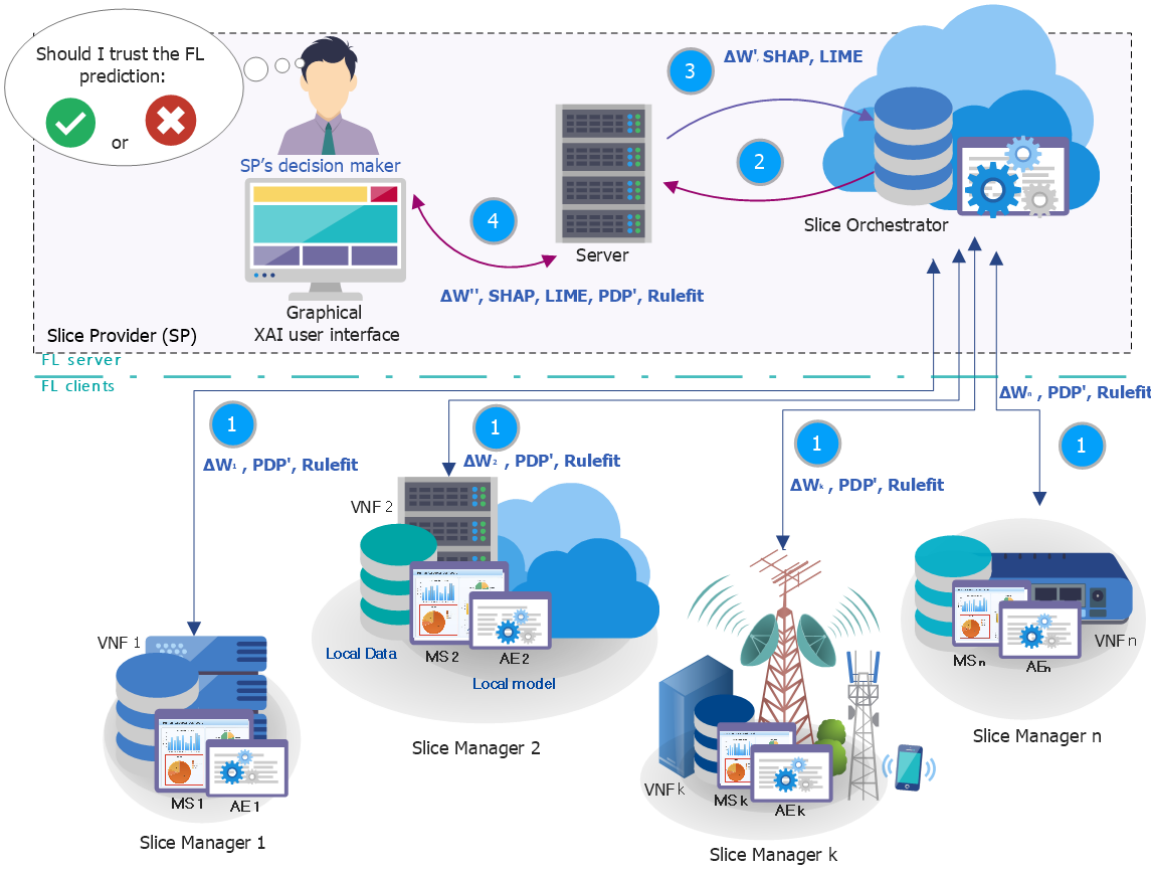


Fig. 6.1 The proposed trust B5G Architecture of our XAI-Empowered FL Framework.

data in disturbed nodes, during the FL process.

6.3.1 System overview Architecture

Fig. 6.1 gives an overview about the architecture related to our FL framework. Our architecture presents different 5GB parties, including the slice provider, the slice managers, and inter-slice manager (orchestrator), that collaboratively train a deep learning model, in federated way. First of all, the slice orchestrator initiates the federated learning process to build a learning model about latency KPI prediction. It defines the learning parameters such as, learning rate, neural architecture (number of layers/neurons), activation functions, and neural weights optimizer, in addition to the needed data.

The orchestrator then sends such information to the involved slice managers in the FL process (step 1 in Fig. 6.1). After that, the slice managers start to build their local learning models, using their own data. Once done, the slice managers send their local models to the slice orchestrator for aggregation Δw_i, in order to generate a global model (steps 1 and 2 in Fig. 6.1). The latter will be deployed at the slice managers level, to predict the latency KPI in real time (steps 3 and 1 in Fig. 6.1).

In addition, we apply four different XAI approaches, to generate local, global, and feature importance-based explanations, related to the latency predictions. Thus, our framework allows the slice provider's decision makers to not only monitoring the latency KPI of running network slices, but also how and why predictions are made, through a graphical XAI user interface (step 4 in Fig 6.1).
It is worth noting that our study is mainly based on a realistic B5G dataset, that we generate using OpenAirInterface, implementing the network slicing concept, called EARCD (Eurecom AMF Resource Consumption Dataset). Our dataset provides the response time (latency) of the Access Mobility Function (AMF), running as virtual network functions (VNF) inside the network slices, to handle user attach requests, while considering various parameters, including available and consumed CPU and RAM memory resources. Therefore, we consider ten running network slices and we generate a sub-dataset for each network slice of 2813 samples (rows). Each sub-dataset comprises five input features (CPU capacity, RAM capacity, used CPU, used RAM, and number of attach request), and one output feature, which corresponds to latency in terms of average duration to process users attachments, by the network slices' AMFs.
Noting that we also leverage 5G UE emulator, my5G-RANTester, to emulate multiple number of users that send a high number of attach request packets, to the ten network slices (AMF functions). Furthermore, each slice manager implements a neural network of one input layer of five neurons (our five input features), four connected hidden layers of 20 neuron nodes, and one output layer of one neuron (latency prediction). The activation function in all layers is the rectified linear activation function.

6.3.2 Explainable FL-based models for B5G networks

Several ML/DL explainable approaches were proposed to show the features impact, on the target labels. In order to interpret "Black-Box" of our FL-based model, we apply model agnostic approaches [24], which aim to understand the inner working of learning models. The model agnostic methods can be divided into two main categories: local model-agnostic and global model-agnostic. In what follow, we present the model-agnostic approaches, that we apply to explain our FL model.

6.3.2.1 Global Model-Agnostic Methods

The Global methods aim interpret learning model working in general way. We apply Partial Dependence Plots (PDP) method [24].

Partial Dependence Plot (PDP)

PDP method shows which input features of the dataset will highly impact the predictions of the learning model. In particular, PDP shows the linear relationship between features and target label (s) [97]. It means that if one features goes up, the target label will go up (or down) too. Thus, A PDP enables to determine whether the relationship type between the target label and each feature is linear, monotonic or more complex. The PDP plots visualize the average partial relationship between the predicted target and one or more features. In our case, the FL clients will only share a partial information of PDP, which are the variation percentage, in order to save their data privacy.

