POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Authorizing Access to Edge Resources using 5G Device Authentication

Tesi di Laurea Magistrale in
Computer Science and Engineering
Ingegneria Informatica

Author: **Luca Giacometti**

Student ID: 963272
Advisor: Prof. Giacomo Verticale
Academic Year: 2022-23

Ai miei genitori ed a mio fratello
con tanto affetto

Abstract

Authentication and authorization are very important aspects for the validation of a user who desires to get access to a network, a system or a service.

Specifically, authentication is the procedure of test "who someone is", authorization is the procedure which verifies "what someone can do".
In our scenario, they mean if a user is recognized for a service and if they are able to gain access to that service.

Recently, the 5G networks are expanding more and more and they present a lot of differences between them and the previous network generations; consequently, the components within it are deployed differently arising new challenges. Moreover, the advent of the SDN approach has disclosed new ways to conceptualize security protocols because of the possibility to separate data plane from control plane.

The Edge Computing paradigm carries with it unmatched authentication and authorization challenges; one example to be taken into account is regarding the implementations of such mechanisms in a centralized or distributed way, with the pros and cons of each of the choices (e.g., if a centralized process is under a disruption or compromised by an attack, the entire system may collapse).
Furthermore, the authentication procedure can trust on users located in the same trust domain (some authentication schemes that exploit location-specific informations have been suggested, especially for those cases where the edge data centers are situated near of end-users).

This thesis illustrates a Customer Edge Switch that is designed to be collocated at the border of a 5G network, which authenticates users with a policy-based control and implements an network-layer protocol for authorizing them to get access to services by leveraging the Mobile Edge Computing paradigm.

The CES component is built with a P4 program as data plane which authorizes the packet traffics, an orchestrator written in Python as control plane to administer the match-and-action tables in the switch and a policy server to authenticate users.

The environment used to simulate the components is based on a set of virtual machines linked to a BMv2 programmable switch.

To test the efficiency of our authorization protocol, it has been led a performance analysis under some remarks.

Keywords: 5g, mobile edge computing, network service header, p4, authorization protocol, sdn

Abstract in lingua italiana

L'autenticazione, l'autorizzazione sono fondamentali per la validazione di un utente che desidera accedere ad una rete, ad un sistema o ad un servizio.

Per autenticazione si intende quel processo che risponde alla domanda "chi è quell'entità", con autorizzazione si definisce quella procedura che soddisfa la domanda "che cosa può fare quell'entità".
Nello senario considerato, queste domande sono utilizzate per capire se un utente è autenticato ed autorizzato per accedere ad un determinato servizio.

Recentemente, le reti 5G si sono espanse sempre di più, facendo emergere delle differenze rispetto alle passate generazioni. Queste differenze si ripercuotono sul funzionamento dei componenti interni. In più, con l'avvento del paradigma SDN, sono sorte nuove metodologie di sviluppo per i protocolli di sicurezza, dispiegati grazie alla possibilità di dividere il piano di controllo da quello dei dati.

Il paradigma basato sul Edge Computing ha portato alla luce nuove sfide di autenticazione ed autorizzazione da risolvere; la possibilità di implementare questi meccanismi di sicurezza in modo distribuito o centralizzato ne è un chiaro esempio (se un sistema centralizzato dovesse fallire per via di un attacco informatico, tutto il sistema potrebbe collassare).
In aggiunta, con questo paradigma è possibile fidarsi di un utente che è presente all'interno dello stesso dominio della rete (infatti, sono stati proposti diversi protocolli di autenticazione che si basano sulle informazioni riguardanti la posizione dell'utente, in particolar modo se questo risiede all'interno della rete).

Questa tesi si basa sull'ideazione di un componente switch per il paradigma Mobile Edge Computing ed è realizzato per essere collocato al bordo di una rete 5G. Questo dispositivo autentica gli utenti tramite una politica di accesso ed implementa un protocollo di rete per autorizzare gli utenti a dei servizi.

Il componente in questione è stato progettato utilizzando il linguaggio P4 per il piano dei dati tramite il quale vengono autorizzati i pacchetti, un controllore scritto in Python per il piano di controllo con il quale vengono gestite le tabelle nello switch ed un server dedicato alla politica di accesso per autenticare gli utenti. L'ambiente utilizzato per simulare il componente ideato è costituito da delle macchine virtuali collegate fra di loro tramite uno switch programmabile di tipo BMv2.

Per verificare l'efficienza del nostro componente, è stato condotto un test sulle prestazioni del nostro protocollo di autorizzazione ottenendo dei buoni risultati.

Parole chiave: protocollo di autorizzazione, mobile edge computing, network service header, p4, 5g, sdn

Contents

Introduction

With the introduction of Mobile Edge Computing (MEC) paradigm, a new way for deploying resources close to data and end-users has emerged. The 5G networks are one opportunity to take advantage of the MEC infrastructure paradigm and to release multiple deployment scenarios by incorporating a fully private edge and a service-provider manager edge where applications are shared between several entities.

The resources of this paradigm are collocated externally the 5G domain. Therefore, they do not have to blindly rely on the 5G authentication and authorization security protocols, because it would be like introducing some potential threats for the edge node, involving a possible impairment of all system.

The project was aimed to design of an Access Control node: it is a device which checks if a user is authenticated for a specific service implementing a policy-based control and authorizes them to access that requested service exploiting a network-layer protocol.

This system is conceived to be deployed at the edge of a 5G network, falling into a scenario which incorporates the previous MEC paradigm. The architecture of this device is designed by following the Software Defined Networking (SDN) paradigm, which is becoming more and more incisive in the development of infrastructures and it is covered in depth in Section 2.2.

This device is in charge of applying the previously mentioned security mechanisms at the network layer, to defend the services from various types of attack, which are inspected in Section 4.5.2. It has also to provide them dynamically adapting to requests, avoiding to impact excessively over the standard performance.

For what concern the performance, we had carried out specific analysis to verify the behaviour of the designed protocol and the results are described in Chapter 5.

This thesis derives from a macro-project called AI-SPRINT.

AI-SPRINT

Over these years, the Artificial Intelligence is growing more and more incrementing the economical benefits in basically every fields, finding continually new use cases. One particular area of application is represented by the Edge Computing due to the presence of interesting assets.

Taking advantage of the edge nodes will make a real impact optimizing the computational resources and improving the security level. Nonetheless, AI applications miss a concrete way to split AI models and components boosting performance, efficiency and level of security.

AI-SPRINT (which is standing for Artificial Intelligence in Secure PRIvacy-preserving computing conNTinuum) is an international project which had received funding from the European Union Horizon 2020 research and innovation program, finalised in drawing up a new framework for the development of AI applications ([1]).

In particular, it is an ongoing framework presented in [33], which has the purpose of resolving the previous challenges becoming a configuration software for providing and partitioning AI applications present in the multitude of cloud-based solutions and AI-based sensor components.

As described in [1, 33], the most important objectives which AI-SPRINT aims to reach are:

- Provide design and development tools: the idea is to create a set of software to implement various AI applications (such as Machine Learning and Deep Learning), which use up resources allocated across the computing continuum.
- Deliver tools for secure execution and privacy presentation: the starting point is engineering a concept which can enable a secure AI models deployment and data processing merging the confidentiality and integrity of hardware with a policy-defined privacy.
- Develop a run-time environment for application execution and monitoring: the will is to realize a framework that can elaborate policies knowing the underneath infrastructure and applications to be executed.
- Provide advanced solutions for AI architecture enhancement: it consists in designing a system which has the job of making an iterative refinement of AI application architecture design and deployment, improving the planning and rollout phases of AI model and application for a target environment by leveraging edge computing.

Use Cases

There are three leading areas of interest where AI-SPRINT is involved ([1]):

- Personalised Healthcare: the target for this sector is to collect a discrete amount of patient data to elaborate the stroke risk and try to prevent it. In order to do so, it is necessary that devices will be able to catalogue anonymously these kinds of information in compliance with privacy and security assumptions.
- Maintenance and inspection: one of challenges for AI is making predictions. In this case, the focus will be to process an enormous quantity of data to improve the AI foretelling of when and what type of maintenance is required in a wind turbine utilizing the knowledge from pictures retrieved from drones and analysis derived from edge-cloud nodes.
- Farming 4.0: the aim is to optimise phytosanitary treatments, in vineyard plantations, using edge and intelligent sensors. Specifically, the result will include the development of new AI models to retrieve data from sensors and to build and deploy solutions.

Architecture

The architecture of AI-SPRINT project is illustrated as the union of two frameworks:

- Design framework: it consists on four modules:
  - Design and programming abstractions, for application components.
  - Performance models, by leveraging machine learning.
  - AI models architecture search, based on deep neural networks.
  - Applications design space exploration, to retrieve new models exploiting operational research methods.
- Run-time framework: it is in charge of handling all phases related to the deployment, enhancement and maintenance of AI applications. It includes tools to:
  - empower the iterative deployment and concurrent execution of AI applications.
  - trigger automated model retraining by leveraging on solutions for training and retraining at the edge.
  - maximize scheduling and allocation of resources among competing training jobs.

Taking into account the run-time framework and one of the key goals related to guarantee a secure environment and the protection of privacy, all the possible use cases could be summed up in this scenario: an amount of users which desire to gain access to a few services. In order to do so, these users need to be authenticated and authorized via a protocol.

Accordingly, the envisaged context consists in a bunch of users connected to a 5G network, a node collocated at the edge of this network that checks out the several packet traffics and some services to which users can get permission to reach in case of positive control of the security mechanisms (this is fully described in Chapter 4).

Thesis structure

This thesis is arranged in six chapters, in this order:

- Chapter 1 - "Related work" - includes some relevant papers and documents which have been considered and studied before and during the evolution of this project.
- Chapter 2 - "Background" - contains basic notions on which this same work is based on; in particular, the topics are concerning the technologies present in a programmable network such as SDN and Edge Computing.
- Chapter 3 - "Project description" - incorporates the theoretical assumptions behind this project such as the scenario in which the components and security model considered are collocated.
- Chapter 4 - "Project implementation" - discloses how this authorization protocol is implemented with a particular dwelling on the division done between control and data plane.
- Chapter 5 - "Performance analysis" - describes the analysis regarding the performance of protocol and the methodology used to retrieve the times samples.
- Chapter 6 - "Conclusions" - encloses the final considerations and a possible future work.

1 | Related work

In this chapter, we want to introduce some documents and papers which have been analyzed during the preliminary phase of the project with the aim of better finding out the project scenario features regarding the network architecture, the behavior of some of its components and how it was possible improving the authorization protocol.

1.1. Private 5G network deployment scenarios

In this web article [34], some deployment scenarios of a private 5G network were debated. They considered two main different methods for the deployment:

- Physically-isolated private 5G network (described in Figure 1.1): one way is creating a private 5G network by enterprises or mobile operator which is standalone respect to the mobile operator's public 5G network.
- Mobile operator's public 5G network with resources shared (presented in Figure 1.2): in this contest, the private 5G network is based on a mobile operator's public resources.

In particular, the author wanted to outline seven different possible cases, describing each of them. It is possible classify these ones in three categories, grouping them by the main focus:

- Isolated 5G LAN built by enterprise (local 5G frequency) or mobile operator with a licensed frequency.
- RAN and control plane or core with end-to-end network slicing sharing between private and public network.
- LBO (Local Breakout) enhancing different interfaces.

For what concern the planned scenario of this project, the situations comparable with the scenario of this work are two: RAN and control plane sharing between private and public network and N3 LBO (local breakout) with the introduction of a MEC data plane in the

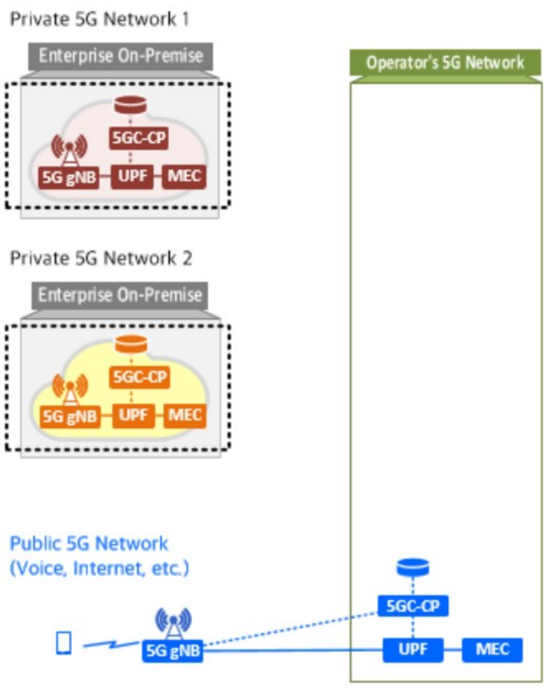


Figure 1.1: Physically-isolated private 5G network independent of the mobile operator's public 5G network ([34])

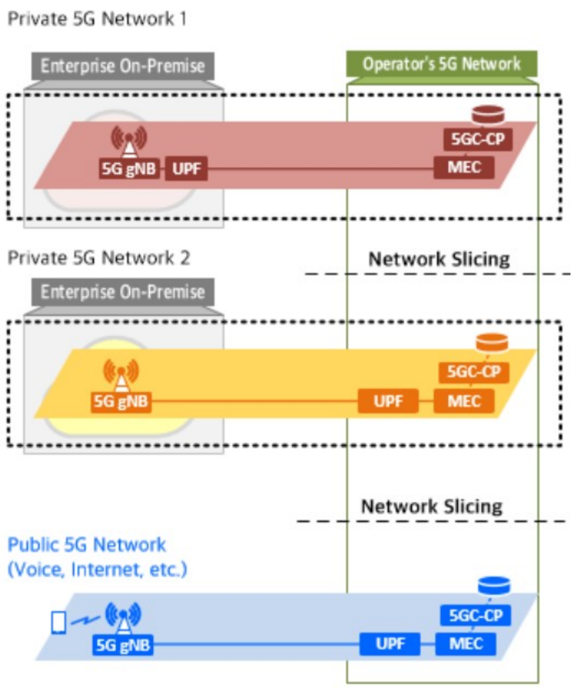


Figure 1.2: Private 5G network sharing mobile 5G network resources of mobile operators ([34])

enterprise structure.

In the first one case, depicted in Figure 1.3, we have the gNB, 5GC CP and UDM shared between public and private slices with a logical separation among public and private network. Meanwhile, the UPF and MEC components are not shared between the two slices and they are physically separated.

In the second scenario, illustrated in Figure 1.4, the case of SK Telecom company in Korea is represented. Here, the gNB is still shared between the private and public slices. It is maintained inside the enterprise system and it handles the public traffic. In this configuration, the Korean holding had introduced a MEC data plane in the enterprise, attached it to the gNB and created a MEC control plane to manage the packet traffic rerouting the packets into the local breakout or towards the UPF public node.

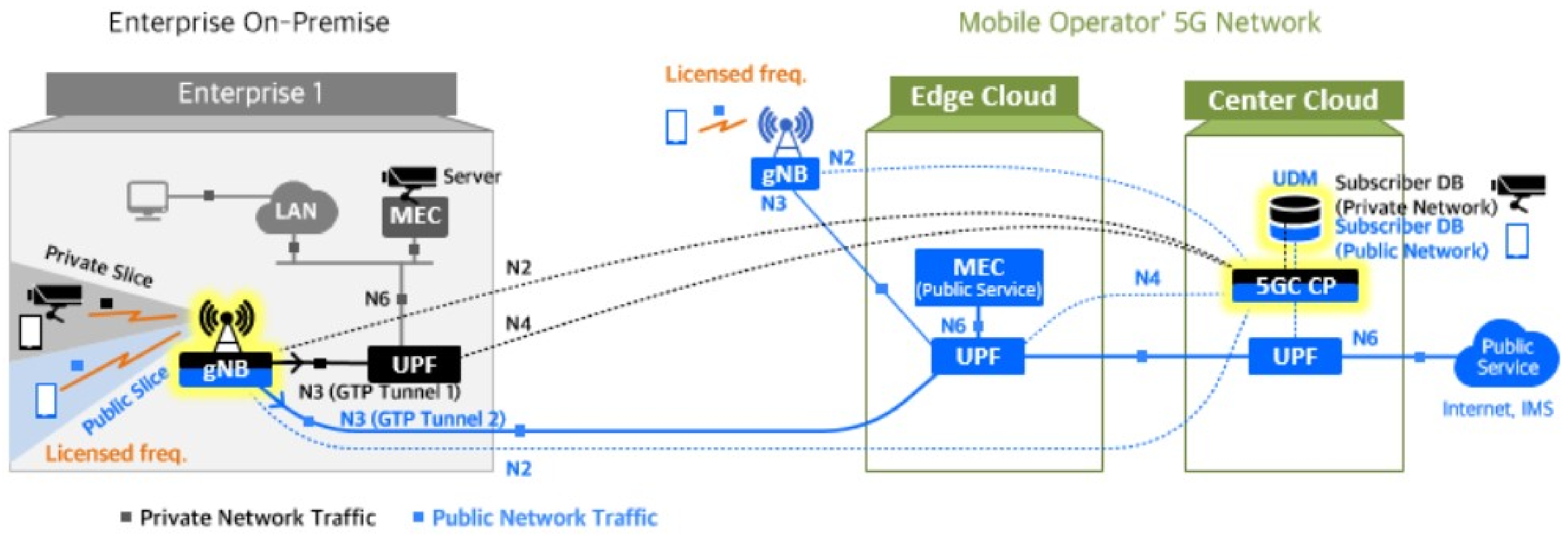


Figure 1.3: RAN and Control Plane Sharing between private and public network ([34])

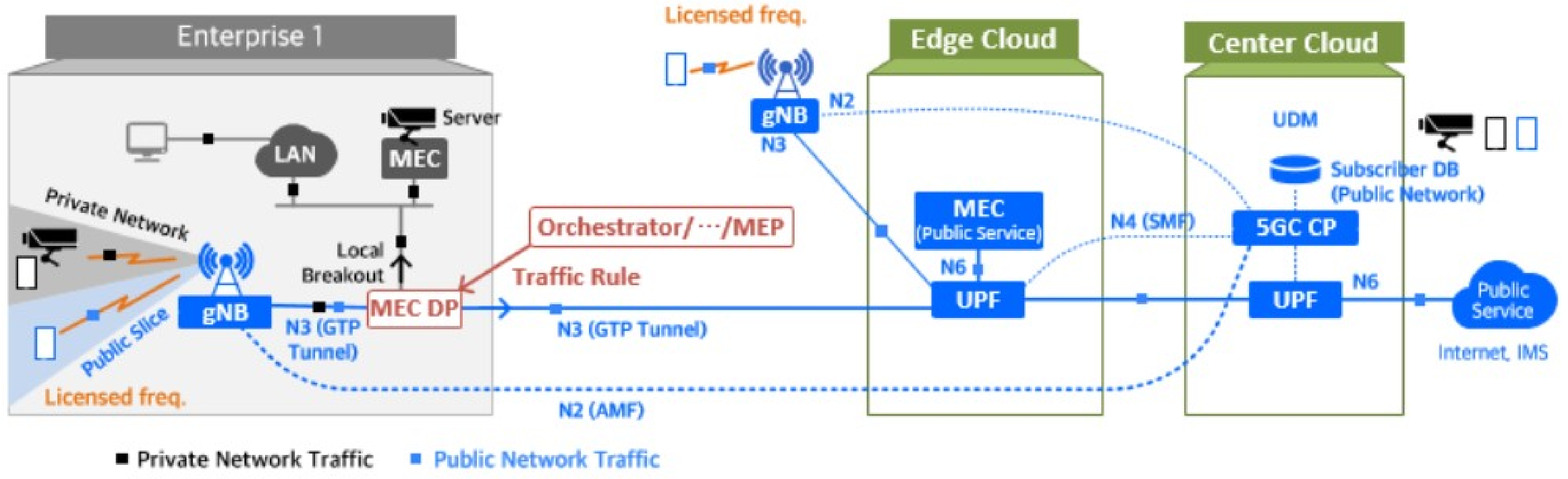


Figure 1.4: Local Breakout on the N3 interface: Case of SK Telecom in Korea ([34])

1.2. Customer Edge Switching (CES)

The second work which is correlated with this thesis is [20], which has proposed a 5G security architecture similar with that used in this project, having also the same goal: utilizing a secure policy to prevent malicious packet traffics.

As presented in the Figure 1.5, the security architecture was designed with two main components: a Customer Edge Switch (CES) and a Security Policy Management (SPM).

The CES component behaves as a firewall applying the policy given by the SPM subsystem. The conception of it is based on SDN principles, in particular on splitting control plane and data plane. Moreover, it is able to schedule the communication policies between the CES nodes via Customer Edge Traversal Protocol (CETP).

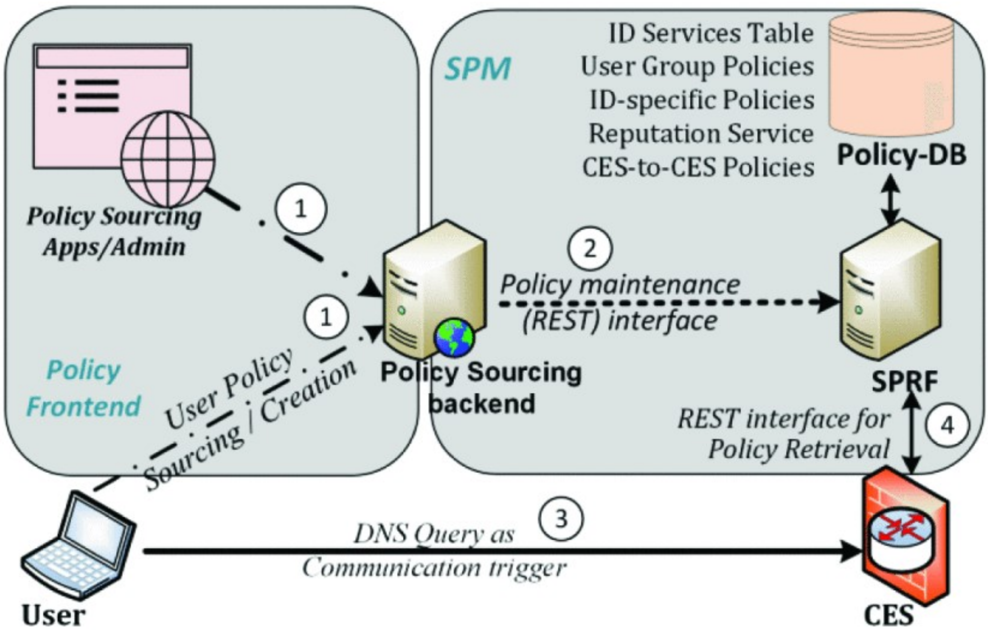

SPM
ID Services Table
User Group Policies
ID-specific Policies
Reputation Service
CES-to-CES Policies
Policy-DB
Policy Sourcing Apps/Admin
1
1
2
Policy maintenance (REST) interface
Policy Frontend
User Policy Sourcing / Creation
Policy Sourcing backend
SPRF
REST interface for Policy Retrieval
4
3
DNS Query as Communication trigger
User
CES

Figure 1.5: Security Policy Management in a CES component ([20])

In this related paper, two types of policies were considered: firewall security and CETP-based communication policies.

The first one consists in filtering the traffic in data plane accepting or rejecting the packets according to the network administrators defined policy groups.

The second policy is relied on the CETP policy concept which is articulated in a set of policy elements interchanged between CES devices managing sender and destination, assuring compliance and, as a consequence, achieving that a node interoperates only with the expected one. It is negotiated at CES-CP level, categorized into host-to-host (H2H) policy and CES-to-CES (C2C) policy.

The Security Policy Rules Function node was implemented with these components (outlined in Figure 1.6):

- REST server: it exposes a standard REpresentational State Transfer interface to propagate the queries for SPM where the usage of TLS makes possible the authentication of entities.
- Policy API: it is the only actor who can interact with the database, executing the basic operations regarded to the policies inside that database.
- Database client: it keeps track of all policies, services and internet entities reputation.

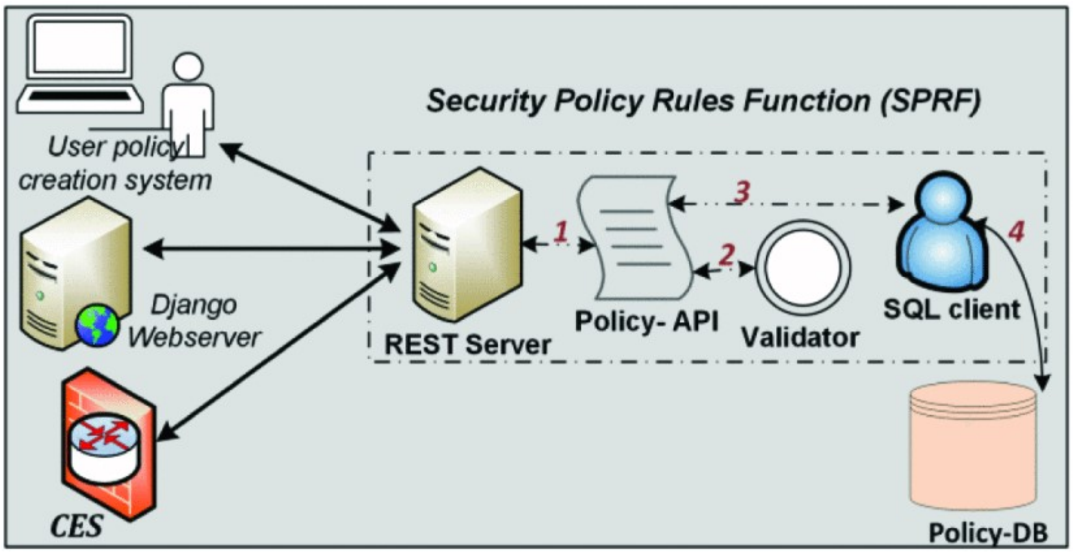


Figure 1.6: Detailed implementation of SPRF node ([20])

Each of the two components (i.e., policy API and REST server) were developed as an asynchronous framework written in Python programming language being able to handle the requests and the responses asynchronously.

There was also another minor component inside the SPM: a validator module. This part of the subsystem had the role to control the inbound queries checking if the policy insertions requested are safe or not. Therefore, it allowed the uploading phase inside the policy DB only in case of a positive result.

1.3. Enhancing 5G SDN/NFV Edge with P4 Data Plane Programmability

A relevant study case is given by this paper [27], where there are illustrated the capabilities of P4 programming language.

With the occurred 5G network generation, new functionalities and a renewed architecture were introduced (it will be debated in Section 2.3), improving various aspects. One of these is the traffic speed thanks to the heterogeneous subsystems. But not all the SDN device are able to handle these type of structures due to its complexity.

From this point of view, the majority part of SDN cannot perform any advanced control, a fine-grained single-packet manipulation, high precision traffic monitoring and high speed services because of the invocations to the SDN controller introducing delay and scalability problems.

To overcome these challenges, a new layer of softwarization was integrated, becoming possible to code the data plane with the usage of APIs and languages. One efficient language is P4, which is described in Section 2.2 and it is being arisen as the deemedest proposal.

P4 switches are very good due to these peculiarities:

- the possibility to define standard and user-defined headers to create proprietary protocols.
- its structure which consists in a not fixed pipeline and controls, that enables advanced tampering operations on attributes of packets and on the control flow.
- the presence of user-metadata to perform custom actions.
- the chance to recreate Finite State Machines (FSM) exploiting stateful objects in order to handle insidious forwarding decisions and in-depth packet manipulations.

In Figure 1.7 is depicted how a P4 switch can be combined with a edge/fog computing infrastructure: the switch can bind multiple different segments such as 5G network, metro-core infrastructures and IoT gateways.

In particular, P4 had introduced various innovative functionalities, enabling new solutions: traffic engineering, telemetry for latency-critical services, 5G VNF offloading and other more.

A prominent utilization of P4 is inside the cyber-security environment: a normal switch

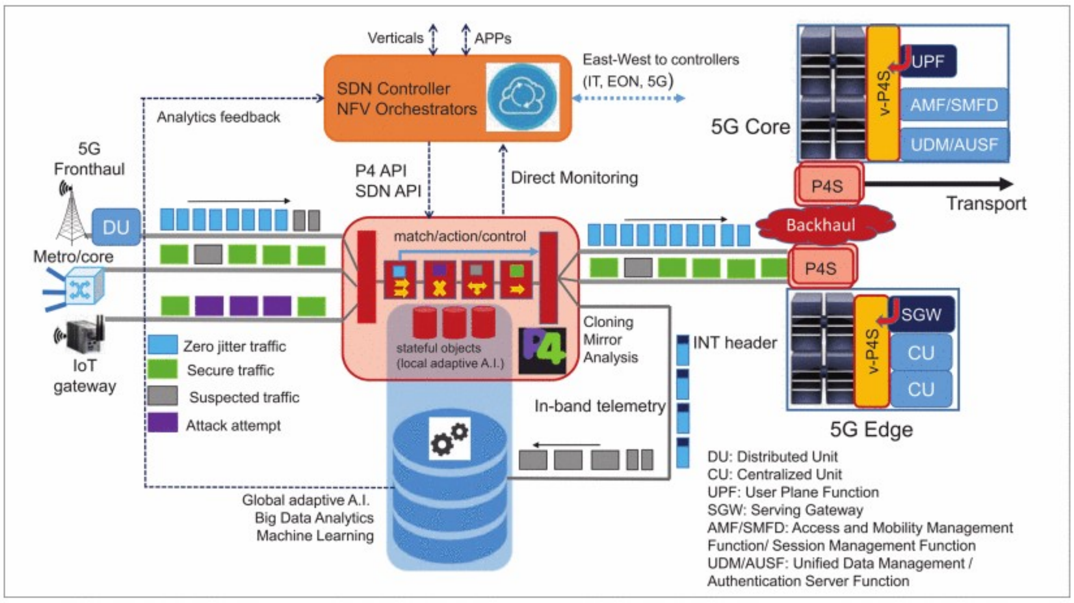


Figure 1.7: P4 switch interfacing edge/fog computing resources and utilization in the 5G landscape ([27])

employed as firewall cannot be developed with the capability to memorize the events and to execute a detailed packet analysis; thus, the packets require one or more SDN controller actions.

With P4 native Deep Packet Inspection abilities, it is possible to replicate secure implementations above every switch inside and, in particular, at the edge of network: a few types of attack can be discovered by applying the stateful capabilities of P4, keeping track of IPs and ports and avoiding malicious overloading attacks like the Denial of Service (DoS) ones.

In addition, the authors of [27] have proved with some performance tests that a P4 switch can dynamically maintain an high speed throughput (around 1 Gb/s with a software target and more than 10 Gb/s in an hardware one) even though it is under a DoS attack.

The outlined work shares with this same project the usage of a SDN component written in P4 language inside a Edge Computing infrastructure.
Specifically, this thesis work takes a similar direction with the described paper. The main focus is to develop an authorization protocol for a Mobile Edge Computing node, exploiting the division between data and control plane on which P4 is based on to achieve a better result in terms of performance.

1.4. Service Function Chaining: Creating a Service Plane via Network Service Headers

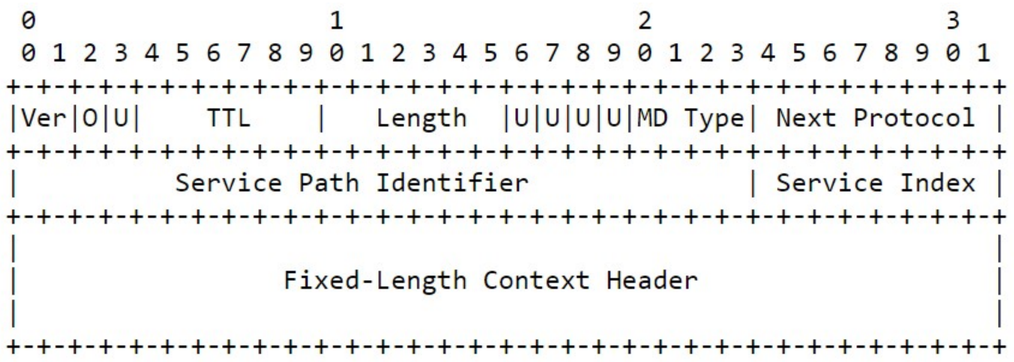
In these decades, the researches were constantly improving technologies and paradigms for every type of standard. Notwithstanding this, there was no lack of security challenges with a special emphasis about the deployment of critical services, which remains a complex and risk-fraught problem.

As suggested in [30], one possibility to address these problems was given by the introduction of Service Function Chaining on behalf of the Internet Engineering Task Force (IETF), detailed in the Request For Comment 7665 ([16]).

Normally, it is not so easy to add a new functionality in a service server because it has to be inserted physically in the data path and this could get worse performance and compromise other functionalities, even if virtualized.

With the advent of the RFC8300 ([29]) in 2018, now it is convenient utilizing the Network Service Header (NSH) to make nimbler the insertion of new services thanks to the characteristic of the NSH to be processed as a resource with attributes that can be exploited also to indicate the service path and to carry on metadata to policy enforcement.