6.3.2.2 Local Model-Agnostic Methods

The local interpretation methods explain individual predictions. Therefore, the predictions of a single instance is described as the sum of feature effects. In this context, we apply three main methods: SHAP, LIME, and RuleFit.

a. SHapley Additive exPlanations (SHAP)

It is based on game theoretically optimal Shapley values, to interpret the outputs of any machine-learning models [24]. SHAP consists to calculate the contribution/impact of each feature to the final ML/DL model output. Specifically, The SHAP approach calculates shapley values from coalitional game theory, where a player, in a coalition, corresponds to a feature value of the data instance. Shapley values show how to fairly distribute the "output" (prediction) among the features. Noting that a player can be a set of features, such as pixels of an image, or one feature value, such as for tabular data. In our study, we apply SHAP method by calculating the shapley values of each individual prediction, where each individual feature of our dataset corresponds to a player in the coalition.

b. Local Interpretable Model-agnostic Explanations (LIME)

LIME method consists to train interpretable models to approximate the predictions of the ML/DL model. In particular, LIME creates a new dataset containing perturbed samples and their predictions of the studied ML/DL model. Then, an interpretable model is trained using the new dataset, which is weighted by the proximity of the sampled instances to the instance of interest. However, the new interpretable model will approximate accurately the ML/DL model predictions locally (local fidelity), and cannot provide a good global approximation. In our case, we first create a perturbed data instances of our EARCD dataset, then we build an interpretable model of our local/individual predictions of the latency KPI.

c. Rulefit Method

The RuleFit algorithm is also another explainable solution to understand the re-

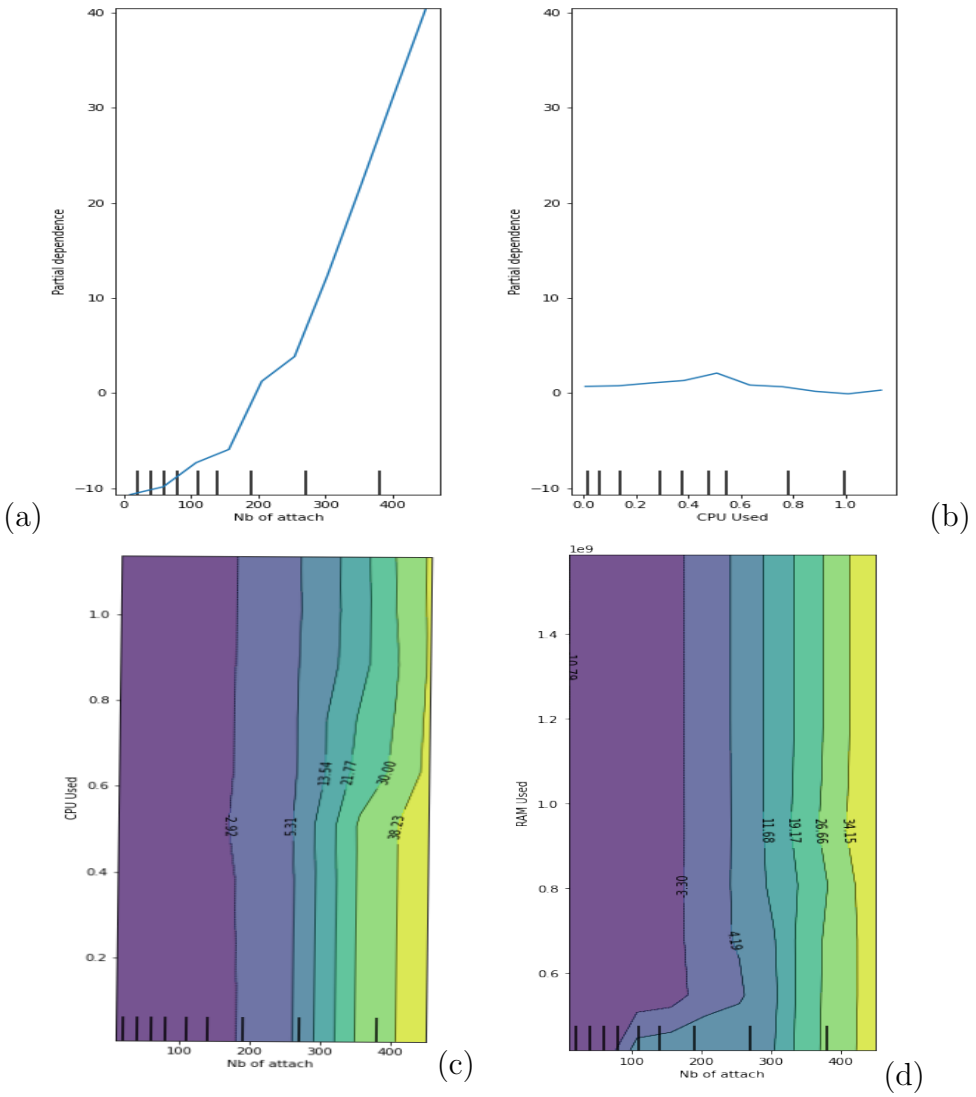


Fig. 6.2 Partial Dependence Plots on EARCD dataset.

lationship between features in the form of decision rules [24]. RuleFit builds a sparse linear model, leveraging both the dataset features and a set of new features that generated from the interactions between the original dataset features (decision rules). These new features are automatically built from decision trees, where each tree path is transformed to a decision rule. In our study, we apply RuleFit to generate hundreds of new decision rules. We then apply additional filtering/combining to determine the most important/informative rules. Therefore, such new rules will help slice provider's decision makers to better interpret and understand our FL-based model predictions.

6.4 Performance and Evaluation

In this section, we validate our XAI-empowered framework through an experimental study. We implemented the proposed XAI framework in Python, leveraging

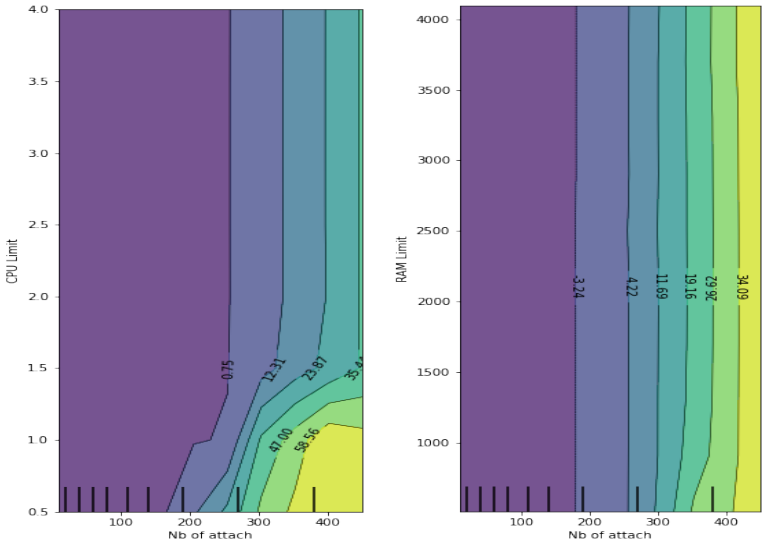


Fig. 6.3 Partial Dependence Plots on EARCD dataset.

tensorflow library as well as XAI libraries, including SHAP, and PDPBox. Following sections, we present the obtained results of the different XAI-techniques used.

6.4.1 Performance evaluation of Global Model-Agnostic Method

Fig. 6.2 shows the obtain partial dependence plots trained on our EARCD dataset. The left plot in the Fig. 6.2 depicts the effect of the number of attach requests on the latency; we can clearly see a linear relationship among them (Straight line), when the number of attach requests is superior to 200. Similarly, we analyze the effect of the used CPU on the latency (Fig. 6.2-b) as well as on the number of attach requests (Fig. 6.2-c), to show the interaction between these two features. We clearly see an interaction (different stripes and colors from purple to yellow) between the “Number of attach” and “CPU Used” features. Furthermore, the purple/blue color reflects the low dependence, while the green/yellow presents the high dependence of the features. For a number of attach less than 300 requests, the latency is nearly independent (=2.92%) of the CPU used, shown in purple and blue; whereas for values greater than 300 there is a strong dependence (30.00%) on CPU used (shown in green, and yellow color). Similarly, there is also a strong dependence between the ”number of attach” feature and ”RAM used”, ”CPU Limit”, and ”RAM Limit” features (Fig. 6.3), when their values are greater than ”300”, ”200”, and ”300”, respectively.

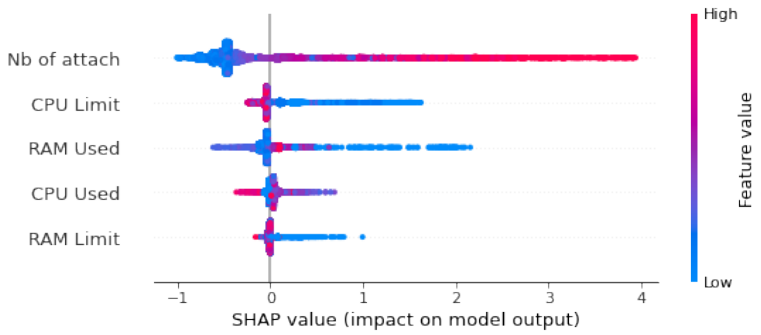


Fig. 6.4 SHAP value (impact on model output).

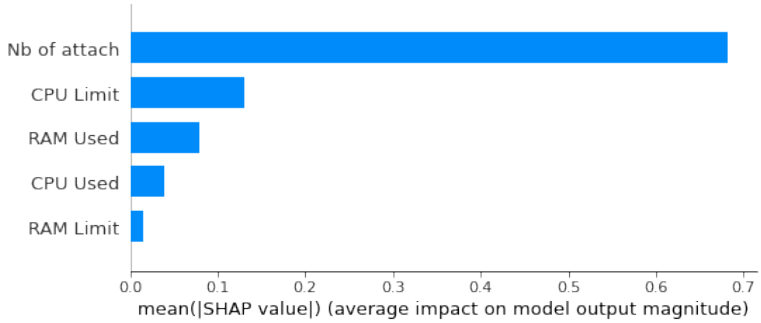


Fig. 6.5 SHAP value (average impact on model output).

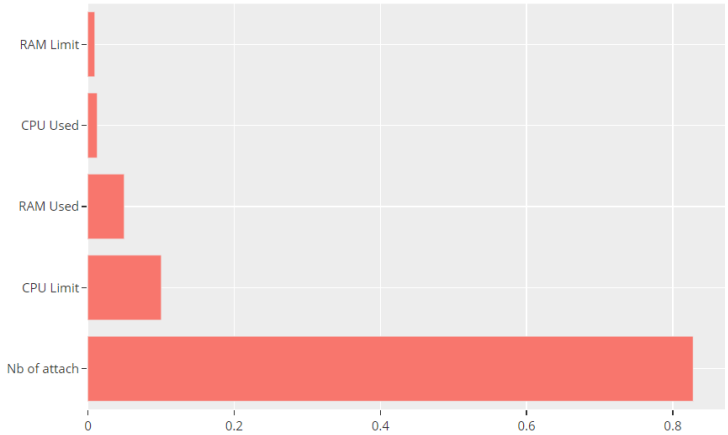


Fig. 6.6 Feature Importance using Rulefit on EARCD Dataset.

6.4.2 Performance evaluation of Local Model-Agnostic Method

Fig. 6.4 and Fig. 6.5 show the most informative features in an orderly manner. For a particular observation, each input feature has either a positive or a negative contribution to the final latency prediction (Fig. 6.4). Moreover, the blue color reflects the low value of the feature, while the Red color presents the high value of the features. Based on the obtained plots, we can notice that the most important feature is the number of received attach request. Also, the more number of attach is high, the more the latency is high. It is reasonable explanation because the processing duration of the received attach requests increases as the number of received requests increases as well.
After that, the second important feature is the "CPU Limit" feature. Indeed, the AMF functions with a high number of CPUs will generate a less latency, as compared with AMFs with low number of CPUs. Hence, the more resources allocated to AMFs are high, the more attach request treatment is faster, so the latency of attach requests decreases. Similarly, using the RuleFit method, Fig. 6.6 shows similar results of feature importance, which confirms the obtained results in Fig. 6.4 and Fig. 6.5 for the SHAP method. In Fig. 6.6, the highest scoring features corresponds to the following features: (1) number of attach: corresponds to the number of Attach Requests sent to the AMF; (2) CPU Limit: corresponds to the Number of CPU allocated to AMF; (3) RAM Used: corresponds to the memory used during AMF processing; (4) CPU Used: corresponds to the CPU used during the AMF processing; (5) RAM Limit: corresponds to the memory allocated to the AMF. Additionally, RuleFit method enables to generate new rules about dataset features and their combinations. Such rules will help to show the importance of the features for the model predictions.

Fig. 6.7 illustrates some rules of the RuleFit method through a table containing five main columns: "Rule" which is either an existing feature or a rule formula (combining more features). "Type", which is either linear for the existing features, or rule for a new generated rule. "Coef" which is the coefficient of the rules

Index	Rule	Type	Coef	Support	Importance
0	RAM Limit	linear	-1.7331745177629882e-05	1.0	0.019120719931916517
1	CPU Limit	linear	-0.3863503198444083	1.0	0.4437676624823355
2	RAM Used	linear	7.339808833175027e-11	1.0	0.02421002610926749
3	CPU Used	linear	-0.0	1.0	0.0
4	Nb of attach	linear	0.02773183326589316	1.0	3.731478647971506
218	Nb of attach <= 315.0	rule	-0.0	0.8380221653878943	0.0
110	Nb of attach <= 305.0	rule	-5.055127499190173	0.8300756170637752	1.898535809011758
371	Nb of attach <= 295.0	rule	-0.0	0.8169375534644996	0.0
152	Nb of attach <= 285.0	rule	-0.0	0.8093473924194927	0.0
214	Nb of attach <= 315.0 & Nb of attach <= 205.0	rule	-0.0	0.7215619694397284	0.0
153	Nb of attach <= 325.0 & Nb of attach <= 205.0	rule	-0.0	0.7190946855751325	0.0

Fig. 6.7 Rulefit results: Top 15 Rows of the Un-filtered Rules Features.