NSH can create a dedicated service plane and it can be inserted inside the packet inside the network layer, eventually under the IP protocol. Its structure is like the one depicted in Figure 1.8: it is built with a base header, a service path header and 16 bytes of mandatory context information and an optional variable-length context data. Usually, it is injected by a network node on ingress into a network.



```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver|O|U|    TTL    |   Length  |U|U|U|U|MD Type| Next Protocol |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Service Path Identifier              | Service Index |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                  Fixed-Length Context Header                  |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1.8: NSH header format ([29])

Naturally, NSH needs to be read by a control plane for sharing the metadata and Service Path Id with the internal network nodes and there is no single mandatory control protocol which has to be followed.

Moreover, NSH makes a service independent from a specific topology providing informations regarding the path to be taken and metadata to be matched with a preexisting policy to access an established service.

This article describes an important concept for the realization of this thesis: the Network Service Header.
This is one of the components used in the designed authorization protocol thanks to the employment of its metadata storage attribute which allows to exchange significant data to enforce policies admitting the packet traffic between client and server. We did not take full advantage of it: we did not apply the possibility to deploy a service function chain (which could still be a potential extension for the project).

2 | Background

In this chapter, the main background concepts of this project are present. They were involved during the design and production phases of our CES component.

2.1. Software Defined Networking (SDN)

One of the major key notions in the project is the SDN component developed to handle the packet traffics. Hereafter, there is a description about SDN and the central characteristics of it.

2.1.1. Definition of SDN

Software Defined Networking is a methodology to define how a network could be implemented and this is becoming more and more important due to its impact on the continuous innovation.
Even if it cannot resolve any technical problems such as routing, congestion control, security and other ones, it can create opportunities to determine new solutions to resolve these and other similar challenges.

As described in [28], the main concept of SDN is to replicate the transformation happened in the market landscape on the router market (Figure 2.1) trying to create an horizontal layer with various network operating systems working on top of bare-metal switches, giving the opportunity to build a rich marketplace of networking application.

The central idea within SDN is to have a network with control plane and data plane divided, where they can communicate towards an open interface (which is accessible also from the external environment to control and check the internal one): in this way, the control plane decides how the network has to behave and the data plane has the duty of implementing that behaviour on the packets.

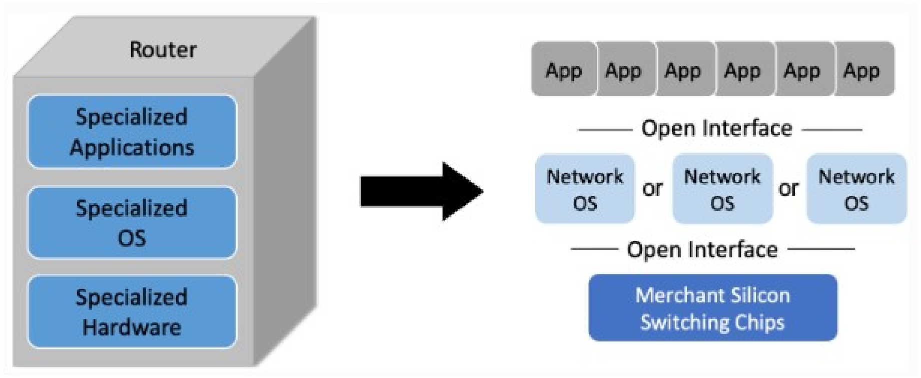


Figure 2.1: Transformation of the vertical router market to a horizontal marketplace with open interfaces and multiple options available at every level ([28])

2.1.2. Control plane

We have two different methods to separate the control plane from the data plane:

- On-switch: this version is called also distributed control plane and it is based on the concept that a switch is to be identified as an autonomous device which needs to interact with the other switches inside the local network to create a local routing table. Therefore, it leads to lower costs and to reduce complexity (only the controller software needs to be installed onto the device), but it is not evolving towards the innovative direction of SDN.

- Off-switch: it is the relatively new concept, which admits an usage of control plane completely independent from data plane and logically centralized, becoming, exactly, a centralized component. Nevertheless, it could be implemented as a distributed system over a few servers having the state of the controller stored in a global data structure shared between them.

The main component of a centralized control plane is the Network Operating System (Figure 2.2) which facilitates the implementation of network control functionalities.

A NOS is based on making abstract all the switch implementations behind the scenes and supplying the application developer with a Network Map.

Thanks to the centralized network abstraction and the rise of NOS, one controller became able to compute globally optimized solutions for every kind of variation on the distributed data plane layer.

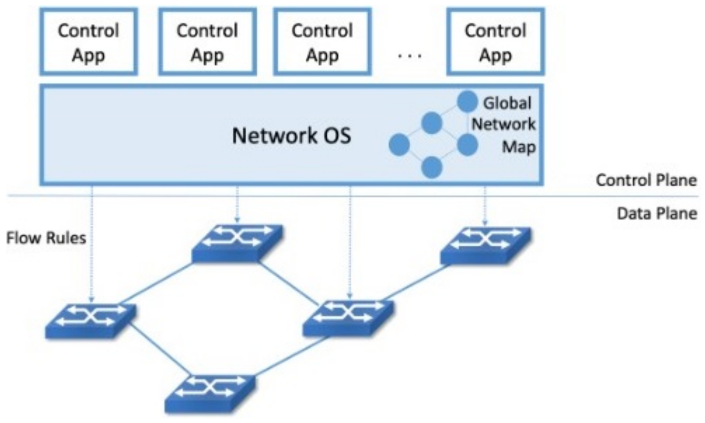


Figure 2.2: Network Operating System (NOS) hosting a set of control applications and providing a logically centralized point of control for an underlying network data plane ([28])

2.1.3. Data plane

The classical data plane behaves in this way: it receives a packet, makes a lookup on the forwarding tables that it has and routes it onto the port refereed by the match table entry. This behaviour could be a reasonable implementation strategy for low-end switches; however, to aim for high-performance, a switch needs a hardware-based forwarding pipeline.

In a forwarding pipeline, a packet is checked towards multiple match table accumulating actions in a set and during the last stage, the switch applies all the actions in order and then it decides if and where forwarding the packet.

As early as the SDN controllers were deployed, two big problems were starting to arise:

- Standards-stucking: in order to have an high-performance switch, the controller demands to know all the structure of the pipeline of the (proprietary) switch to handle it. Consequently, that controller requires to be specialized on a particular implementation of an hardware switch, becoming stuck and limited to that platform.
- The evolution of protocol stack: the protocol stack can evolve in various unpredicting ways. Thereby, also the header fields can be modified continuously, so the switches need to be updated everytime with a new paradigm or adding the possibility to be able to read the new headers.

Recently, the programmable forwarding pipelines and a high-level pipeline language were developed and combined between them to address both of these challenges. These technologies came up under the names of Protocol Independent Switching Architecture (PISA) and P4 programming language. This is one of the main background notions of this thesis and it will be described in the next Section 2.2.

In conclusion, the SDN concept was born to create a centralized control plane to be programmable and it evolved, making possible to include also a programmable data plane.

2.2. P4 - Open Networking Foundation (ONF)

The Programming Protocol-independent Packet Processors (P4) is a domain-specific language for network devices ([9]), which implements the behaviour of data plane devices during the packet processing phase. It was born in 2013, the first official P4 documentation was released in 2014 ($P4_{14}$) and the current release is $P4_{16}$.

2.2.1. P4 workflow

The P4 programs and compilers cannot be used for any device because they are target-specific. In particular, there are two different targets:

- Hardware-based: FPGA or Programmable ASICs (e.g. Barefoot Tofino switch)
- Software-based: software which can run on x86 (e.g. BMv2 switch)

and there are various architectures like V1Model and PSA.

A P4 program classifies packets checking its header and decides what kind of actions has to be applied on incoming packets, routing or forwarding the packet depending on the available layers.

Specifically, the P4 programmer can define ([10, 11]):

- Headers: the headers already present in the literature and in the telco protocol (e.g., ethernet, ipv4, arp, tcp, udp, dns) plus the programmer-defined ones.
- Metadata: the internal packet parameters which are important for the data plane to direct that packet.
- Registers: we can consider they as stateful arrays whose values can be read and written during packet forwarding under the control of the data plane. They could be utilized to implement a particular behaviour which cannot be enforced by meters and counters.
- Counters: they are a mechanism for statistics topics which could be read by control plane and they are associated with a specific P4 table.
- Meters: they are more complex mechanisms for keeping statistics about packets.

They are used for dropping or marking packets that exceed an average packet or bit rate.

- Match-and-Action tables: tables which describe a match-action unit. They perform a key-match on the packets in order to compute some actions.
- Apply sections: they are sections inside every control function of the data plane that can say which match and action table needs to be applied.

2.2.2. P4 compiler

Probably the main component of P4 language is the reference compiler called p4c ([12]). P4c is modular, it ensures a standard frontend and mid-end which can be bring together with a target-specific backend to create a complete p4 compiler.

As outlined in Figure 2.3, a P4 compiler can generate a few file resources:

- .P4info: this file represents the run-time mapping metadata. With this, the control plane and the data plane can communicate between them with the help of P4Runtime.
- .bin: this is an executable for the target data plane, which contains the header formats and corresponding actions for the target device.
- .txt: it is possible retrieve a text format of .P4info file by adding an option. It describes the tables and other object in the P4 program that have an auto-generated control plane API.

2.2.3. $P4_{16}$ Data Plane model

The data plane of a target device is thought like a Protocol-Independent Switch Architecture as illustrated in Figure 2.4.

As described in the P4 language tutorial ([8]), PISA incorporates three basic components:

- Programmable parser: it is modeled as a simple finite state machine, that checks packet data and identifiers headers. The transitions between states are typically made by controlling at specific fields in the previously identified headers.
- Programmable Match-Action pipeline: this pipeline is thought as a set of stages which compares a certain amount of data of packet respect to various tables containing some entries. After these controls, some corresponding actions are applied on the packet.

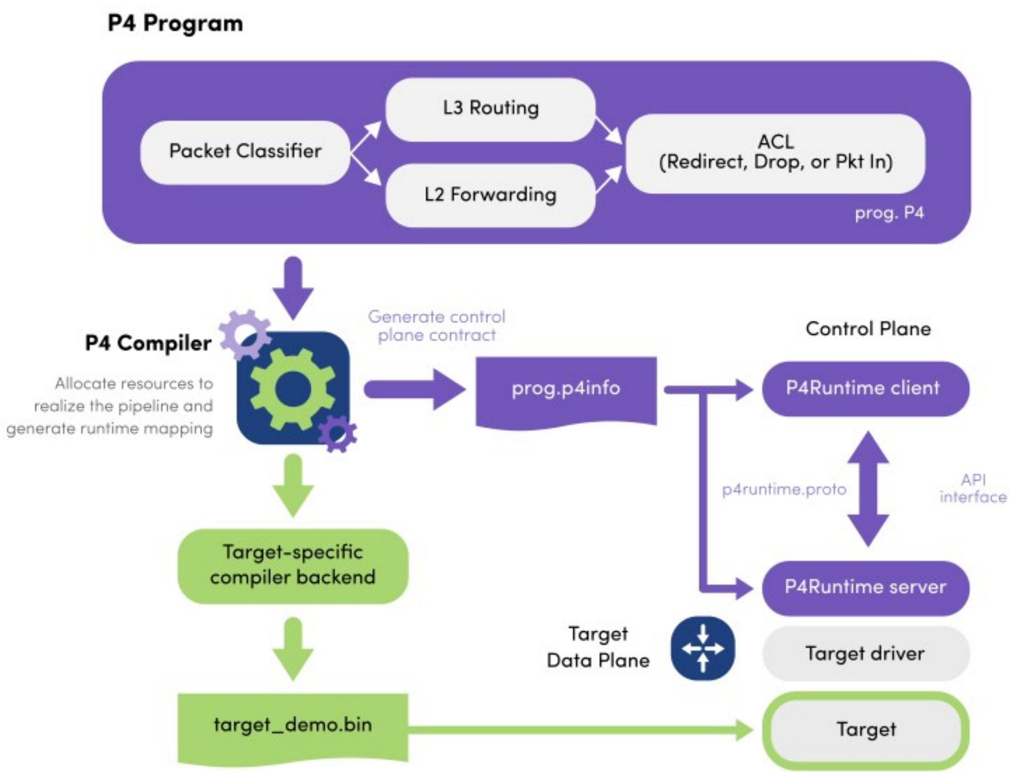


Figure 2.3: The complete workflow of a program and a compiler in P4

- Programmable deparser: it reassembles the headers of packet serializing its, in order to send back the packet onto the wire.

P4 programs are composed of various different components: parsers, controls, expressions, data types, architecture description and extern libraries.

Parsers and controls are the core programmable blocks that a programmer could describe in P4: parsers are intended as a state machine and they aim to extract header fields from the packet in a programmer-defined way. The term controls refers to tables, actions and control flow statements which are responsible for the principal match-action processing. Inside these components, the programmer can describe how packet headers and metadata are manipulated. They are built using the various data types and expressions defined in the language.

But there are not only constructs predefined by P4 language: the architecture description and the possible extern libraries have to be detailed by the vendor of those structures. The vendor needs to specify the configuration of programmable parsers, control blocks, and, eventually, specific blocks.

They may also offer an extern library which provide some functionalities that are not

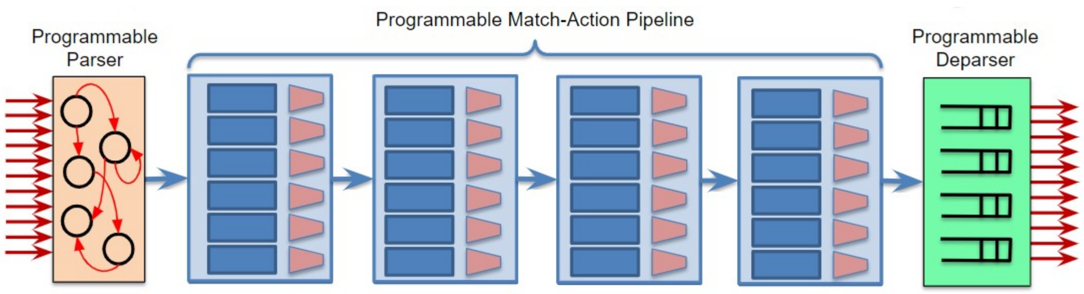


Figure 2.4: The Protocol-Independent Switch Architecture considered in $P4_{16}$

implemented in P4, but an interface needs to be created and exposed by the seller; in a way that it can interact with or be invoked by P4 programs like an API. These extern functionalities may be target-dependent, but different targets could also support those interacting with the same interface, which improves portability of P4 programs.

2.2.4. P4Runtime

At the moment, the existing approaches to run-time control are these ones:

- P4 compiler auto-generated run-time APIs: they are program dependent and are hard to provision new P4 program without restarting the control plane.
- BMv2 CLI: it is program independent but target specific, in particular the control plane is not portable.
- OpenFlow: it is target independent but protocol dependent, the protocol headers and actions are bounded to the specification.
- OCP Switch Abstraction Interface (SAI): it is target independent but protocol dependent.

The Table 2.1 depicts a summary of the properties of these run-time control API:

API	Target-independent	Protocol-independent
P4 compiler auto-generated	✓	✗
BMv2 CLI	✗	✓
OpenFlow	✓	✗
SAI	✓	✗

Table 2.1: The properties of the existing run-time control APIs ([8])

The P4.org API Working Group has established the last P4Runtime specification [13] at the end of 2020 to regulate vendor-independent, protocol-independent run-time APIs for the data planes implemented in P4, overcoming the problem of all those APIs: the impossibility to be target and protocol independent at the same time. This is an API for P4-defined/described data planes and it is designed to be the conjunction between control and data plane which needs to be written in $P4_{16}$.

P4Runtime API is an open source framework for run-time control of P4 targets defined by a protobuf-based API (p4runtime.proto) and gRPC server (a program-independence server), as depicts in Figure 2.5, and they most interesting property is the field-reconfigurability: it can upload a new P4 program without recompiling the software stack of target switches.

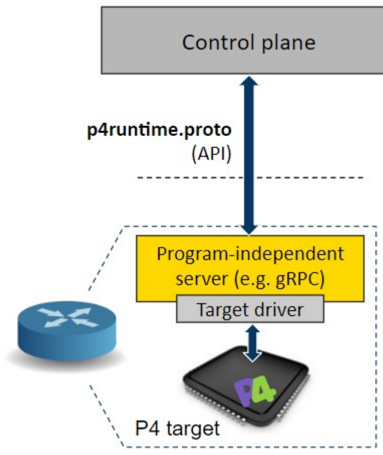


Figure 2.5: The P4Runtime components related to BMv2 behavioral model ([8])

In practice, P4Runtime allows a local or remote entity to upload the pipeline/program, send/receive packets, read and write forwarding table entries, counters and other features.

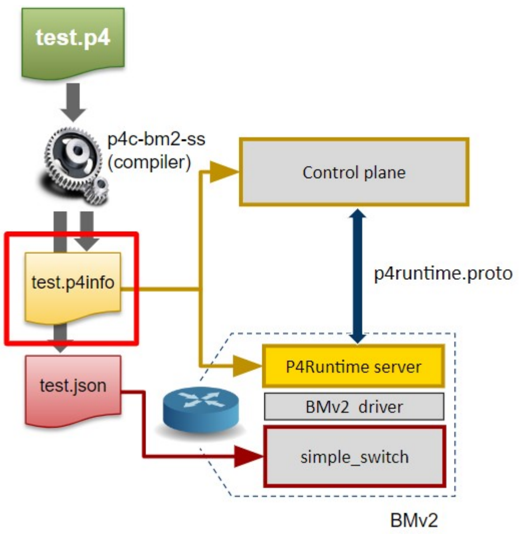


Figure 2.6: P4Runtime workflow ([8])

The gRPC server and the control plane need to agree on the message format for communicating. This format is given by the file P4info which is produced by the P4 compiler. This file includes the description of all the P4 program attributes required for run-time control, it is similar to a contract between control and data plane, it is target-independent and its format is based on protobuf ([14]).

2.3. 5G Networks

2.3.1. Introduction

5G technology is the current last technology standard for broadband cellular networks which has been used worldwide since 2019 and it is considered the heir of the 4G networks.

There were previous generations before 5G. 1G and 2G were focused on audio transmission; in particular, the first was analog and 2G was the very first one technology completely digital. With the advent of 3G generation, there were some improvements regarding the data transferring (up to hundreds of kilobits per second) and voice quality. The precursor of 5G is the 4G network. This standard had introduced a new peak for the download and upload data transfer (dozens of megabits per seconds) and extended the application domain.

5G brings with it two major improvements: an higher download speed (up until 10 gigabits per second) and an higher bandwidth which can allow more different device to be connected increasing the quality of Internet services in areas where there is an high density of people (which implies more mobile systems).
In addition, with this new technology there was a huge impact for what concern the architecture of the entire network, specifically for accessing phase to the network.

5G is not only a new generation for broadband cellular networks but also a technology which has been able to bring a strong innovative concept, the Mobile Edge Computing, which will be described in Section 2.4, and a distinguished quote of the authors of [21] makes a parallelism with 3G to explain that pioneeristic idea:

> In the same way that 3G defined the transition from voice to broadband, 5G's promise is primarily about the transition from a single access service (broadband connectivity) to a richer collection of edge services and devices.

Because of this new paradigm, the application domain of 5G is very large and it is possible finding fields like immersive user interfaces, mission critical applications and Internet of Things among those applications.

5G enables not only private usage but also an industrial appliance such as Industrial Internet of Things (IIoT), supporting humans and waves of independent devices collaborating with each other by themselves.

2.3.2. Architecture

As illustrated in Figure 2.7, the mobile cellular network is made by two principal entities: the Radio Access Network (RAN) and the Mobile Core.

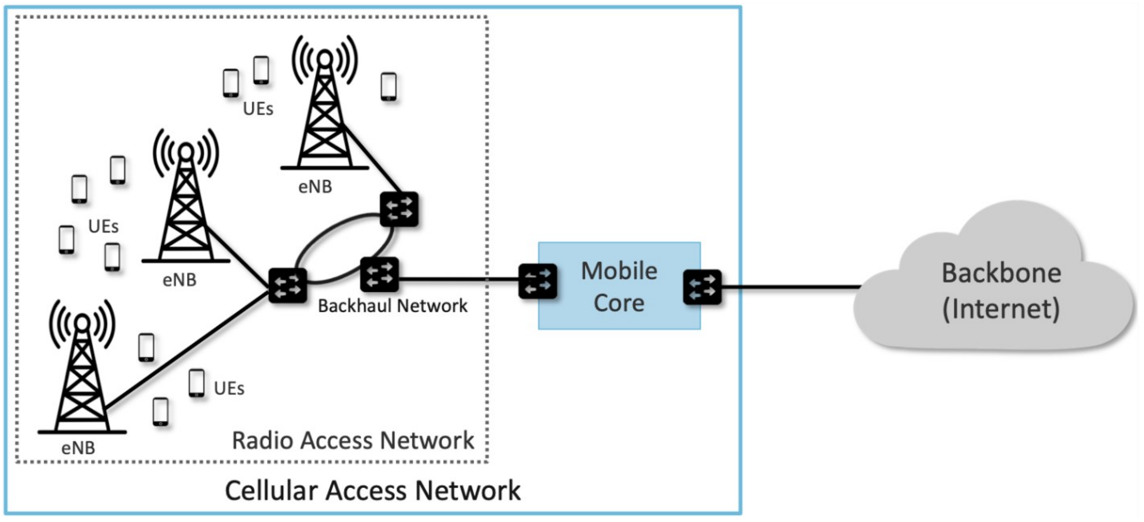


Figure 2.7: Mobile cellular network ([21])

The RAN controls the radio spectrum, ensuring that it is used efficiently and it fulfills the quality of service requirements of every user. It has to be considered as a distributed group of base stations, which are named eNodeB or eNB (which is short for evolved Node B) in 4G, while in 5G, its are classified as gNB, where the "g" stands for next Generation.

The Mobile Core supplies IP connectivity to the RAN. It authenticates UEs when they connect, traces them as they change base station due to moving between two different areas without interrupting the services, makes certain that this connectivity meets the promised QoS requirements, and meters usage for billing.

It could be seen as an architecture composed by microservices as shown in Figure 2.8 and the main ones are:

- UPF (User Plane Function): it is in charge of forwarding traffic between the RAN and the Internet, it is also responsible for policy enforcement, lawful intercept, traffic usage measurement and QoS policing.
- AMF (Core Access and Mobility Management Function): this component is ac-

countable for connection and reachability management, mobility management, access authorization and location services.

- SMF (Session Management Function): it handles all UE sessions, including IP address allocation, selection of associated User Plane function, control aspects of QoS and UP routing.
- AUSF (AUthentication Server Function): it authenticates the UEs.
- UDM (Unfield Data Management): it handles user identity and generation of authentication credentials.
- PCF (Policy Control Function): it manages the policy rules for the rest of the Mobile Core CP.

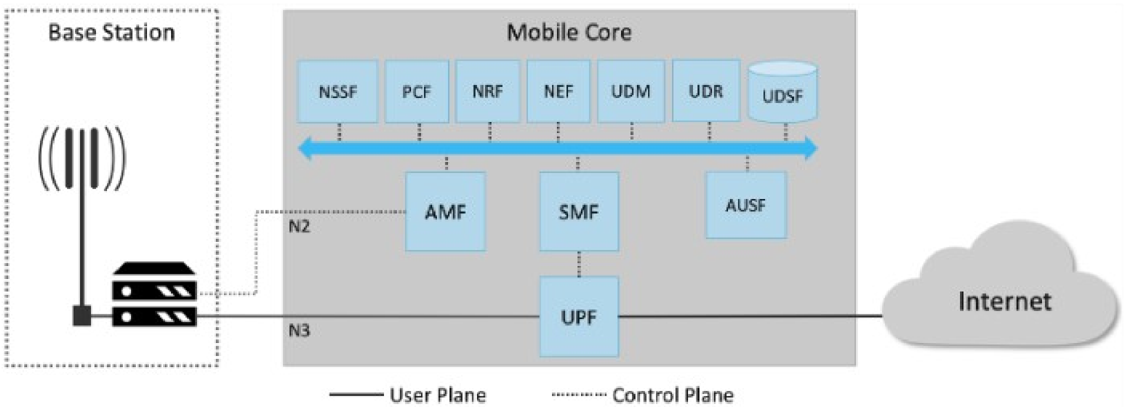


Figure 2.8: G Mobile Core (NG-Core), represented as a collection of microservices ([21])

The high frequencies of 5G carries with it the poor signal propagation characteristic, resulting in potential problem for a cellular network that provides coverage for a wide area. Therefore, as described by the authors of [7] and represented in Figure 2.9, the natural evolution of the actual cellular technologies is the heterogeneous paradigm with multi-tier coverage utilizing small cells with high throughput.

2.3.3. Privacy

With the arrival of 5G networking and the mobile phone usage for an easier access to personal sensible data, some security challenges have been rediscovered; in particular, the concept of privacy protection and objectives as reported in [22].

Bringing together 5G network and the new technologies (e.g. Cloud Computing) some new privacy problem were arisen. In Table 2.2, the main 5G privacy issues generated by SDN, NFV and CC are reported. Hereafter, there are listed the relevant ones:

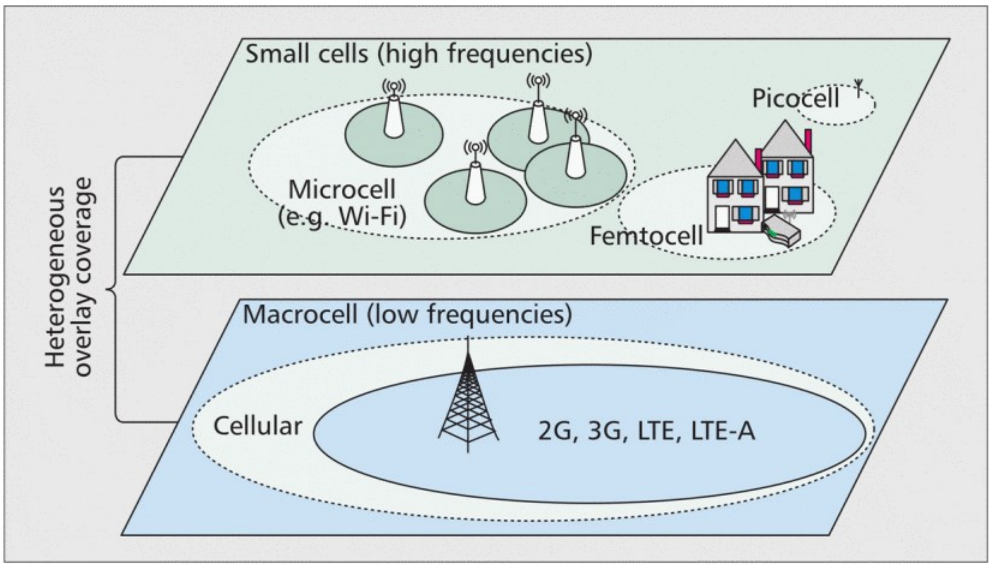

Small cells (high frequencies)
Picocell
Microcell (e.g. Wi-Fi)
Femtocell
Heterogeneous overlay coverage
Macrocell (low frequencies)
Cellular
2G, 3G, LTE, LTE-A

Figure 2.9: 5G heterogeneous network structure with densified small cells and overlay coverage ([7])

- End to end (E2E) data confidentiality: inside a 5G network there are multiple providers and operators which have access to sensible user data without their explicit approval. Therefore, it needs one or more secure mechanisms to provide end to end data confidentiality.
- Loss of data ownership: between end-user and application server there are a multitude of players and third parties which do not have a clear role regarding their responsibilities. As a consequence, the property of user data has to be determined between all the actors by using a privacy enabled service agreement.
- Different objective for trust: since that not all the participants could share a common security or business goal, they should not cooperate to provide the whole amount of security and privacy assumptions.
- Hacking: like the last standard, 5G is an IP-based open architecture, which means that those networks are vulnerable to the full range of IP and web-based security attack, including hacking which can cause relevant privacy problems to the users.
- Providing information for third party: within 5G there are various subsystems which are handled by different third parties. They may exchange some user informations with other parties exploiting their access privilege. Moreover, the dynamic nature of applications inside the 5G network can rise up inconsistencies problems between access rights.

	Promote digital market	Balance of interests	Privacy legislation in global context	Foster interoperability and data portability	Applicable law must be easy to define	Right to erasure and rectify	Increased responsibility and accountability
End to end data confidentiality	✓	✓	✓	✓			✓
Responsability ambiguity and Data ownership	✓		✓		✓	✓	✓
Bylaw conflict and Location of legal disputes	✓	✓	✓		✓	✓	✓
Shared Environment		✓	✓	✓	✓		✓
Different objectives for trust	✓	✓	✓	✓	✓	✓	✓
Loss of governance and Loss of control	✓	✓	✓		✓	✓	
Service provider lock-in	✓			✓			✓
Visibility	✓	✓	✓	✓	✓	✓	✓
Trans-border data flow	✓	✓	✓				
Hacking			✓		✓		✓
Providing information for Third party	✓					✓	✓
IoT privacy				✓		✓	✓

Table 2.2: Summary of privacy objectives with privacy issues of high impact and relevance ([22])

As well as these matters, there are possible mechanisms to protect the privacy of 5G networks ([22]), inter alia, schemes including SDN and cloud:

- Privacy-aware Routing Mechanisms by using SDN: the utilization of SDN components in 5G networks allows the definition of traffic rules that make possible a transition of packets exclusively inside the national boundaries, increasing the privacy of final users.