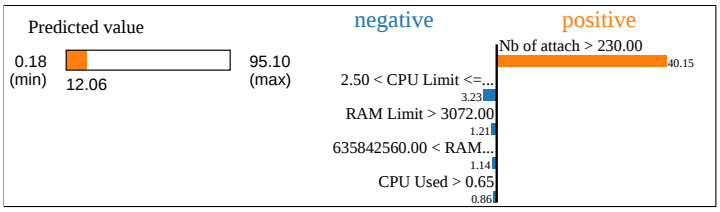
Index	Rule	Type	Coef	Support	Importance
108	Nb of attach <= 305.0 & Nb of attach <= 205.0	rule	-7.960556588113531	0.7169623846699787	3.5860281771803635
84	Nb of attach <= 305.0 & Nb of attach <= 195.0	rule	-0.32092101036293835	0.698094425483504	0.14732986873044338
109	Nb of attach <= 195.0 & RAM Used > 273678336.0 & Nb of attach <= 295.0	rule	-0.5977991989325068	0.6815906165069375	0.2784902910759459
26	Nb of attach <= 195.0 & RAM Limit > 384.0 & Nb of attach <= 295.0	rule	-4.476594397374145	0.6684180630502699	2.1074984685432097
269	Nb of attach <= 305.0 & Nb of attach <= 195.0 & RAM Limit > 384.0 & Nb of attach <= 135.0	rule	-1.0005563936082231	0.5568020448736154	0.4970394456376536

Fig. 6.8 Rulefit results: Top 5 Rows of the Filtered Rules.

and features. "Support" presents what proportion of the dataset supports the corresponding rule/feature. "Importance", which reflects the importance of each rule on the predictions. Noting that the feature importance is deduced from the weights of the regression model. Besides, in Fig. 6.8, we eliminate the insignificant features based on several criteria, such as (1) eliminate the rules with 0.00 coefficient since they are not significant; (2) sort the rules based on their support value; and (3) eliminate the leanest rules to keep more important rules.
As results, all the most significant rules involves the "Nb of attach" feature, which confirms the previous results (Fig. 6.4, Fig. 6.5, and Fig. 6.6). Therefore, we notice that the most significant rule combinations also mostly contain these features. As an example, the first rules above work as follow: **IF** "Nb of attach" $\leq$ 305 and (**and**) "Nb of attach" $\leq$ 205, **THEN** we have 3.58 as a feature importance for the KPI predictions. The second rule is very similar to the first one. Therefore, it might make sense to combine or eliminate some of the similar rules for a more interpretable set of rules. Ideally, we need to apply some filtered methods to group the similar rules and create a leaner model, which would cover all the interaction effects, which would ultimately help the ML analyzer to understand the working mechanism of our FL trained model.

Fig. 6.9 depicts the positive/negative impact of the features on our FL model predictions (latency KPI), using the LIME method. Using a test instance, the value of the predicted label, which is the latency, is equal to 12.06 seconds. It is presented by a bar on the left, and depicted by the given vector. We can see the colors blue and orange, depicting negative and positive associations, respectively. As we observe, the "number of attach" feature has a higher and positive impact (sign (+)) on the predicted latency, and when the number of attach is more than 230 requests, the higher latency we obtain. In addition, we also see that the "CPU Limit" feature has a lower/negative impact on the predicted latency (sign (-) in Fig. 6.9).

In general, we can deduce that our XAI-empowered framework provides a good performance in explaining our FL-based model, when predicting the latency KPI. It mainly helps to interpret and understand the local and global functioning of our FL-based model by showing the most informative features conducting to each



Features	Impact
Nb of attach > 230.00,	40.1456199741355
2.50 < CPU Limit <= 3.50,	-3.2278344868730575
RAM Limit > 3072.00,	-1.211170243871387
635842560.00 < RAM Used <= 835198976.00,	-1.1390523650426132
CPU Used > 0.65	-0.8561447261580017

Fig. 6.9 Lime results displaying the predicted label and the top five features impact.

prediction. Therefore, our XAI-based framework will help decision makers of slice providers, to not only detecting latency SLA-related violations for each running network slice, but also to determine the main reasons that conduct to such SLA violations, and hence studying how to anticipate them and use the results for the most advantageous network slice re-configuration (i.e, the resource allocated for the slice: RAM, CPU, etc.).

Additionally, according to the FL concept and the characteristics of the XAI algorithms, the latter can be classified into 2 categories: (1) XAI models run in the FL client side, which are PDP and Rulefit (because these algorithms are related to the local dataset and the local model (model fit)), and (2) XAI models run in the FL server side, which are LIME and SHAP (because these algorithms are related to the label prediction). However, for PDP, the FL clients will only share the plots that show the percentage of feature impact on the target label, in order to save their data privacy. Moreover, the reasons behind the usage of different XAI algorithms, that present similar results, is about increasing confidence in the outcomes of XAI algorithms.

6.5 Conclusion

In this chapter, we designed a novel XAI-powered FL-enabled framework that enables not only the prediction of the latency KPI, but also the interpretation of critical predictions made by the FL-based model. First, we built a new DNN model to predict the KPI (latency) of network slices, in federated way. Then, we developed multiple XAI models, i.e., RuleFit, LIME, SHAP, and PDP, on top of our federated DNN architecture, to enable more trust, credibility (in the local data, the local model and the FL global model), transparency, and explainability of the predictions made by our FL-based framework to different B5G ML users, such

as the slice provider. The in-depth experiments results on Latency predictions, showed the efficiency and explainiblity of our proposed FL framework. This makes it a promising FL framework and explainable Deep Learning FL Framework for its users.

CHAPTER 7

Conclusions

This chapter summarizes and lists important conclusions. Besides, different perspectives left for future work are discussed in the second part of this chapter.

7.1 Conclusions

The first contribution of this thesis, i.e., chapter 3, addressed the problem of trust between the different stakeholders in 5G network slices. The Network Slice provider sells end-to-end network slices (virtual end-to-end mobile network) to the vertical while leasing virtual and physical resources from Infrastructure Providers to enforce these end-to-end network slices. Accordingly, there is a need to establish SLA among these actors to ensure: (1) that the service is well-delivered to the vertical and (2) the infrastructure providers are respecting their involvement with the network slice provider. In the first contribution, we have proposed a trust architecture that establishes trust communication between different stakeholders with the introduction of Network Slicing, namely the vertical or tenant, the network slice provider, and the infrastructure provider. Actually, we propose a trust architecture to automatically manage the SLAs and apply penalties and compensations if the SLAs are not respected by one of the involved actors.