- Hybrid Cloud approach: the purpose of this method is storing sensible data on an in-house cloud to move locally the majority part of the elaboration, making less usage of public cloud. In this way, there is more control regarding the data shared with extern entities. The Mobile Edge and Fog computing role inside 5G network promotes the usage of this mechanism.

2.3.4. Micro-segmentation

With the introduction of the SDN and NFV, it is possible to increase the security of 5G network. Software Defined Network technology are applied for monitoring and orchestrating networks taking advantage of dynamic network isolation. The Network Function Virtualization concept employs the virtualization of mobile network entities and functions to improve the security.

Network Slicing is one of the theoretical principle which is used in 5G networks. It can be defined as a logical instantiation of a physical network with all the needed functionalities which are required for running a specific service. The slices are a portion of network isolated and restricted to the specific assigned resources ([24]). The Virtual Private Networks (VPNs) are a basic version of NS.

In 5G the micro segmentation consists into dividing the network is multiple isolated parts like in network slide but with a more fine-grained isolation and segmentation with specific access controls. These are built above a few security policies grounded in unique trust models of application services.

The micro-segmentation can be utilized for the implementation of trust policies such as Network Domain Security for IP (NDS/IP) and the Zero Trust model (ZT) and for AAA (Authentication, Authorization and Accounting) functionalities.

2.3.5. Authentication

The introduction of small cells arose several potential challenges including a security problem regarding how the access points can authenticate users in the initial phase. Due to the volatility of mobile phones, 5G users could move from one cell to another introducing too much handover-induced latency in 5G.
Consequently, users and access points need to carry out a mutual authentication more frequently as countermeasure for personification and man in the middle (MITM) attacks.

A possible solution is represented by the inclusion of SDN components inside the network. This leads to an easier method to implement security protocol, network policy including the straightforward network management thanks to the division between control plane and data plane, separating the control logic from the switches and routers into the centralized SDN controller.

One example is given by [7], where the authors designed a SDN-enabled authentication handover mechanism. They developed a module in a SDN controller which can anticipate and supervise the location of users and, consequently, setting up the relevant cells before they can move themselves ensuring a seamless handover authentication.

Another one is given by this same project, because of one of the major security consideration done consists in the policy-based authentication method, which follows the previous approach.

2.3.6. Next generation: 6G

The 6G will be the subsequent generation to 5G and it is currently under development. It is designed still for wireless communications and based on cellular data network which enables the division of the service area into small geographical areas called cells, allowing a faster data transmission and much more heterogeneous small areas.

As illustrated in [18], the 6G technology is estimated to progress in green networks, providing an higher quality of service and energy efficiency. This is going to impact the current mobile usage domain bringing up more scenarios such as virtual and augmented reality (VR/AR) and pervasive intelligence in the Internet of Things (IoT) which could introduce the vehicle to vehicle communication (V2V).

2.4. Mobile Edge Computing (MEC)

The term Mobile Edge Computing (MEC) was firstly showed up in a report of IBM and Nokia Siemens Network ([19]) where they announced that their new platform could run applications within a mobile base station at the edge of the network in 2013.

A year later, the Mobile Edge Computing technology was introduced by the ETSI ISG on MEC ([17]), exposing the cloud-computing capabilities inside the Radio Access Network (RAN) and an information technology service environment, the Mobile Edge servers, at the edge of the mobile network, closely mobile subscribers. The goal of this technology is reducing the network latency, increasing the efficiency of the network operation and service delivery and improving the user experience.

The MEC was built upon the concept of virtual platform, enabling the applications running on the edge of the mobile network and extending the previous architecture of the Network Functions Virtualization (NFV) technology as much as possible, in order to have a common base structure to reuse with multiple benefits.

This leads to a revolution which involves not only the services for the consumers but also for the industries.

The market drive of MEC consists in 4 different types: business transformation, technical integration, industry collaboration and many "minor" use cases such as IoT, health, caching services and many others (Figure 2.10).

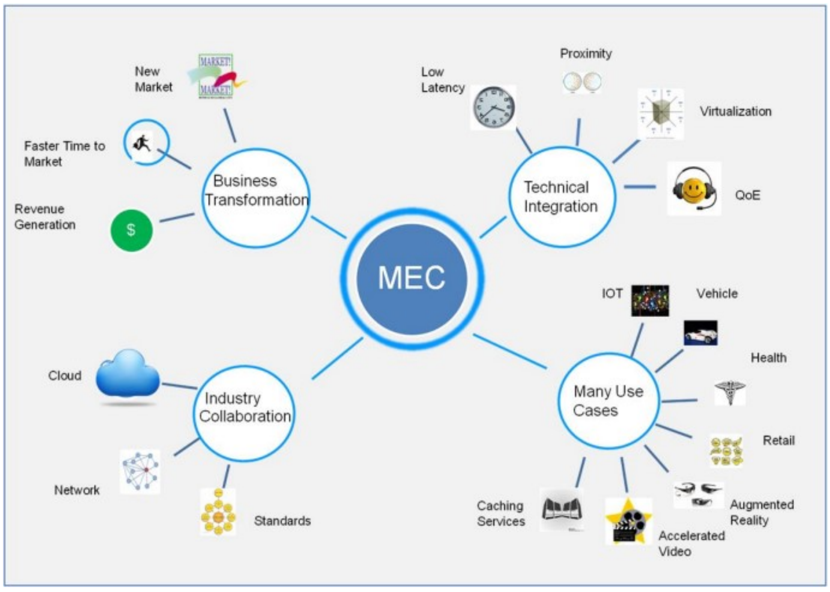


Figure 2.10: Mobile Edge Computing market drive ([17])

2.4.1. The security in MEC

The introduction of an new technology takes with it challenges and problems to engage; in particular, the security topic is one of them, the focus on threats and what type of threats that could be arisen.

In this article [31], there is a concrete report about the security not only for MEC technology but also for all the Edge Computing. There are many reasons for considering the security in these paradigm:

- The presence of multiple different network inside the core of most edge paradigm needs to be orchestrated in order to handle all the different security mechanisms.
- There are a few studies for security of an entire system making it a severe warning. In particular, once a EC paradigm is collocated around the border of a network, it could create new situations which can compromise the other security mechanisms. In addition, it has to take into account the presence of mobile devices which can access to the edge data centers in every moment from everywhere.
- Furthermore, all the system inherits the security threats of every architecture or component within itself, extending the previous attack surface.

For what concern MEC paradigm, there usually is one single company which owns not only many edge data centers installed at distinct geographical locations, but also part of the core networks that is connected to these data centers.

This infrastructure has to be monitored constantly with a security policy for the authentication checks and protected by physical attacks. Thanks to the centralized control, the attack surface should be smaller, reducing the possibility of being compromised.

One of the security challenges appeared with the EC, is the security regarding the protocols and networks. The first aspect of this area is the distribution of credentials for negotiating the session key. At the moment, one of the solution to resolve or mitigate it is to trust in a certification authority which can distribute the credentials to all the trusted element present in a known domain.

The second is the security of the virtualized network infrastructure. One way to overcome this aspect is using both SDN and NFV which could be useful in many manners such as isolating different types of traffic, isolating insecure network devices redirecting the traffic towards other security components of the network and reconfiguring the system in real-time.

3 | Project Description

The scope of this chapter is to illustrate the starting concepts for the development of our Customer Edge Switching (CES) component, which implements a policy-based authentication and authorization protocol at network layer.

3.1. Scenario

A mobile phone user requires to access a service collocated in an extern network. Between these two entities there is a 5G MEC environment similar to the one described in Sections 2.3 and 2.4 and depicted in Figure 3.1.

Usually, there are some secure end-to-end security mechanisms in the application layer, like TLS authentication and packet encryption. Although they are well-tested concepts, this kind of security cannot prevent internal attackers or attackers, that compromised legitimate nodes, from launching newer attacks towards network resources. This is known as lateral movement.

In this work, we present a network-layer security model which works mainly on the first layers. This solution also makes possible to improve the security of legacy and IoT devices that are hard to patch regularly and updating these ones would result in a problem.

The 3GPP[1] consortium released the 5G NR world standard [6] describing how the packet traffic has to flow inside a 5G network (as described in the Section 2.3). In particular, it states that a packet of a user has to be sent to the User Plane Function (UPF) node at border of the 5G network and then it transits towards the internal network of the edge node.
Here, the packets are checked by the CES component in order to control if the traffic is authenticated or not. Subsequently, if the authentication check is matched and the authorization phase is passed, the packets can access the service entrypoint interacting with it.

[1]The 3rd Generation Partnership Project

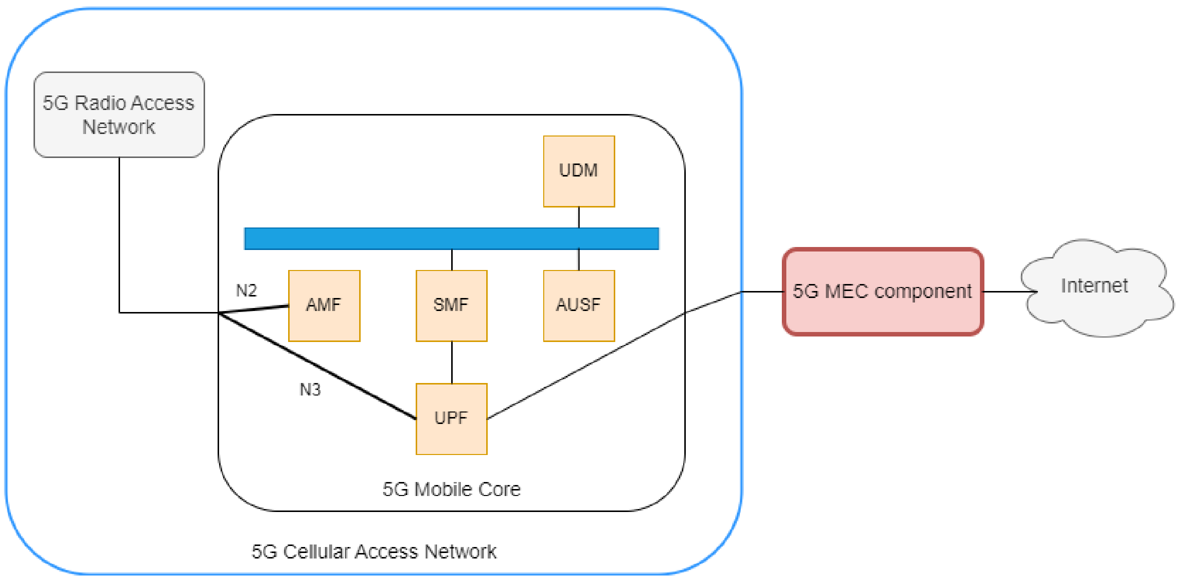


Figure 3.1: The 5G MEC environment assumed in our scenario

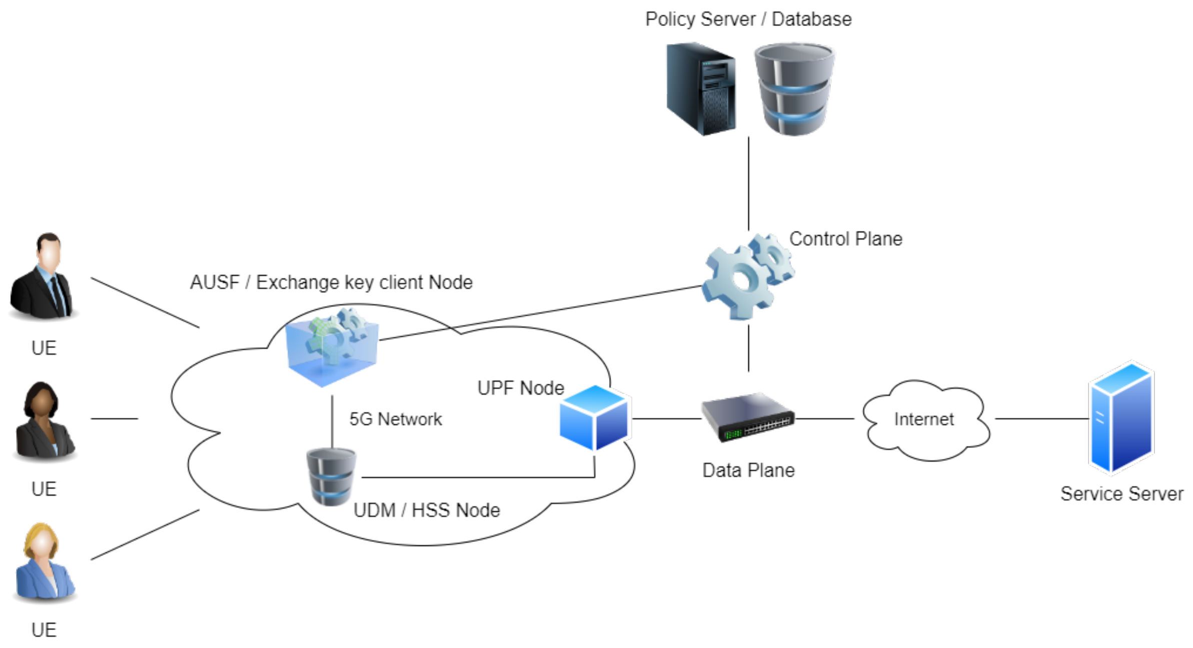


Figure 3.2: The reference architecture of our project

3.2. Architecture

The architecture of this project is represented in Figure 3.2 and it is structured using the following entities:

- 5G Network: this part of architecture is deployed like the standard cases proposed previously in Section 2.3. For what concern our project, these structures are particularly relevant:
  - AUthentication Server Function (AUSF)/Exchange key client Node: it is a component which needs to negotiate the master key with the control plane and then stores it in the UDM/HSS node inside the 5G Mobile Core.
  - User Data Management (UDM)/Home Subscriber Server (HSS) Node: This server saves customer profile data and authentication information along with encryption keys.
  - User Plane Function (UPF) Node: it is the last node of the 5G network where the packet traffic flows.
- CES component: it is an agglomeration of more sub-components:
  - Data Plane: it is a switch which receives the packet traffics forwarded from the 5G network and screens the internal network from malicious packets.
  - Control Plane: this is an orchestrator which controls the data plane, following the SDN paradigm, querying the policy server and implementing the security model described in Section 3.3.
  - Policy Server/Database: it is a server which is accessed by the controller to retrieve the access point of the requested service and checking the current security policies stored.
- Service Server: the entity which employs the CES component is in charge to build this server. The services have to be developed with the Function as a Service (FaaS) paradigm.

3.3. Security model

Our project is based on the Zero Trust Security (ZTS) principle ([15]). Therefore, every entity is not automatically trusted, even if it is inside the internal network perimeter.

The central concept behind this security model is "never trust, always verify". So, every entity has to provide an authentication evidence to gain access.

This principle is different from the traditional security approach due to the methodology used by the system to release the level of privilege and the attributes of a resource to be considered for the purpose of classify that resource safe.
In the classical security model, the type of privilege can vary and every source inside the network perimeter is considered trusted.
With the ZTS principle, no source is secure and when access is granted, the level of privilege given is the least amount necessary to complete the request.

For the aim of determining if the access to a service can be granted, the system needs to apply security policies based on various parameters such as the attributes of the data, the user entity and the type of environment using Attribute-Based Access Control (ABAC). This zero-trust data security approach should prevent any malicious access to data as described in [37].

The network architecture has to be defined in such a way to be protected against the lateral movement network attacks provided by a user which has obtained an access permission. This kind of offense is severe, because, exploiting the previous authentication, an attacker could explore all the network searching for sensible data and try to gain access through a privileges escalation attack to other services to which they should not be able to be admitted.

To avoid this behaviour, there are various solutions. One of these is the micro-segmentation technique, which consists in dividing all the services in different components or units in order to force the user to be authenticated for a service to get access to that particular one.

Another way of doing that is instantiating the services taking advantage of the Function as a Service (FaaS) paradigm ([23]). In this way, the server needs to destroy the instance previously created once the usage of that instance has been ended.

3.3.1. Description of possible attackers

Our component and protocol can be subject to an attack performed by an entity just like every other information system.

Our assumption on the 5G network is that every component is working without any malicious intention. In particular for the 5G network:

- The AUSF node negotiates adequately the informations regarding the users without emulating any malevolent behaviour of an attacker.
- The UDM component stores the master keys of each user without favouring any attacker.
- The UPF node retrieves the packets and injects the authorization tag only in the first phase of the TCP three-way handshake of a connection, without manipulating them on behalf of an attacker.

For what concern our CES component, it cannot be corrupted; i.e., it cannot be manipulated redirecting the packets in a way to satisfy the attacker's will. The policy server can dialogue only with the AUSF node in the 5G network (assumed as not malignant) and with the service server which is considered to be working with the right behaviour.

We are also assuming that the service server is instantiating the services using the FaaS paradigm and that the cryptography between client and server is ideal.

With regard to a possible attacker, there are two types of them: an extern attacker and one which is inside the intranet-work.
We are considering both of them with unlimited power computation.

The insider is a user which had compromised one or more services and wants to extend the assault to other services present in the system. They need to result authenticated for a service (for which they are not) and take control of that machine.

The external attacker could be an outsider compromised machine used to intercept a packet of a TCP connection in order to retrieve an authorization tag for guessing another tag to simulate the connection of another user or to retrieve some informations from a packet with the purpose of breaking the user's privacy.

4 | Project Implementation

In this chapter, it is outlined the implementation of the building blocks of this project: data and control plane, the developed protocol, the policy management and some considerations regarding the properties of this model.

4.1. Protocol Description

This authorization protocol was built considering two core phases and an out-of-band phase which is aimed to resolve the challenge regarding the knowledge of the credentials of service server. The main concept behind this protocol is the authorization of a packet traffic by taking advantage of a tag inside the network layer.

4.1.1. Service discovery phase

In this phase, the device needs to receive the service name and the access point credential of the service, which they would access somehow. This challenge is considered out-of-band; so, in our assumptions when the user wants to be admitted to a service, they already know the entrypoint of server. The diagram of this phase is illustrated in Figure 4.1.

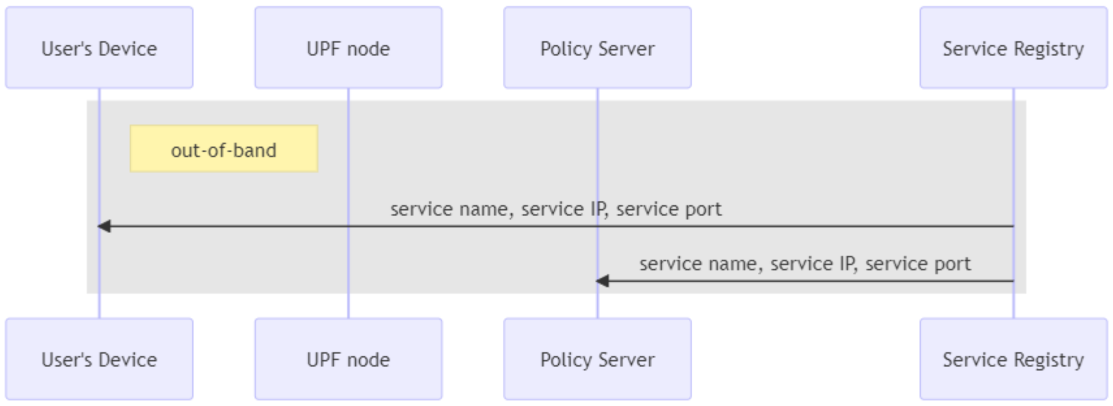


Figure 4.1: The out-of-band service discovery to retrieve the access point and name of service server

4.1.2. Key exchange phase

Our first designed part of protocol is concern the master key exchange between the authorization node and the control plane because the assumption for the authorization phase is that both these entities share a symmetric key.
Normally, this negotiation is accomplished between the control plane and an entity inside the 5G network which has the role to handle the authentication and authorization attributes for the users, acting as an intermediary between users and controller. From literature ([6]), we know that the AUthentication Server Function node (AUSF node) assumes this behaviour inside a 5G network.

We have considered that, from now on, the controller knows the MAC addresses of any actor, for the sake of simplicity.

When the AUSF node receives a first open TCP connection from a user, it has to derive from that packet the name of service and imsi of the sim card presented in the user's mobile phone. After that, it needs to exchange a master key for that user, between itself and the control plane of SDN component.

This trading can be actuated with any key exchange algorithm. For our case, we have selected the Diffie-Hellman algorithm because it is one of the most used. Therefore, the AUSF node sends to our control plane: imsi, name of service and the chosen parameters for DH. At this point, the control plane must check inside the policy server if that imsi have been authenticated for the requested service, searching if a service with that name is exposed and the imsi is authenticated for that service.

Right after this control, if a service with that name does not exist or the imsi cannot obtain the permission due to it was not previously authenticated, the TCP connection request is rejected by the control plane; else, the control plane extracts various attributes from the packet to elaborate the rules which will be inserted inside the tables of data plane, after the computation of the master key and the first HMAC. Thus, the control plane computes the DH parameters to be sent back and returns them to the AUSF node.

Now, the AUSF node stores the master key inside a storage node called User Data Management. In our project, it is implemented as a file from where it is possible read and write this key and an associated counter for when the control plane needs to retrieve the next HMAC for another connection of that user.

Hereafter, in Figure 4.2 is represented this part of protocol.

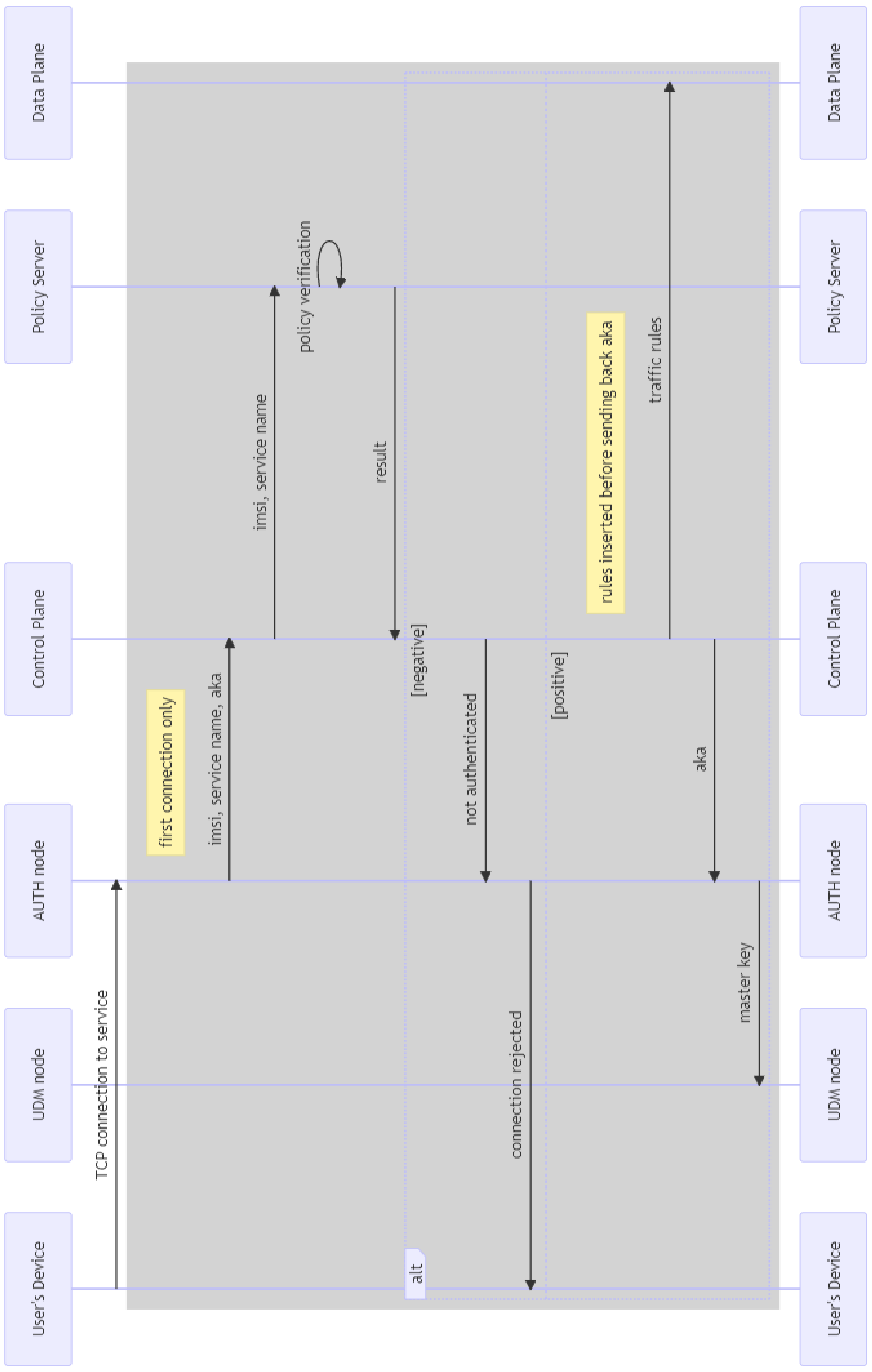


Figure 4.2: Key exchanging phase between auth node and control plane

4.1.3. Authorization phase

For what concern this phase of the protocol, we had assumed that the session key was already traded in the previous phase and that the MAC addresses of service server and UPF node are known by the controller.

When a user is trying to reach a service, their packet is intercepted by the UPF node because it is the last node of the 5G network. So, from now on, the intermediate actor will be the UPF node.
Obtained the first TCP connection of a specific packet traffic, in order to retrieve the master key and the current counter for the user from whom that traffic is originated, the UPF node needs to interrogate the UDM node, which in our case is implemented as a set of files, one per each user.

Subsequently, the UPF node must add inside the previous packet, an authorization tag. This tag is computed by using a hash function which has to consider: the parameters read on the UDM node regarding that user and an object with all the attributes related to the user and their packet traffic.

The hash function which we have selected is Shake-128, which is a subset of the cryptographic primitive family Keccak. The Shake family hash function allows an arbitrary output length, it is faster then the SHA3 hash function family but it has a preimage resistance property which is half of the one of SHA3. Anyway, the Shake family maintains an equal collision resistance property, which is the one we are interested in.

After the production and the injection of this authorization tag inside the network layer of the packet, the UPF node sends it to the data plane. There, that packet is inspected verifying the tag and in case of a negative result the packet is dropped resulting in a malicious one (this case is not reported in Figure 4.3). If there is a match, a clone of that packet is sent to the controller to update the temporary rules inside a match-and-action table while the original packet is forwarded to the service server.

Down there, the server replys to the request of connection by sending a TCP ACK. This one goes through the data plane and then comes to the UPF node which will forward it to the user's device.

From this point, there will be a TCP connection between user, UPF node and service server where the user's device transmits a packet which will be received by UPF node, then by data plane and, in the end, by the service server. The return package travels the same route but in reverse.
The diagram of this protocol phase is depicted in Figure 4.3, at the next page.

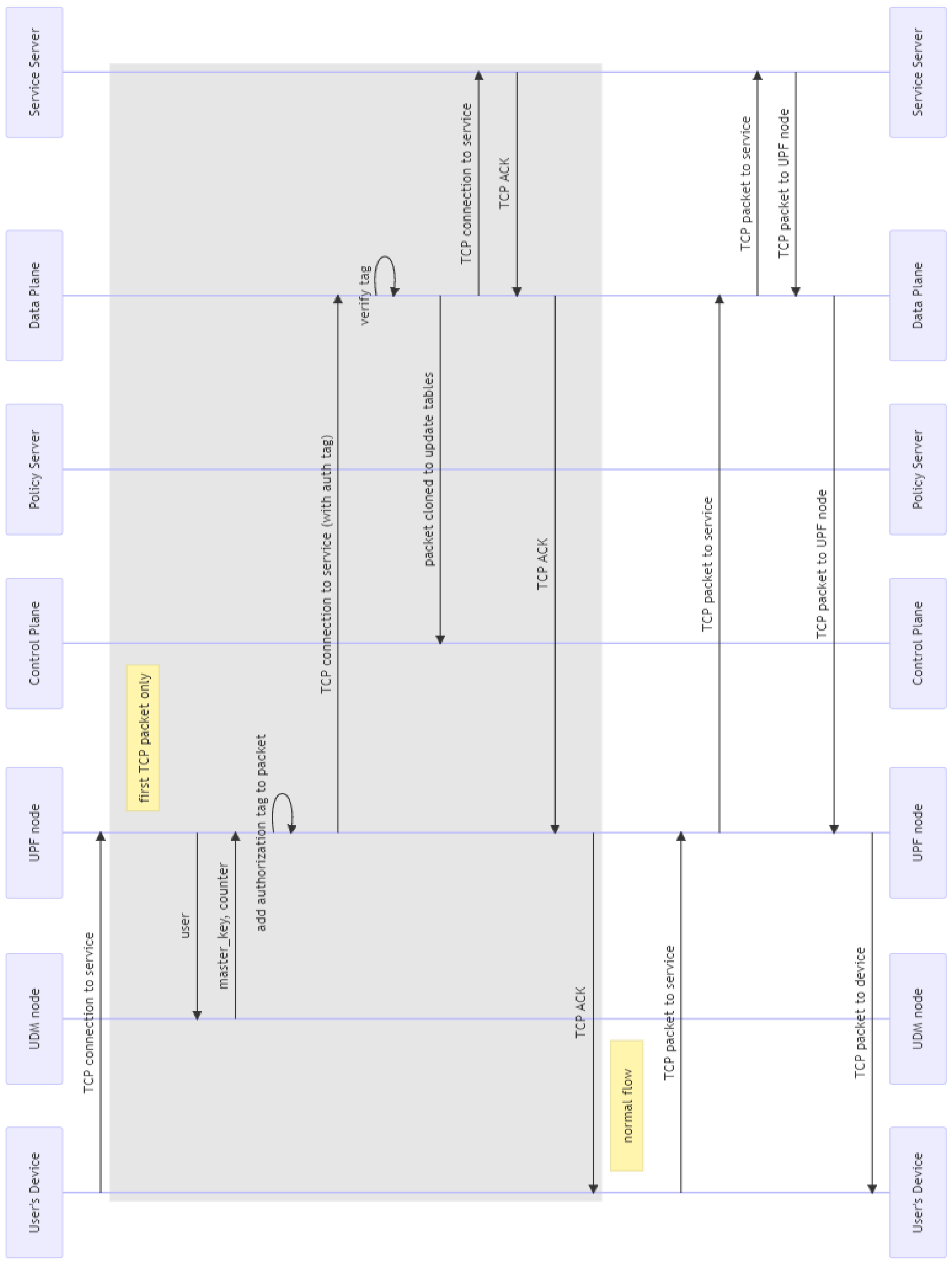


Figure 4.3: Transition of first packet with authorization metadata to authorize that packet traffic

4.2. Data Plane

In this project, the data plane was designed with only one switch but the architecture can be extended taping to SDN paradigm with multiple switches and a load balancing methodology to decrease the load of packet traffics on each of them.

Our P4 data plane is structured for v1switch architecture and it is thought as a two phases packet checker.

4.2.1. Already authorized phase

The first phase is dedicated to check if the packet received is regarded to a packet traffic previously authorized in a earlier control. This information is stored in a dedicated register that is accessed via hash function that includes IPs, ports and protocol used.

The hash algorithm we have chosen is Cyclic Redundancy Check ([25]) with 32 bits (CRC32). Usually, it is an algorithm used for detecting some accidental changes or errors present in digital data, but we have used it only to index packet traffics into this register. This hash algorithm is very popular due to its peculiarity to be simple to implement in binary hardware and its efficiency.

Another way to build this authorization register could be represented by a Bloom filter ([3]), but this would mean involving a probabilistic indexation which it should be avoided to not fall into a false positive case, giving an authorization to an unintended packet traffic.
Hereinafter, this procedure is illustrated in the Algorithm 4.1 and in Figure 4.4.

Algorithm 4.1 Already authorized phase

```
if TCP or UDP not present then
    packet traffic is not authorized
else
    if TCP is present then
        store TCP source port and destination port
    else if UDP is present then
        store UDP source port and destination port
    end if
    calculate hash
    retrieve from authorization register if this packet traffic was authorized or not
end if
```

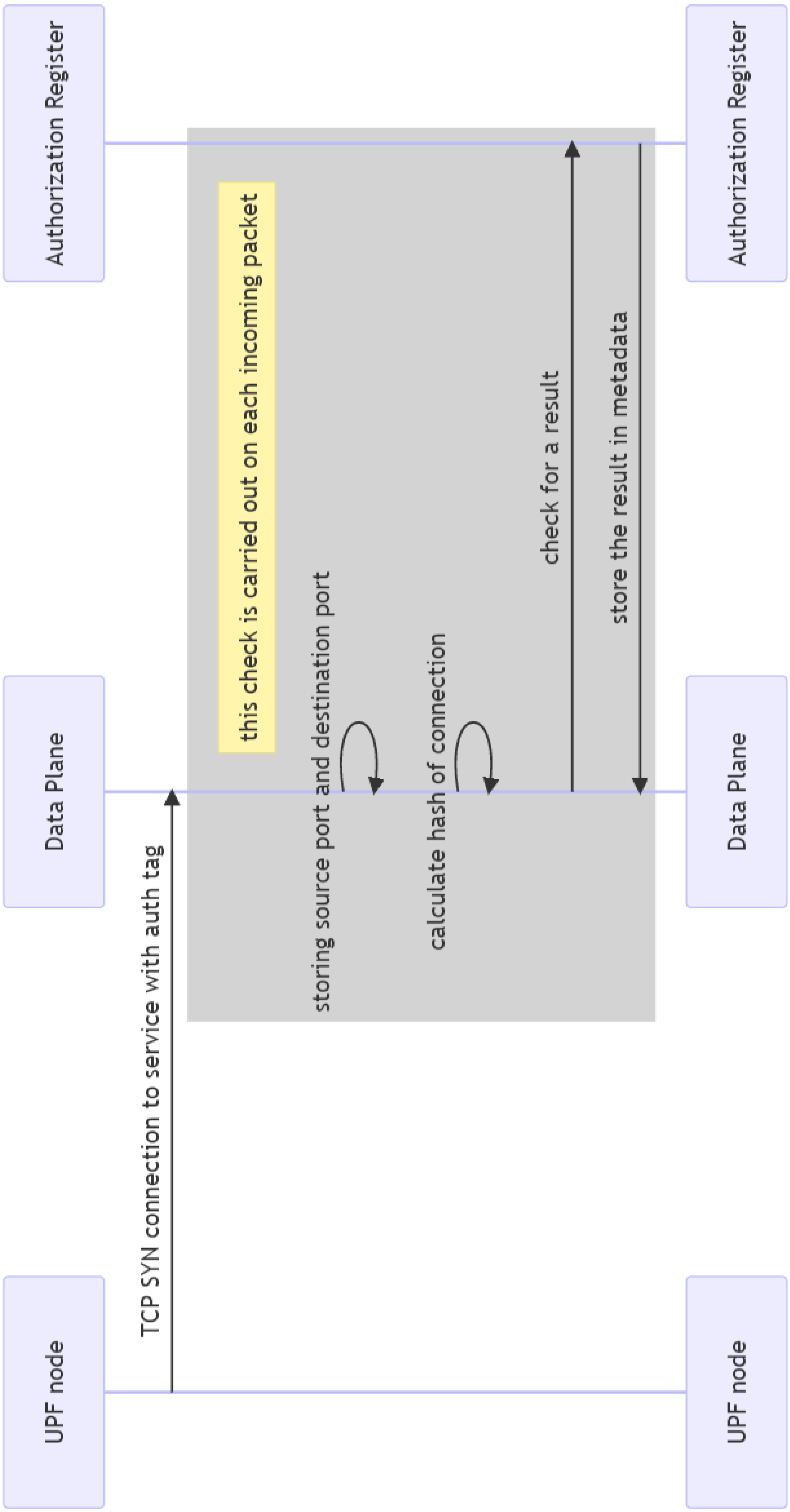

UPF node
Data Plane
Authorization Register
TCP SYN connection to service with auth tag
this check is carried out on each incoming packet
storing source port and destination port
calculate hash of connection
check for a result
store the result in metadata
UPF node
Data Plane
Authorization Register

Figure 4.4: Transition of a packet inside the data plane during the already authorized phase

4.2.2. Handling packet phase

There is also a second concept block inside our data plane and its role is to check the result of the preceding phase. If the result is positive, it means that the packet can access directly the forwarding table for matching one of the rules inside that table and then, in case of a positive match, the packet is forwarded; else it is dropped.

Otherwise, that packet traffic needs to be authorized by going first through the HMAC table which contains various hashes precalculated from our control plane.
The current packet has to include a NSH header with an HMAC inside for matching one of the previous hashes. In case of a positive outcome, that packet traffic is set as authorized in the previous stateful register and the packet can be forwarded thanks to the action inserted previously by the control plane.

When a packet tries to match a rule inside the forward table, it needs to have exactly the same IPs, source and destination ports, which are used as wildcards when the control plane inserts the rules (it does not know the final UPF node port during the key exchange phase, due to the 5G natting).
Only after having received the NSH packet, it can update the round trip rules making them permanent.

In the event of a not authorized packet, which does not contain any NSH header but it has a valid IP layer without a level-four layer, it is sent to the controller to check if it is an ICMP or an ARP packet. Else, it is discarded because we can consider it as a misformatted packet.

There is also an egress section where if the packet contains a NSH header, the ethertype of that packet is changed in an IPv4 type; in this way, it can be forwarded in the outside network.
Moreover, if it is a cloned packet, it is sent to the controller as a trigger for making some updates to traffic rules: adding a port due to the 5G network which screens the source port until the passage of first packet with NSH header.

Before one packet exits from data plane, in order to make that packet traffic able to be accepted by the kernel of the machines that host our data plane and the other components (client and server) which receive their respective packet traffics, the two checksums regarding IP and TCP/UDP layers have to be computed.

During the deparsering, the NSH header is removed because it has to be forwarded toward an extern network, matching the documentation regarded to this header which associates the presence of a NSH header only inside a local network ([29]).

The handling packet and egress phases are represented in Algorithm 4.2 and 4.3. In Figure 4.5 is depicted the transaction of a packet with the NSH header which can be forwarded to the service server.

Algorithm 4.2 Handling packet phase