In the second contribution of this thesis, we focused on developing an automatic closed loop composed by different components, including MS, AE, DE, in order to avoid the DDoS in mMTC slice. The proposed solution focuses on security inside a single domain security of the domains, employing the three key components: MS, AE, and DE. These components offer an automatic and intelligent closed-loop to detect and mitigate security threats and attacks in real time. The real-time data collected from VNF are filtered by MS and exposed to AE that analyses and detects abnormal behaviour of the NS component in the heat of the moment, which makes DE intervention faster and in a timely manner before the threat spreads.

In the third contribution of this thesis, we introduced a novel secure federated learning framework, against poisoning attacks. Our framework automatically detects malicious participants in the federated learning process. To do so, we first generate a realistic dataset related to the latency as service-oriented KPI of running network slices. This dataset is used to build an analytic function about latency prediction, in a federated way. In addition, the basic idea of our framework is to dynamically select one participant as trusted entity, by leveraging deep reinforcement learning. The trusted participant will then be in charge of identifying poisoning model updates, using unsupervised machine learning. The numerical results show the efficiency of our framework in detecting poisoned models/data, and mitigating such attacks.

In the fourth contribution, we introduced a novel XAI empowered federated learning framework. Actually, FL-based solutions give "machine-centric" decisions about running network slices and their performance, which will be then executed/applied by managers, i.e., slice manager staff/module. However, FL-enabled solutions do not provide any details about why and how such decisions were made, and thus such decisions cannot be properly trusted/understood by slice managers. To alleviate this issue, XAI paradigm can be leveraged, which aims to improve the transparency of black-box FL decision-making process. In particular, XAI helps to explain the FL-based decisions to make them interpretable/trustable by network slices managers.

7.2 Perspectives

In this thesis, we have investigated some threats in network slicing aspects. We showed the effectiveness of our contributions on top of the considered scenarios. However, there are still several open and critical challenges that should be addressed/considered. In this section, we present some perspectives in link with our thesis's context.

7.2.1 Trust management between 5G stakeholders

The trust system is crucial to the security of network slicing. Trust between the different 5G stakeholders, and the tenants must be taken into consideration at several layers, since it is directly tied to the business model and overall architecture. There must be a level of confidence between the 5G parties, which should be reflected through legal agreements. This applies in the relation to tenants at any of the different architectural layers: resources, and slice providers. As an example, a slice manager cannot ,by default, trust the host platforms, and the host platforms cannot, by default, trust the slice manager, especially when the network slice utilizes resources from several physical networks. In our work, we introduce an end-to-end trusted architecture for network slicing between the tenants, slice provider, and infrastructure providers. However, the Monitoring Module of the different 5G stakeholders, used in the proposed 5G framework, should also be trusted.

7.2.2 5G Iot devices threats

The security in the 5G network has evolved and is more efficient than in the previous generations. The used 5G technologies increase the attack surface for threat actors. One of the primary points of attack has been identified as 5G customer

devices. Priorities in research should focus on reducing risks associated with end devices. Additionally, security measures brought by emergency access must be taken into account. Emergency services have proven to be a vulnerable point of access. Thus it will be further examined how to handle them at the slice level.
Moreover, The growth of the Internet of Things is accompanied by serious cyber-security risks, especially with the emergence of IoT botnets. In this context, intrusion detection systems (IDSs) proved their efficiency in detecting various attacks that may target IoT networks, especially when leveraging ML techniques. However, the proposed ML-based solutions make "machine-centric" decisions about intrusion detection in the IoT network, which are then executed by cyber security. One of the future work is to integrate XAI algorithms, which is a promising paradigm that helps to explain the decisions of Ml-based IDSs to make them understandable to cyber-security experts.

7.2.3 A secure isolation among network slices

Slice isolation is a major constraint to network slicing security. In 5G networks, various services have particular needs. The service quality for each slice must therefore be guaranteed. In fact, efficient slice isolation should make sure that a security problem or malfunction on one slice won't affect the other slices. Additionally, each slice's privacy should be protected. Future works in the research should treat the network slicing isolation challenge. A secure 5G architecture is needed that can ensure secure isolation for networks and prevent inter-slice attacks and interference between multiple slices.

LIST OF PUBLICATIONS

1. BEN SAAD Sabra, KSENTINI Adlen, and BRIK Bouziane: "A Trust architecture for the SLA management in 5G networks", In : ICC 2021-IEEE International Conference on Communications, 2021. p. 1-6.

2. BEN SAAD Sabra, KSENTINI Adlen, and BRIK Bouziane: "An end-to-end trusted architecture for network slicing in 5G and beyond networks", Wiley International Journal of Communication Systems: Security and Privacy, 2022, vol. 5, no 1, p. e186.

3. BEN SAAD Sabra, BRIK Bouziane, and KSENTINI Adlen: "A Trust and Explainable Federated Deep Learning Framework in Zero Touch B5G Networks", in : IEEE Global Communications Conference, 2022. p. 1-6.

4. BEN SAAD, Sabra, BRIK Bouziane, and KSENTINI Adlen: "Toward Securing Federated Learning against Poisoning Attacks in Zero Touch B5G networks", in IEEE Transactions on Network and Service Management, 2023.

5. Redouane Niboucha, Sabra Ben Saad, Adlen Ksentini, and Yacine Challal: "Zero-touch security management for mMTC network slices: DDoS attack detection and mitigation", in IEEE Internet of Things Journal, vol. 10, no. 9, pp. 7800-7812, 2023.

REFERENCES

[1] THALES, "Introducing and technology networks, white paper," Available at: https://www.thalesgroup.com/en/markets/digital-identity-and- security/mobile/inspired/5g, 2022.

[2] ETSI, "Network functions virtualisation (nfv); architectural framework," GS NFV 002 V1.2.1, 2014.

[3] Marius Corici, Pousali Chakraborty and Thomas Magedanz, "A study of 5g edge-central core network split options," Network, Berlin, Germany, 2021.

[4] Slawomir Kuklinski, Lechoslaw Tomaszewski, Robert Kolakowski, Anne-Marie Bosneag, Ashima Chawla, Adlen Ksentini, Sabra Ben Saad, Xu Zhao, Luis A. Garrido, Anestis Dalgkitsis, Bahador Bakhshi, Engin Zeydan, "Ai-driven predictive and scalable management and orchestration of network slices,," 16 November 2022, ITU Journal on Future and Evolving Technologies, Volume 3 (2022), Issue 3, Pages 570-588.

[5] T. Doukoglou, V. Gezerlis, and K. Trichias et al, "Vertical industries requirements analysis targeted kpis for advanced 5g trials," in 2019 European Conference on Networks and Communications (EuCNC), 2019, pp. 95–100.

[6] A. Ksentini and N. Nikaein, "Toward enforcing network slicing on ran: Flexibility and resources abstraction," IEEE Communications Magazine, vol. 55, no. 6, pp. 102–108, 2017.

[7] X. Li, M. Samaka, and H. A. Chan et al, "Network slicing for 5g: Challenges and opportunities," IEEE Internet Computing, vol. 21, no. 5, pp. 20–27, 2017.

[8] A. Ksentini, P. A. Frangoudis, A. PC, and N. Nikaein, "Providing low latency guarantees for slicing-ready 5g systems via two-level mac scheduling," IEEE Network, vol. 32, no. 6, pp. 116–123, 2018.

[9] N. Tuan Le, M. A. Hossain, A. Islam, D. Kim, Y. J. Choi, and Y. M. Jang,, ""survey of promising technologies for 5g networks," in hindawi publishing corporation mobile information systems, no. 2676589,," 2016, pp. 1–25.

[10] NGMN Alliance, "The next generation mobile networks (ngmn) ,," 5g white paper, Frankfurt, Germany, Feb. 2015.

[11] Symeon Papavassiliou, "Software defined networking (sdn) and network function virtualization (nfv)," Future Internet, 2020.

[12] 3GPP, "Study on architecture for next generation system, release 14," Sophia Antipolis, France, Rep. TR 23.799,, Dec. 2016.

[13] Int. Telecommun, "Framework of network virtualization for future networks, next generation network—future networks," Union, Geneva, Switzerland, ITU-T Recommendation Y.3011, Jan. 2012.

[14] I. Afolabi, T. Taleb, P. A. Frangoudis, M. Bagaa, and A. Ksentini, "Network slicing-based customization of 5g mobile services,," IEEE Network, vol. 33, no. 5, pp. 134–141, 2019.

[15] 3GPP Technical Specification TR 28.801, "Study on management and orchestration of network slicing for next generation network,," Sept. 2017.

[16] Y. Zaki, Liang Zhao, C. Goerg, and A. Timm-Giel, "Lte wireless virtualization and spectrum management," in WMNC2010, 2010, pp. 1–6.

[17] R. Mahindra, M. A. Khojastepour, H. Zhang, and S. Rangarajan, "Radio access network sharing in cellular networks,," in 2013 21st IEEE International Conference on Network Protocols (ICNP), 2013, pp. 1–10.

[18] R. Kokku, R. Mahindra, H. Zhang, and S. Rangarajan, "Nvs: A substrate for virtualizing wireless resources in cellular networks,," IEEE/ACM Transactions on Networking, vol. 20, no. 5, pp. 1333–1346, 2012.

[19] 3GPP, "Feasibility study on new services and markets technology enablers – stage 1,," vol. TR 22.891 release 14.

[20] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions,," IEEE Communications Surveys Tutorials, vol. 20, no. 3, pp. 2429–2453, 2018.

[21] A. D. La Oliva, X. C. Perez, A. Azcorra, A. D. Giglio, F. Cavaliere, D. Tiegelbekkers, J. Lessmann, T. Haustein, A. Mourad, and P. Iovanna, "Xhaul: toward an integrated fronthaul/backhaul architecture in 5g networks,," IEEE Wireless Communications, vol. 22, no. 5, pp. 32–40, 2015.

[22] A. S. Thyagaturu, Y. Dashti, and M. Reisslein, "Sdn-based smart gateways (sm-gws) for multi-operator small cell network management,," IEEE Transactions on Network and Service Management, vol. 13, no. 4, pp. 740–753, 2016.

[23] Hui Yang, Yizhen Wu, Jie Zhang, Haowei Zheng, Yuefeng Ji, and Young Lee, "Blockonet: Blockchain-based trusted cloud radio over optical fiber network for 5g fronthaul," *Optical Fiber Communication Conference, San Diego, California United States*, 2018.

[24] Hui Yang, Haowei Zheng, Jie Zhang, Yizhen Wu, Young Lee and Yuefeng Ji, "Blockchain-based trusted authentication in cloud radio over fiber network for 5g," *16th International Conference on Optical Communications and Networks (ICOCN), Wuzhen*, 2017.

[25] Babak Mafakheri, Tejas Subramanya, Leonardo Goratti and Roberto Riggio, "lockchain-based infrastructure sharing in 5g small cell networks," *14th International Conference on Network and Service Management, Rome*, 2018.

[26] Morocho-Cayamcela, M.E.; Lee, H.; Lim, W, "Machine learning for 5g/b5g mobile and wireless communications: Potential, limitations, and future directions,," IEEE Access 2019, 7, 137184–137206.

[27] Le, H.A.; Van Chien, T.; Nguyen, T.H.; Choo, H.; Nguyen, V.D, "Machine learning-based 5g-and-beyond channel estimation for mimo-ofdm communication systems,," Sensors 2021, 21, 4861.

[28] Aldossari, S.M.; Chen, K.C, "Machine learning for wireless communication channel modeling: An overview,," Wirel. Pers. Commun. 2019, 106, 41–70.

[29] Zakaria Abou El Houda, Bouziane Brik, and Lyes Khoukhi, "Ensemble learning for intrusion detection in sdn-based zero touch smart grid systems," 2022 IEEE 47th Conference on Local Computer Networks (LCN), 2022.

[30] N. Marir, H. Wang, G. Feng, B. Li, and M. Jia, "Distributed abnormal behavior detection approach based on deep belief network and ensemble svm using spark," IEEE Access, vol. 6, pp. 59 657–59 671, 2018.

[31] N. Moustafa, B. Turnbull, and K.-K. R. Choo, "An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things," IEEE Internet of Things Journal, vol. 6, no. 3, pp. 4815–4830, 2019.

[32] B. Brik and A. Ksentini, "On predicting service-oriented network slices performances in 5g: A federated learning approach," *International conference on Local Communications Networks*, nov 2020.

[33] P. A. F. M. B. Ibrahim Afolabi, Tarik Taleb and A. Ksentini, "Network slicing-based customization of 5g mobile services," *IEEE Network*, 2019.

[34] Bakri, Sihem and Brik, Bouziane and Ksentini, Adlen, "On using reinforcement learning for network slice admission control in 5g: Offline vs. online," *International Journal of Communication Systems*, vol. 34, no. 7, p. e4757, 2021, e4757 dac.4757. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/dac.4757

[35] O. Petar Popovski, Kasper F. Trillingsgaard and Giuseppe Durisi, "5g wireless network slicing for embb, urllc, and mmtc: A communication-theoretic view," *IEEE Access*, 2018.

[36] A. Amokrane, A. Ksentini, Y. H. Aoul, and T. Taleb, "Congestion control for machine type communications," *Proceedings of IEEE International Conference on Communications, ICC 2012, Ottawa, ON, Canada, June 10-15, 2012*, pp. 778–782, 2012.