```
if packet traffic is not authorized then
    if NSH is present then
        if NSH metadata == a table entry then            ▷ applying HMAC table
            decrementing TTL
            $egrassPort \leftarrow serviceSwitchPort$
            set the MAC addresses
            if TCP or UDP protocol then
                calculate hashes
                set True in authorization register at indexes equal to the previous calcu-
lated hashes
            end if
            clone packet toward the controller
        else
            drop packet
        end if
    else if IPv4 layer is present and layer TCP/UDP is not then
        send packet to the controller
    else
        drop packet                         ▷ no significant packet or a malicious one
    end if
else
    if TCP is present then
        store TCP source port and destination port
        apply forward table
    else if UDP is present then
        store UDP source port and destination port
        apply forward table
    else
        drop packet                                          ▷ it is misformatted
    end if
end if
```

Algorithm 4.3 Egress process

```
if NSH is present then
    $etherType \leftarrow 0x800$
end if
if P4 packet type is a clone then
    $egressPort \leftarrow controllerPort$
end if
```

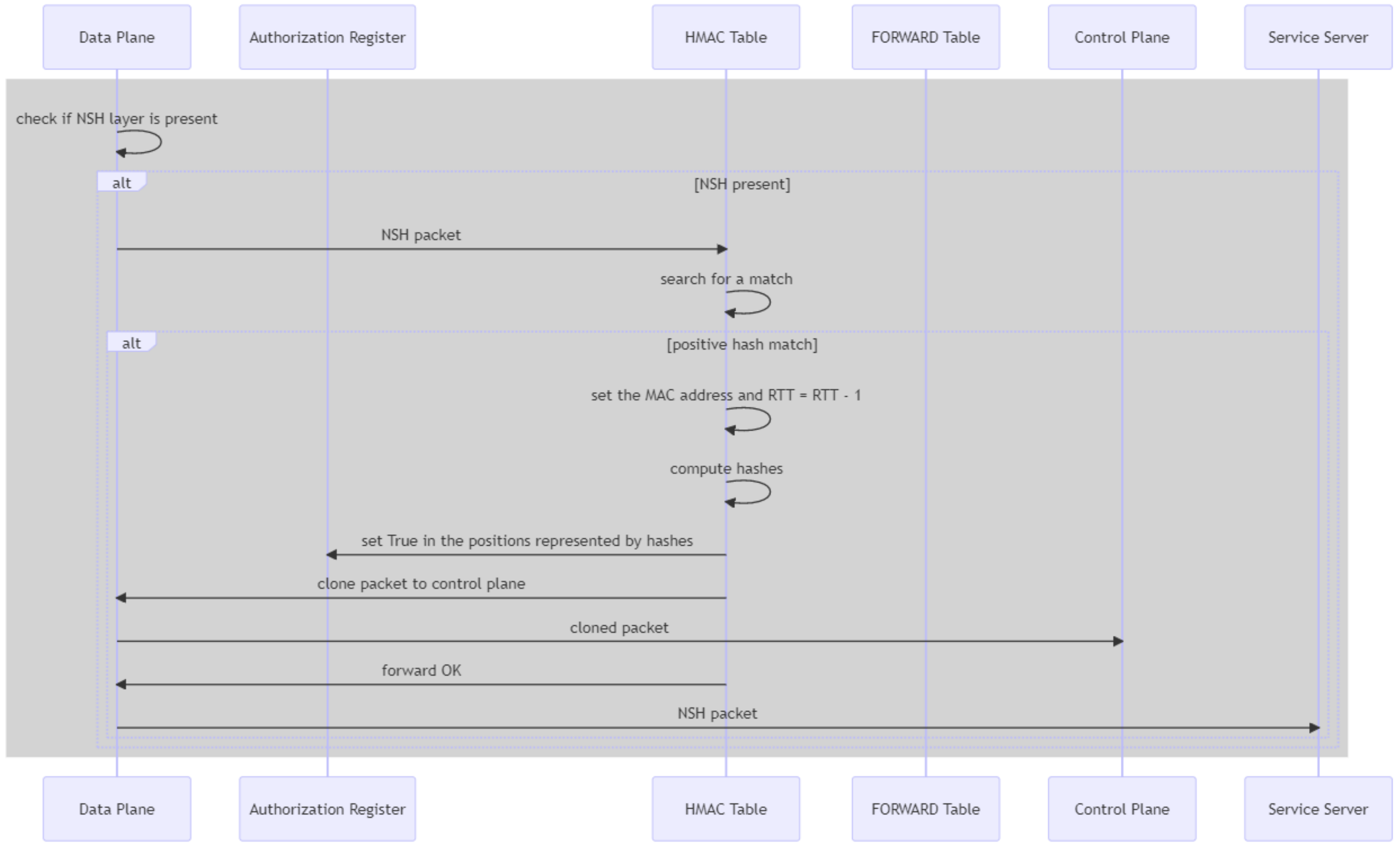


Figure 4.5: Transition of a packet inside the data plane during the handling packet phase in case of a packet with NSH and a positive match in the HMAC table (only the relevant passages are shown)

4.3. Policy Server

In our implementation, we have designed the policy server as a .yaml file for making the access to it easier. It is read by the control plane during the key exchange phase to get if a user entity is authenticated or not for a specific service and when a policy change is done.

4.3.1. Policy list

This file is structured as a list of policies, where each of its is built with these fields:

- serviceName: the name reserved to a service.
- ip: IP address associated to the entrypoint of service server.
- port: dedicated port of the entrypoint of service server.
- protocol: the type of transport protocol utilised to be admitted to the service server.
- allowedUsers: it is a list of authorized users for a specific service.
  - method: this is the variable through which a user is identified. This could be an IP address (which usually is dynamic), IMSI or a token.
  - user: the value related to the method used.
- tee: a boolean value which indicates if the option Trusted Execution Environment is required.
- fsEncr: a logic value which states if a File System Encryption is necessary.
- netEncr: a True/False value which points out if a Network Encryption is needed.
- secBoot: a bit value which specifies if Secure Boot option must be applied.

4.4. Control Plane

Control plane was structured as a single controller which orchestrates our data plane, according to the SDN paradigm.

It includes a set up configuration to handle the manipulation of rules inside the data plane, a part where it retrieves the policies from the policy database, a clean up section for removing the already present rules inserted before the restarting of controller and its core is made by three threads running in parallel.

4.4.1. DH key exchange

Inside this thread a TCP socket server is launched and gets connections in loop. Everytime a connection is accepted, it obtains the imsi of the mobile sim, the service name and the parameters for the computation of Diffie-Hellman key.

Immediately after, it starts a procedure to verify if the imsi is an authorized one by checking inside the policy file. In case of a positive match, it starts the key computation procedure creating the user's master key, inserts the traffic rules inside the data plane and sends back to the client a parameter to allow the client to calculate the master key by themselves.

For what concern the master key, it is computed using the SHA256 hash function with a parameter retrieved from the packet sent by client. Then, it is utilized to forge the HMAC with a counter and an object which contains all the information regarded the packet traffic between client and server.
The abstract composition of this thread is reported in the Algorithm 4.4.

Algorithm 4.4 Diffie-Hellman exchange procedure