[37] K. Piamrat, A. Ksentini, C. Viho, and J. Bonnin, "Qoe-aware vertical handover in wireless heterogeneous networks," *Proceedings of the 7th International Wireless Communications and Mobile Computing Conference, IWCMC 2011, Istanbul, Turkey, 4-8 July, 2011*, pp. 95–100, 2011.

[38] N. P. Boubakr Nour, Adlen Ksentini and Hassine Moungla, "A blockchain-based network slice broker for 5g services," *IEEE Networking Letters*, 2019.

[39] B. B. R. L. Z. G. Eder J. Scheid and B. Stiller, "Eder j. scheid, bruno b. rodrigues lisandro z. granville and burkhard stiller. "enabling dynamic sla compensation using blockchain-based smart contracts"," *IEEE International Symposium on Integrated Network Management*, 2019.

[40] Euromoney, "How does a transaction get into the blockchain?" Available: https://www.euromoney.com/learning/blockchain-explained/how-transactions-get-into-the-blockchain, 2020.

[41] Widya Nita Suliyanti, Riri Fitri Sari, "Evaluation of hash rate-based double-spending based on proof-of-work blockchain," International Conference on Information and Communication Technology Convergence (ICTC), Jeju, Korea (South), 2019.

[42] Li Zhetao, Kang Jiawen, Ye Dongdong, Deng Qingyong and Zhang Yan, "Consortium blockchain for secure energy trading in industrial internet of things," EEE Transactions on Industrial Informatics, 2017.

[43] Nada Chendeb, Nour Khaled, and Nazim Agoulmine, "Integrating blockchain with iot for a secure healthcare digital system," 8th International Workshop on ADVANCEs in ICT Infrastructures and Services (ADVANCE 2020), Candy E,2020.

[44] J. H. Noh and H.-Y. Kwon, "A study on smart city security policy based on blockchain in 5g age," *International Conference on Platform Technology and Service (PlatCon), Jeju, Korea (South)*, 2019.

[45] Nick Szabo, "Smart contracts," *https://www.fon.hum.uva.nl/rob/Courses/eInfor*, vol. mationInSpeech/CDROM/Literature/LOT winterschool2006/szabo.best.vwh.net/smart.contracts.html, 2020.

[46] Konstantinos Christidis and Michael Devetsikiotis, "Blockchains and smart contracts for the internet of things," *IEEE Access*, 2016.

[47] Chaer Abdulla, Salah Khaled, Lima Claudio, Ray Partha and Sheltami Tarek, "Blockchain for 5g: Opportunities and challenges," *IEEE Globecom, USA*, 2019.

[48] Baran Köseoğlu, "Data science studies: Ethereum, blockchain-based distributed computing platform," *https://medium.com/colendi/data-science-studies-ethereum-Blockchain-based-distributed-computing-platform-eae96cde70f7*, 2019.

[49] Stefan Beyer, "What is ethereum ganache?" *https://www.mycryptopedia.com/what-is-ethereum-ganache*, 2019.

[50] Mayank Sahu, "What is truffle suite? features," *https://www.upgrad.com/blog/what-is-truffle-suite/*, 2020.

[51] Ethereum, ". "solidity"," *https://solidity.readthedocs.io/en/v0.7.1/*, 2020.

[52] Arne Holst, "Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2030," Aug. 2021, uRL: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/. [Online]. Available: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/

[53] ETSI, "Architecture enhancements for 5g system (5gs) to support network data analytics services (3gpp ts 23.288 version 16.4.0 release 16)," July 2020. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/123200_123299/123288/16.04.00_60/ts_123288v160400p.pdf

[54] Ahmed Amokrane, Adlen Ksentini, Yassine Hadjadj Aoul, and Tarik Taleb, "Congestion control for machine type communications," pp. 778–782, 2012. [Online]. Available: https://doi.org/10.1109/ICC.2012.6364152

[55] O. Arouk and A. Ksentini, "Multi-channel slotted aloha optimization for machine-type-communication," pp. 119–125, 2014. [Online]. Available: https://doi.org/10.1145/2641798.2641802

[56] Ericsson, "A guide to 5G network security 2.0," Sep. 2021, uRL: https://www.ericsson.com/4a66f8/assets/local/news/2021/09172021-a-guide-to-5g-network-security-2.0.pdf. [Online]. Available: https://www.ericsson.com/4a66f8/assets/local/news/2021/09172021-a-guide-to-5g-network-security-2.0.pdf

[57] Jordan Lam and Robert Abbas, "Machine Learning based Anomaly Detection for 5G Networks," Mar. 2020, uRL: https://arxiv.org/abs/2003.03474. [Online]. Available: https://arxiv.org/abs/2003.03474

[58] Grant Millar, Anastasios Kafchitsas, and Orestis Mavrooulos et al, "D2.1: 5G Security: Current Status and Future Trends," jul 2020, in "INtelligent Security and PervasIve tRust for 5G and Beyond. [Online]. Available: https://www.inspire-5gplus.eu/wp-content/uploads/2020/05/i5-d2.1_5g-security-current-status-and-future-trends_v1.0.pdf

[59] Amir Alimohammadifar, Suryadipta Majumdar, Taous Madi, and Yosr Jarraya, "Stealthy Probing-Based Verification (SPV): An Active Approach to Defending Software Defined Networks Against Topology Poisoning Attacks," in *23rd European Symposium on Research in Computer Security, ESORICS 2018*, Aug. 2018, uRL: https://users.encs.concordia.ca/~wang/papers/esorics18amir.pdf. [Online]. Available: https://users.encs.concordia.ca/~wang/papers/esorics18amir.pdf

[60] Antoine Delplace, Sheryl Hermoso, and Kristofer Anandita, "Cyber Attack Detection thanks to Machine Learning Algorithms," Jan. 2020, uRL: https://arxiv.org/abs/2001.06309. [Online]. Available: https://arxiv.org/abs/2001.06309

[61] João Henrique Corrêa, Patrick Marques Ciarelli, Moises R. N. Ribeiro, and Rodolfo da Silva Villaca, "On Machine Learning DoS Attack Identification from Cloud Computing Telemetry," Apr. 2019, uRL: https://arxiv.org/abs/1904.06211. [Online]. Available: https://arxiv.org/abs/1904.06211

[62] Rohan DDoShi, Noah Apthorpe, and Nick Feamster, "Machine Learning DDoS Detection for Consumer Internet of Things Devices," apr 2018.