```
create and bind a TCP socket
while True do
    retrieve DH parameters, imsi and service name from the current connection
    $serviceIpPort \leftarrow lookForPolicy()$ ▷ searching if there are a policy with that name and the associated imsi
    if servirceIpPort == -1 then
        $B \leftarrow -1$ ▷ no policy with that service name or imsi is not authenticated
    else
        $B \leftarrow keyComputation()$ ▷ everything is fine, starting computing the key returning a DH parameter for the client
    end if
    send B back to the client
end while
```

4.4.2. Detector policy modifications

Within this thread, there is a procedure which involves a notify adapter to signal if some event is raised; in particular, after a write event inside the policy file a manager procedure began to compare the entire previous policy list stored before with the new one.

Then, the function compares every aspect of each policy: IPs, ports, the user entities and the security parameters, modifying it or signaling that there was a change of an attribute.

This detector procedure is reported in Algorithm 4.5.

Algorithm 4.5 Detector policy modifications procedure

```
while True do
    instantiate a notify object
    attach it to the policy file
    for after every write event do
        $tmpList \leftarrow oldPolicyList$
        for every policy in tmpList do
            for every policy in newPolicyList do
                if policyTmp matches policyNew then
                    if a IP is changed then
                        modify tmpList
                    end if
                    if a port is changed then
                        modify tmpList
                    end if
                    if a UE is add then
                        report the changeover
                    end if
                    if a UE is removed then
                        delete it from tmpList
                    end if
                    if a security parameter is changed then
                        report the changeover
                    end if
                else
                    remove that policy from tmpList
                end if
            end for
        end for
    end for
end while
```

4.4.3. Packet manager

The packet manager thread has the role to capture the packets incoming from the previous selection of data plane. After receiving them, it needs to enforce another filter to be sure to obtain packets which have the access point of service server as destination or as source.

Once this thread gets a packet, it is captured by sectioning and storing its layers.

Afterward, there are some checks involving the nature of packet; i.e., if it is an ARP or ICMP packet or, alternatively, contains a TCP/UDP layer. In the first case, the MAC address is retrieved from the data link layer and associated with the IP from which the packet came from.

In the situation where the packet includes a transport layer and it is directed to the service server, there is a procedure to update the rules inside the data plane which have been inserted during the key exchange phase without the source port due to the presence of 5G natting.

The management of which rules has to be updated is done with this methodology: when there is the key exchange phase, the controller keeps track of the rules inserted in a list. Therefore, when it has to add the source port in a rule, the controller needs only to recall the relating instances inside this list and manipulate it.

Hereinafter, the thread description is written in the Algorithm 4.6.

Algorithm 4.6 Packet management procedure