[63] Faisal Hussain, Syed Ghazanfar Abbas, Muhammad Husnain, Ubaid U. Fayyaz, Farrukh Shahzad, and Ghalib A. Shah, "IoT DoS and DDoS Attack Detection using ResNet," in *IEEE 23rd International Multitopic Conference (INMIC)*, Dec. 2020, uRL: https://arxiv.org/abs/2012.01971. [Online]. Available: https://arxiv.org/abs/2012.01971

[64] Xiaoyong Yuan, Chuanhuang Li, and Xiaolin Li, "DeepDefense: Identifying DDoS Attack via Deep Learning," in *IEEE International Conference on Smart Computing (SMARTCOMP)*, May 2017, uRL: https://ieeexplore.ieee.org/document/7946998. [Online]. Available: https://ieeexplore.ieee.org/document/7946998

[65] Maryam Ghanbari and Witold Kinsner, "Extracting Features from Both the Input and the Output of a Convolutional Neural Network to Detect Distributed Denial of Service Attacks," in *IEEE 17th International Conference on Cognitive Informatics and Cognitive Computing (ICCI*CC)*, Jul. 2018, uRL: https://ieeexplore.ieee.org/document/8482019. [Online]. Available: https://ieeexplore.ieee.org/document/8482019

[66] Roberto Doriguzzi-Corin, Stuart Millar, Sandra Scott-Hayward, Jesus Martinez-del-Rincon, and Domenico Siracusa, "Lucid: A Practical, Lightweight Deep Learning Solution for DDoS Attack Detection," vol. 17, no. 2, Feb. 2020, uRL: https://arxiv.org/abs/2002.04902. [Online]. Available: https://arxiv.org/abs/2002.04902

[67] A. Michael Schachter, "DDoS and 5G: The bigger the pipe, the stronger the threat,," June 26 2018.

[68] Sean Newman, Enterprises Beware:, "Variations on the mirai malware still feeding ddos attacks," 2022.

[69] Mahmoud Ammar, Giovanni Russello b, Bruno Crispo, "Internet of things: A survey on the security of iot frameworks, journal of information security and applications," 2018.

[70] Raja Ettiane, Rachid EL Kouch, "Mitigating denial of service signaling threats in 5g mobile networks," (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 12, No. 2,2021.

[71] A. S. Mamolar, Z. Pervez, Q. Wang et al, "Towards the detection of mobile ddos attacks in 5g multi-tenant networks," 2019 European Conference on Networks and Communications (EuCNC), 2019, pp. 273-277, doi: 10.1109/EuCNC.2019.8801975.

[72] Florian Metzger, Tobias Hoßfeld, André Bauer, Samuel Kounev, and Poul E. Heegaard, "Modeling of Aggregated IoT Traffic and its Application to an IoT Cloud," *Proceedings of the IEEE*, Mar. 2019. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8674845

[73] Markus Laner, Philipp Svoboda, Navid Nikaein, and Markus Rupp, "Traffic Models for Machine Type Communications," in *ISWCS 2013; The Tenth International Symposium on Wireless Communication Systems*. VDE, Aug. 2013. [Online]. Available: https://www.eurecom.fr/en/publication/4079

[74] Hatim Chergui, Adlen Ksentini, and Luis Blonco, et al, "Toward zero-touch management and orchestration of massive deployment of network slices in 6g," accepted in IEEE Wireless Magazine, special issue on 6G: The paradigm for future wireless comunications, 2021.

[75] ETSI, "5G; Security architecture and procedures for 5G System (3GPP TS 33.501 version 15.1.0 Release 15)," Jul. 2018. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/133500_133599/133501/15.01.00_60/ts_133501v150100p.pdf

[76] Mason, Llew and Baxter, Jonathan and Bartlett, Peter and Frean, Marcus, "Boosting algorithms as gradient descent," in *Advances in Neural Information Processing Systems*, vol. 12. MIT Press, 1999. [Online]. Available: https://proceedings.neurips.cc/paper/1999/file/96a93ba89a5b5c6c226e49b88973f46e-Paper.pdf

[77] Vu, Quang and Ruta, Dymitr and Cen, Ling, "Gradient boosting decision trees for cyber security threats detection based on network events logs," 2019, pp. 5921–5928.

[78] Shah Zeb, Aamir Mahmood, Syed Ali Hassan, et al, "Industrial digital twins at the nexus of nextg wireless networks and computational intelligence: A survey," Journal of Network and Computer Applications, Volume 200, 103309, ISSN 1084-8045, https://doi.org/10.1016/j.jnca.2021.103309, 2022.

[79] Etsi, "Zero touch network service management (zsm)," [Online]. Available: https://www.etsi.org/technologies/zero-touch-network-service-management, 10 January 2022.

[80] Slawomir Kuklinski and Lechoslaw Tomaszewski, "Key performance indicators for 5g network slicing," IEEE Conference on Network Softwarization (NetSoft), pp. 464–471, June 2019.

[81] Virat Shejwalkar, Amir Houmansadr, Peter Kairouz , et al, "Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning," the IEEE Symposium on Security Privacy, 2022.

[82] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov, "How to backdoor federated learning," CoRR,2018.

[83] Peva Blanchard, Rachid Guerraoui, Julien Stainer, et al, "Machine learning with adversaries: Byzantine tolerant gradient descent," In NeurIPS, pages 119–129, 2017.

[84] Clement Fung, Chris J.M. Yoon, Ivan Beschastnikh, "Mitigating sybils in federated learning poisoning," CoRR, 02 Sep 2018.

[85] Gu, T., Dolan-Gavitt, B., Garg, S, "Badnets: identifying vulnerabilities in the machine learning model supply chain," CoRR, 2017.

[86] Yi Liu, Jialiang Peng, Jiawen Kang, et al, "A secure federated learning framework for 5g networks," IEEE wireless communications magazine, March 2020.

[87] Phillip Rieger, Thien Duc Nguyen, Markus Miettinen, et al., "Deepsight: Mitigating backdoor attacks in federated learning through deep model inspection," CoRR, 2022.

[88] Mustafa Safa Ozdayi, Murat Kantarcioglu, Yulia R. Gel, "Defending against backdoors in federated learning with robust learning rate," CoRR, 2020.

[89] Matthew Jagielski, Alina Oprea, Battista Biggio , et al., "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," 2018 IEEE Symposium on Security and Privacy, 2018.

[90] Y. Zhao, J. Chen, J. Zhang, D. Wu, J. Teng, S. Yu. , "Pdgan: a novel poisoning defense method in federated learning using generative adversarial network proceedings of ica3pp," Springer, 2019.

[91] Mahesh Subedar, Nilesh A. Ahuja, Ranganath Krishnan, et al. , "Deep probabilistic models to detect data poisoning attacks," CoRR, 2019.

[92] Close J. Steinhardt, et al., "Certified defenses for data poisoning attacks proceedings of neurips," pp. 3518-3530, 2017.

[93] B. McMahan, E. Moore, D. Ramage, S. Hampson, et al, "Communication-efficient learning of deep networks from decentralized data," in Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, USA, 2017.

[94] M. Isaksson et al, "Secure federated learning in 5g mobile networks," IEEE GLOBECOM, 2020.

[95] Guan Wang, "Digital and smart analytics, interpret federated learning with shapley values," FML Workshop, Hong Kong, 2019.

[96] Jelena Fiosina, "Explainable federated learning for taxi travel time prediction," VEHITS, 2021.

[97] Christoph Molnar, "Interpretable machine learning a guide for making black box models explainable," Chapter 6, 2022.