```
while True do
    retrieve a packet from data plane
    classify the layers of that packet
    if packet protocol is TCP/UDP and it contains the access point of service server then
        update the corresponding rules inside the data plane tables
    else if packet protocol is ICMP then
        discard the packet    ▷ It cannot be filtered before because there is no header in P4 to handle it
    else if packet protocol is ARP then
        retrieve MAC address
        associate it with the IP of packet
    end if
end while
```

4.5. Design criteria

In this section, some additional considerations are presented about this implementation.

4.5.1. Table size modeling for an optimal edge slack

There are some relevant aspects of this implementation which were not analyzed before: the optimal HMAC table size, the number of requests after which it is necessary to trigger the HMAC table update per user and a possible number of devices to which the services are to be provided.
Specifically, if a user would want to connect more times to different services or change their source port and then reconnect to the same service, the HMAC table should have other hashes to ensure that the authorization functionality of the data plane remains active without any loss of packets for users. Therefore, it was required to establish a number of requests after which these updates are activated.

So, we were required to imagine a possible situation where the control plane is overloaded but it can handle these table updates.

With the objective to retrieve these informations, we have relied on the queuing theory to approximate our architecture to a statistic model.

For what concern our situation, we needed to refine the analysis because we wanted to find a reasonable number of users which can be handled by our MEC component, considering that the queue is not a problem in our case.
The variables of this problem are reported inside the Table 4.1.

Variable	Its meaning
λ_i	The arrival rate of device i
λ_{tot}	The arrival rate of all devices together
w	The number of all devices
N	The requests after which the update needs to be performed per user
$T_{service}$	Time for adding an hash inside the HMAC table
D(N)	The average time spent by a packet inside the system before to be processed
c	The estimated size of HMAC table

Table 4.1: All variables related to our queue for the edge slack sizing considerations

Therefore, the envisioned situation is similar to the one reported in Figure 4.6.

Specifically, we had to figure out an edge slack to avoid any interruption, giving the right time to the control plane to update HMAC table with new HMACs and maintaining the

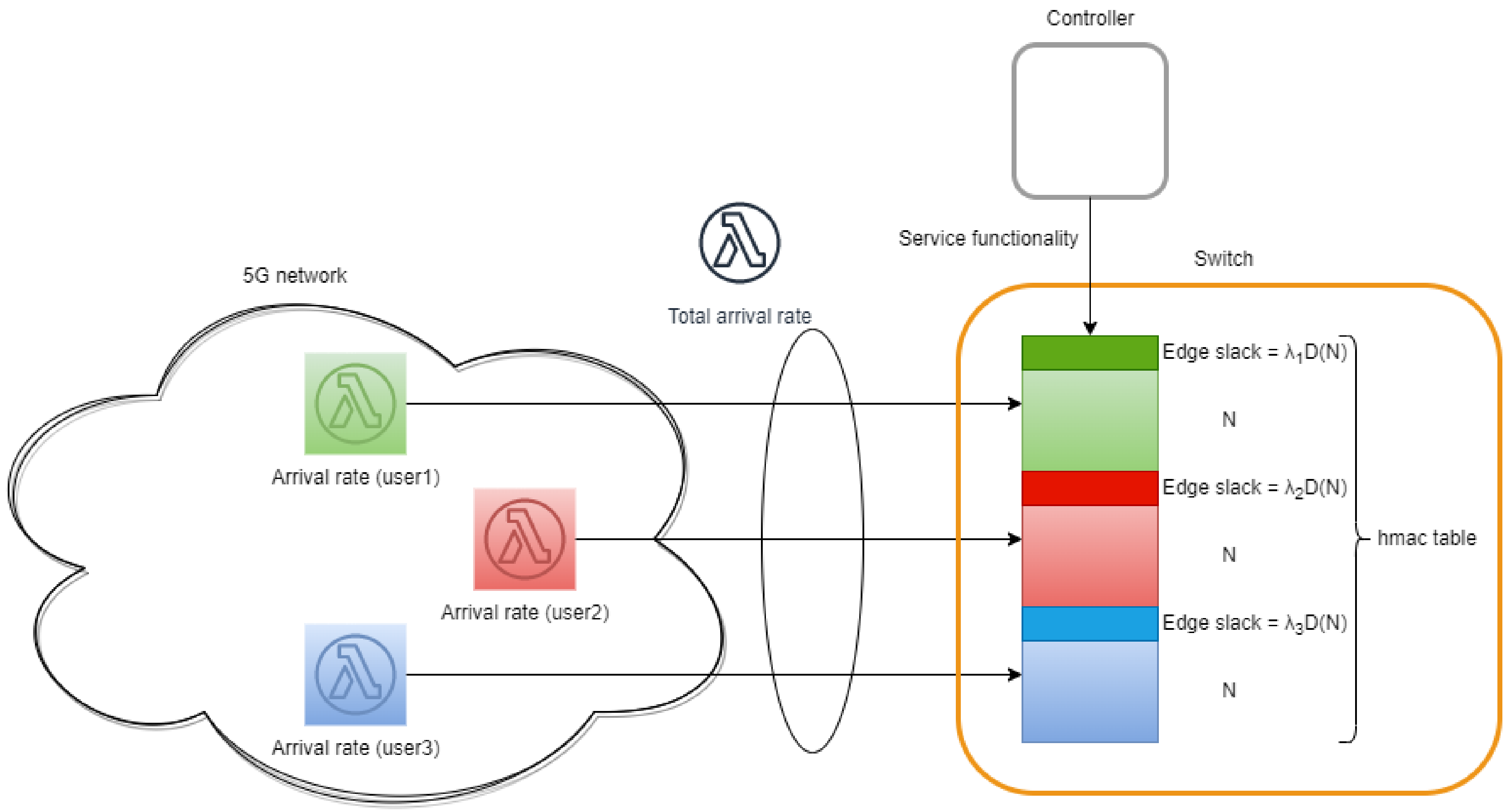


Figure 4.6: Our HMAC table size modeling problem approximated with the usage of the queuing theory in case of 3 users

service available to a consistent number of users.

We have taken into account that the number of users (w) and the optimal number of packets per user (N) are the two unknown variables, λ_{tot} can be parameterised fully representing the loquacity of all devices and that $T_{service}$ can be calculated empirically.

After some tests, we obtained that the average time spent by the controller to compute and insert one HMAC instance inside the HMAC table is $T_{service} = 5.5\,\text{ms}$. The box plot in Figure 4.7 represents the taken samples.

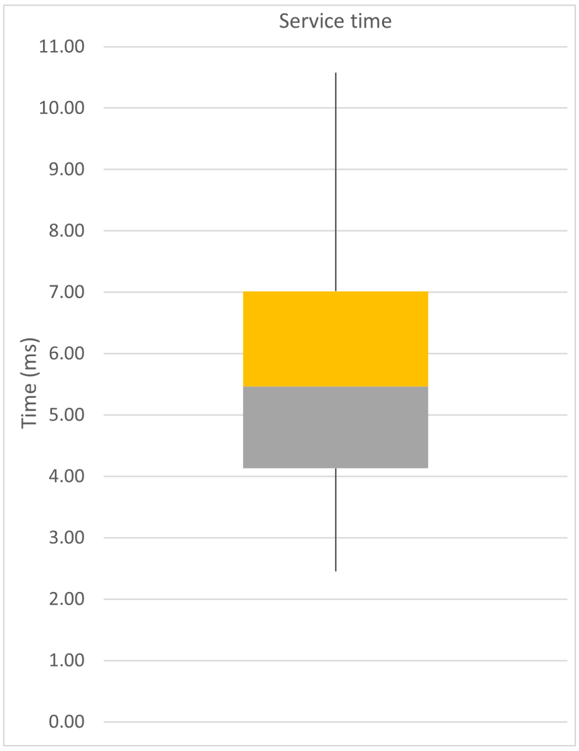


Figure 4.7: The measurements of the amount of time taken by the control plane to insert a new HMAC into the data plane table

To retrieve the other informations, we had considered to select the *M/M/1 queue* which is consistent with our architecture due to its characteristics.

It is based on the fact that the incoming packets arrive to destination with a Poisson distribution, the service time is approximated with a mutual exponent distribution and there is only one server for that exposed service (in our case the authorization procedure).

From literature, we have been able to find that the time spent in our data plane before an authorization packet is processed can be represented by the Inequality 4.1:

$$D(N) = \frac{T_{service}}{1 - \lambda_{tot} T_{service}} = \frac{T_{service}}{1 - \frac{w\lambda_i}{N} T_{service}} = \frac{N T_{service}}{N - w\lambda_i T_{service}} \tag{4.1}$$

Where $\lambda_{tot} = w\lambda_i$ was possible because we have assumed, for the sake of simplicity, that every user's device had the same loquacity λ_i.

In particular, to resolve our problem, we were required to define an inequality which balances the total hashes already consumed plus the incoming ones to the actual table capacity, obtaining the Inequality 4.2:

$$wN + D(N)\lambda_{tot} \leq c \implies wN + \frac{N w\lambda_i T_{service}}{N - w\lambda_i T_{service}} \leq c \tag{4.2}$$

In addition, we would not want to create a probable overloading situation for our control plane. So, we were needed to add one constraint: the number of updates to the control plane per user has to be less than the capacity of controller to serve them.
This problem can be represented by the Inequality 4.3:

$$\frac{w\lambda_i}{N} < \frac{1}{T_{service}} \implies N > w\lambda_i T_{service} \tag{4.3}$$

In our data, we had missed the effectively table capacity of our data plane to implement our HMAC table. This number depends on the P4 compiler and the target architecture which could be a software or an hardware one. Moreover, many hardware devices are protected by Non Disclosure Agreement (NDA), making impossible to predict a concrete (but target-dependent) number.

Nevertheless, we had taken into consideration that P4 is a cross-compliant language for packet manipulation which is becoming progressively the *de facto standard* (e.g., some compilers for Field-Programmable Gate Array (FPGA) are used, like p4-to-FPGA [5] and it is started thinking about how to include P4 inside the linux kernel [26]) and that the current reference hardware implementations are:

- Tofino intel
- The Netronome switches
- Xlinxs FPGA

Therefore, we had decided to consider the Alveo U280 Data Center Accelerator Card FPGA produced by Xlinxs as our target example to retrieve a possible number to size our table. We were conscious that a FPGA would have implied a trade-off for memory space between code and hardware resources but we had chosen that it could be a good reference point for an approximation number due to the previous considerations and also because its datasheet is available online ([2]).

This FPGA has a SRAM of 41 MB. So, we had considered: an hypothetical value of 16 MB to store inside that FPGA our HMAC table and that our authorization tag is long 16 B.
Then, we had retrieved that our table may be composed by 10^6 rows.

The plots in Figure 4.8 that has been scaled to show a significant result illustrate the inequality previously analysed after: having calculated that the service time plus a safety margin is equal to $T_{service} = 6\,\text{ms}$, estimated that a feasible capacity for our table is 10^6 rows and chosen that the loquacity of a single device is 20 packet/s.

In these plots we had selected as Cartesian axes:

- x-axis: the number of devices (w).
- y-axis: the number of packets per user after which the updates are triggered for that user (N).

Therefore, it can be seen that the black line in the bi-dimensional plot is our limited condition to not overload the control plane and, in order to not risk to accumulate packets, the state of our system had to be set above that line.
In the three-dimensional plot, we can notice that the area where our switch can work without any problem is the double red one, which is the intersection of our two inequalities, while the blue areas are the not-safe ones.

So, in order to be conservative, a reasonable number of users can be set between 2500 and 2600 with an hash-consumed trigger that can be defined between 340 and 360 to avoid to expose the control plane to too much work and having an affordable number of users at the same time.

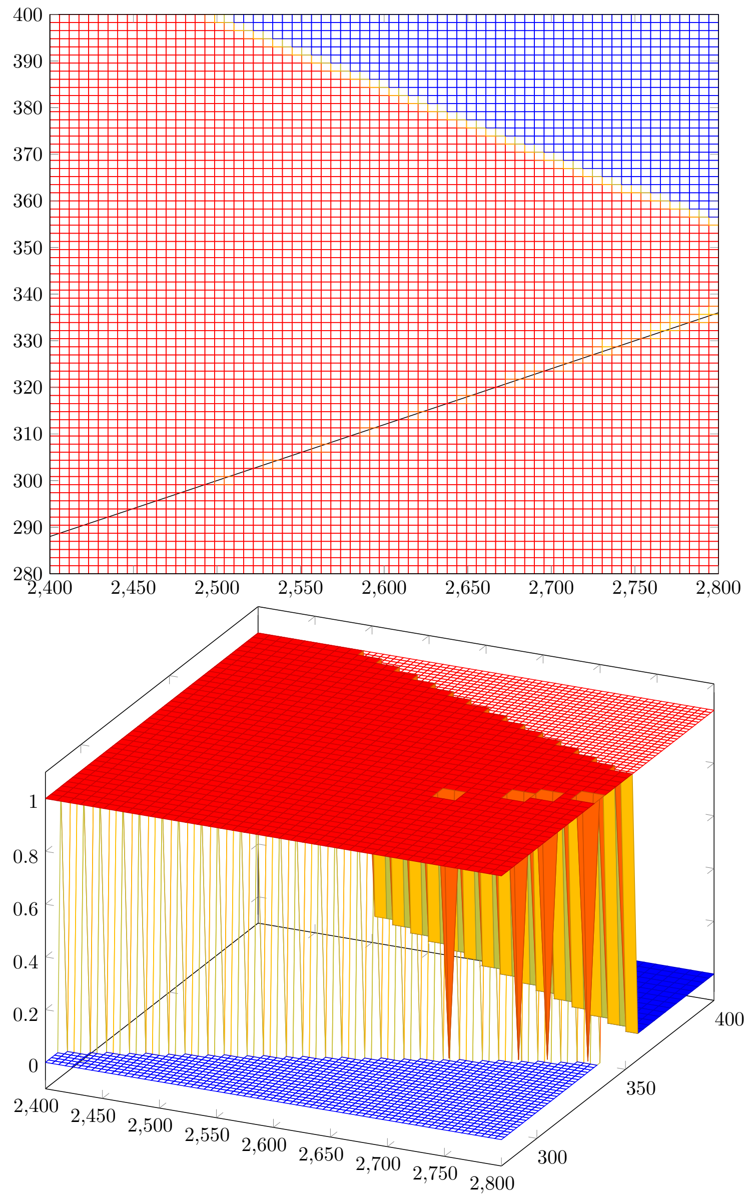


Figure 4.8: Plots of the optimal number of packets in the system to trigger the updates per user on HMAC table in data plane (on y-axis) and the users handled by the system (on x-axis)

4.5.2. Security considerations

It is possible to make some considerations in relation to potential cyberattacks that could be released against this architecture by the actors described in Section 3.3.1, that can be classified in:

- Internal attacker: a user who has compromised a machine of the system or bypassed an authentication policy.
- External attacker: a external entity (a person or a manipulated machine) which wants to attack the system.

To make an affordable analysis for the possible types of attack, a practical list is given by [38], where the authors had drawn up the Table 4.2, which itemizes the most relevant network security attacks.

	ATTACKERS	DEFENDERS AND DEFENSE MECHANISMS	POTENTIAL VICTIMS
FLOODING ATTACKS	unsafe users	network defends with traffic filtering and/or dynamic resource allocation safe users defend with dynamic resource allocation	network and/or unsafe users
SUBVERSION ATTACKS	unsafe users	network defends with traffic filtering and/or cryptographic protocols safe users defend with cryptographic protocols	network and/or unsafe users
POLICY VIOLATION	unsafe users	network defends with traffic filtering	authority responsible for the policy
SPYING AND TAMPERING	network and/or unsafe users	safe users defend with cryptographic protocols and/or compound sessions and overlays	safe users

Table 4.2: A practical classification of network security attacks ([38])

Every network architecture cannot be fully protected from a Denial of Service attack. It works sending a multitude of requests continuously towards a server putting it in a overloaded condition resulting in a temporary suspension of its service or more than one.

It could be possible to implement a splicing TCP to prevent a SYN flood attack or a dynamic speed adaptation which can restrict the impact of the attack. For what concern our environment, the switch filters the traffic packets, allowing only the packet directed to one of the services.

Anyway, we have not considered this type of attack due to the absence of incentives for an attacker. The computational power required to carry on this attack is not so easy to achieve and even in that case, the malicious actor needs to be located in a geographical position near our CES component.
Moreover, the attacker does not retrieve any huge award after a positive impairment.

The first type of attack we have taken into account is the policy violation attack, which can be described as an attempt to bypass some policies defined by the authority of that network.

The most effective way to defend the network from this offense is implementing a packet traffic filtering procedure.
In our case, we have protected our system from this type of threat during the key exchange phase of our protocol. After the reception of parameters to compute the master key, the control plane interrogates the policy server to check whether the alleged user from whom the packet originates is authenticated for that service or not.

Another type of offensive we have borne in mind is the subversion attack. Its purpose is to take possession of a victim's machine and controlling it as the attacker wants, affecting also other machines located in the same network.

We have done a couple of considerations for this attack. The service server instantiates the services utilizing the FaaS paradigm to handle this possibility, avoiding the expansion of the attacker from a machine allocated. In addition, our protocol avoids to allow any malicious TCP connection, requiring that the first packet has to incorporate the authorization tag.

The last type of attack to be looked at it is the spying and tampering, that consists in an attacker which can scan the packet traffics selecting the ones they are interesting to read or to block, for modifying a packet and then to redirect that packet to the original addressee entity.

One of these compromising threats is the reply attack. It consists in spoofing a packet, reading a specific layer and then obtaining the informations for which the attacker is looking for and using it to gain authorization/authentication for a targeted service.
In our case, the attacker needs to retrieve the data inside the NSH header. But it would

be useless thanks to the removing procedure of the hashes carried out by control plane immediately after the reception of a packet clone from the data plane.

Against an hijacking attack the system is not completely safe. It consists in a redirection of the service packet traffic towards the attacker and then they start to emulate the server against the client who needs to access to the service.
Nevertheless, a possible trade-off is represented by a continuous requests of new hashes after a time window, e.g. 5-10 seconds. In this way, the attacker cannot infer anything related to the hash to supply; hence, the system could be under attack only during a predetermined time window.

An attack that can be brought against our protocol concerns the key exchange phase; in particular, the Diffie-Hellman key exchange algorithm suffers against man-in-the-middle attacks. A reasonable countermeasure is to instantiate an asymmetric key, by leveraging an algorithm like RSA, in order to publish a public key by which it can be made an encryption phase before the symmetric key exchange, nullifying all spoofing attempts. Moreover, to be sure to establish a key with the AUSF node of 5G network, it is convenient communicating towards TLS sockets, relying on the Transport Layer Security for a secure connection.

5 | Performance analysis

The source code and tests (performed also using Vagrant ([36])) of this project are publicly available under the permissive MIT licence[1].

5.1. Methodology

To test the efficiency of the presented authorization protocol we have performed a performance analysis to compare a switch implementing this protocol to a packet traffic condition where a switch with preexisting rules were admitting the passage of packets for that traffic.

The machine where the environment was set up had incorporated as CPU an Intel Xeon Silver 4114 with a frequency of 2.2 GHz, 10 cores per socket, 2 threads per core, a RAM of 64 GB and it was shared between several users. The time samples were retrieved from the interface of the virtual machine where the Client part was hosted, utilizing the timestamps extracted from the informations of the packets dumped by the tcpdump tool ([35]).

These performances were influenced by the following components considered to simulate the environment illustrated in Section 3.2:

- Client part: it has been represented by two Python scripts running on a first virtual machine (VM) reproducing the AUSF and UPF nodes of the 5G network.
- Data plane: it has been instantiated as a BMv2 software switch functioning on a second virtual machine, which obtains the packet traffics from the previous VM.
- Control plane: it has been considered as an orchestrator Python script which behaves as the controller of data plane, working on the same VM of our switch.
- Policy server: it has been structured as a text file (a simple .yaml file) which emulates a database but it can be implemented also as a database or as a REST server (like in [20]).

[1]https://github.com/ANTLab-polimi/Customer-Edge-Switch.git

- Service part: it has been designed as a simple server started within a Python file, running on a third virtual machine getting the authorized packet traffics from the client part.

Specifically, the virtual machines can send and receive the packets under private networks established directly in a Vagrantfile, which instantiates all the three VMs. This private network was made to avoid the loss of some packets, simplifying our analysis.

We had reproduced an environment where our client part is taking on the appearance of two components: the first one is the AUSF node which has the role to exchange the master key of a user and the second one is a device inside the UPF node of a 5G network which manipulates the first user mobile packet to a service by inserting inside the network layer the NSH header. After this injection, our UPF-node component sends the packets towards the private network to our second virtual machine.
Inside this VM there are a BMv2 switch and a controller which are running, checking the packet traffic. Afterward, there is a service server on the third VM receiving the authorized packets from the switch.

5.2. Protocol application

In order to verify the time spent by the usage of our authorization protocol, we have tested the first TCP three-way handshake of some connections, in two different configurations, and the key exchange phase.

The key exchange phase has taken place between two TCP sockets written in Python. The client socket was located inside the first VM which emulated the client part, in particular the AUSF node, and the server socket was situated within the server part accounting for the real service instantiated with the FaaS paradigm.

This condition had reduced the possible delay due to the environment because a socket TCP connection is handled inside the kernel space. In fact, the average time spent for this phase was around 115 ms, which is a reasonably short amount of time and it would be payed only once, during the first TCP three-way handshake of the first connection required by a user.

The box plot in Figure 5.1 presents the results of this phase.

In our analysis, there is one core case and it is related to the first Round Trip Time (RTT) of the initial TCP three-way handshake of a user's connection, because it is the only difference between a normal TCP handshake and a TCP handshake which includes

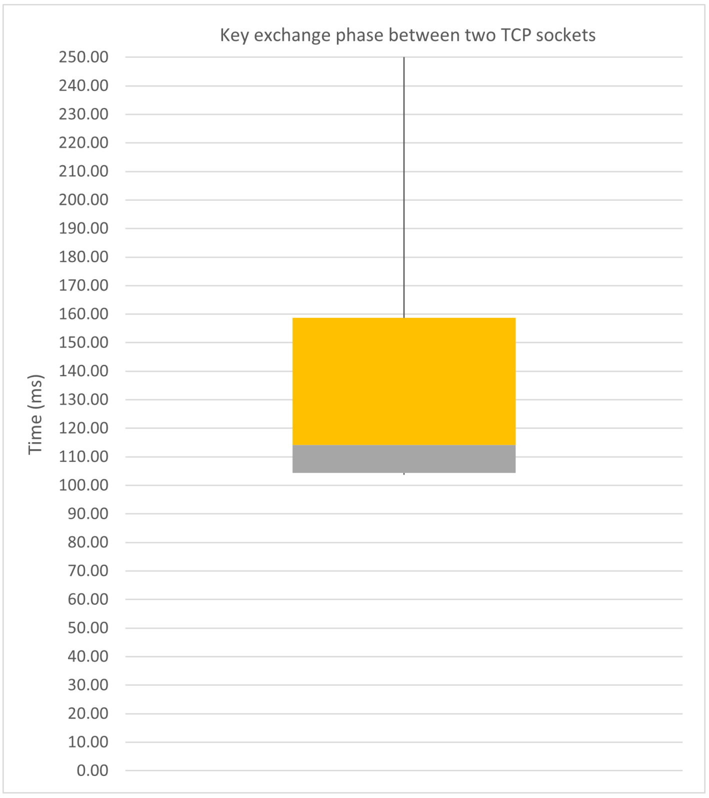


Figure 5.1: The performance of the key exchange phase using two TCP sockets to simulate the AUSF node and the control plane

our authorization tag.

For this phase we have utilised Scapy ([32]). It is an interactive packet manipulation library written in Python, which is capable of forge packets adopting several protocols.

This choice was made because Scapy is easy-accessible and a simple way to forge packets for our goal. In fact, the injection of the NSH header inside the TCP handshake cannot be done with a classical TCP socket. Moreover, Scapy had allowed us to act in the user space level simulating a TCP connection without opening a TCP socket or working on the packet traffics between the first and the second VM with the usage of an additional switch to inject that header.

For what concern the types of analysis, we have chosen two different cases:

- A switch implementing our authorization protocol.
- A switch with pre-inserted rules inside, in order to not block the packet traffic.

The first RTT of the TCP three-way handshake of a connection had spent an average time of 6.5 ms with the usage of our protocol, introducing an average delay time of 7%

more than the case without our protocol. This result was given to us because all the authorization checks are done inside the data plane and the temporary rules are inserted in it immediately before the end of the key exchange phase.

The box plots inside the Figure 5.2 represent the outcome of these two depicted situations.

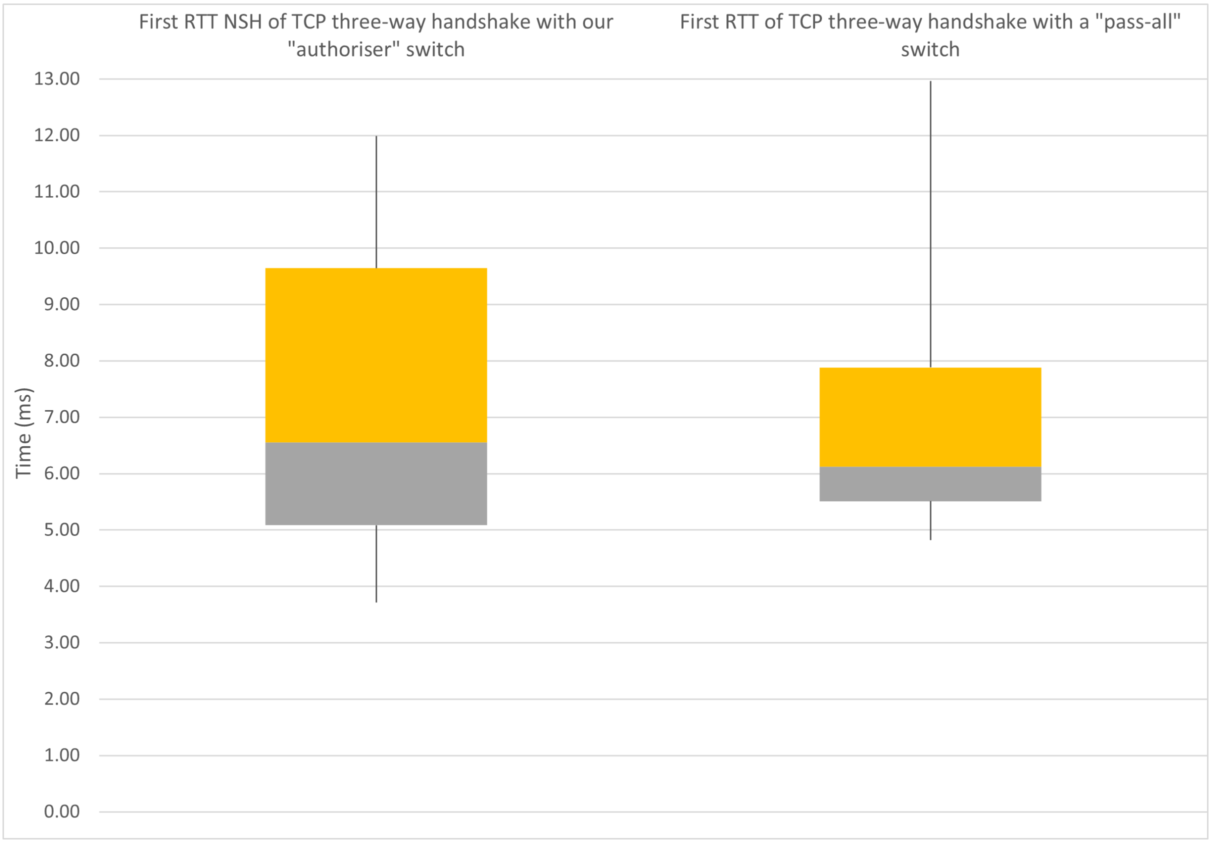


Figure 5.2: The performance comparison between the first RTT of a TCP three-way handshake executed with a switch implementing our authorization protocol and the first RTT of a TCP three-way handshake with a switch that already includes rules for the circulation of packets

6 | Conclusions

The authentication and authorization protocols can be developed with a plethora of several methods at various levels.
Nowadays, the application of security protocols can be done also at network layer improving the security related to an environment, including also the levels under the application layer.
This is available thanks to the introduction of the SDN paradigm and, specifically, to the possibility of separating data plane from the control plane.

Our project was aimed at implementing a SDN component which can satisfy the demand of devices that enforce authentication and authorization procedures. Our component exploits a policy-based control and a network-layer protocol to provide users to access services via a reproduced 5G network, within a Mobile Edge Computing scenario.

Taking into account the FaaS cloud services, the dynamic modulation of these services requires that this presented protocol has to be adequately reactive to deal with the continuous policies adjustments and entities updates, keeping the service itself always available as well as the authorization procedure.

Moreover, the 5G networks are becoming increasingly widespread bringing with themselves an huge amount of requests to satisfy.
Consequently, the concept of security is becoming more and more dominant with the major assertions that it cannot slow down the performance with the introduction of a relevant delay and, at the same time, it has to ensure privacy to all the users.

After the comparison of the performance between the packet traffics in a normal condition and with our authorization protocol, it can be observed that the introduction of our security mechanism does not add a marked delay in relation to a normal traffic flow, making this same protocol enforceable.

6.1. Future works

6.1.1. BLS digital signature

An improving concept is the introduction of Boneh-Lynn-Shacham digital signature ([4]) which enables a user to verify that a signer is authentic.

It is possible to conceptualise a new scenario where there is not only one SDN component deployed implementing this same authorization protocol, but also multiple sub-systems with this type of device in an intranet with the purpose of forward packets. It is necessary to sign the packet elaborated at each step of this intranet to prevent a security attack.

An interesting application of BLS digital signature is represented by this situation thanks to one of its properties: the aggregate signatures.
Any party that has all the signatures can aggregate one more sign and this aggregation is done incrementally. It is secure because a possible attacker should be able to make an existentially forge attack resolving the co-Computational Diffie-Hellman problem which is currently known as a hard challenge.

Therefore, every SDN component in the path needs to sign the packet making an aggregation sign. In this way, the destination server is able to check the path of the packet to be sure that it was picked up by those devices in a designated sequence.

Bibliography

[1] *AI-SPRINT project.* URL: `https://www.ai-sprint-project.eu/`.

[2] *Alveo U280 Data Center Accelerator Card datasheet.* URL: `https://www.xilinx.com/products/boards-and-kits/alveo/u280.html#specifications`.

[3] Burton H Bloom. "Space/time trade-offs in hash coding with allowable errors". In: *Communications of the ACM* 13.7 (1970), pp. 422–426.

[4] Lynn Ben Boneh Dan and Shacham Hovav. "Short Signatures from the Weil Pairing". In: *J Cryptology* 17.23 (2004), pp. 297–319. URL: `https://doi.org/10.1007/s00145-004-0314-9`.

[5] Zhuang Cao et al. "P4 to FPGA-A Fast Approach for Generating Efficient Network Processors". In: *IEEE Access* 8 (2020), pp. 23440–23456. DOI: `10.1109/ACCESS.2020.2970683`.

[6] Erik Dahlman, Stefan Parkvall, and Johan Skold. *5G NR The Next Generation Wireless Access Technology.* Academic Press, 2020. ISBN: 9780128223208.

[7] Xiaoyu Duan and Xianbin Wang. "Authentication handover and privacy protection in 5G hetnets using software-defined networking". In: *IEEE Communications Magazine* 53.4 (2015), pp. 28–35.

[8] Open Networking Foundation. *P4 language tutorial.* URL: `https://bit.ly/p4d2-2018-spring`.

[9] Open Networking Foundation. *P4 webpage.* URL: `https://p4.org/` (visited on 03/11/2023).

[10] Open Networking Foundation. *P416 Language Specification.* URL: `https://p4.org/p4-spec/docs/P4-16-v-1.2.3.html#sec-actions` (visited on 03/19/2023).

[11] Open Networking Foundation. *P416 Portable Switch Architecture (PSA).* URL: `https://p4.org/p4-spec/docs/PSA-v1.2.html` (visited on 03/19/2023).

[12] Open Networking Foundation. *p4c complier.* URL: `https://github.com/p4lang/p4c` (visited on 03/19/2023).

[13] Open Networking Foundation. *P4Runtime Specification v1.3.0.* URL: `https://p4.org/p4-spec/p4runtime/v1.3.0/P4Runtime-Spec.html#sec-introduction-and-scope` (visited on 03/19/2023).

[14] Open Networking Foundation. *PI LIBRARY REPOSITORY - protobuf.* URL: `https://github.com/p4lang/PI/tree/main/proto` (visited on 03/19/2023).

[15] Jason Garbis and Jerry W. Chapman. "What Is Zero Trust?" In: *Zero Trust Security: An Enterprise Guide.* Berkeley, CA: Apress, 2021, pp. 7–18. ISBN: 978-1-4842-6702-8. DOI: `10.1007/978-1-4842-6702-8_2`. URL: `https://doi.org/10.1007/978-1-4842-6702-8_2`.

[16] Joel M. Halpern and Carlos Pignataro. *Service Function Chaining (SFC) Architecture.* RFC 7665. Oct. 2015. DOI: `10.17487/RFC7665`. URL: `https://www.rfc-editor.org/info/rfc7665`.

[17] Yun Chao Hu et al. "Mobile edge computing—A key technology towards 5G". In: *ETSI white paper* 11.11 (2015), pp. 1–16.

[18] Tongyi Huang et al. "A survey on green 6G network: Architecture and technologies". In: *IEEE access* 7 (2019), pp. 175758–175768.

[19] "IBM and Nokia Siemens Networks announce world first mobile edge computing platform". In: (2013). URL: `http://www-03.ibm.com/press/us/en/pressrelease/40490.wss`.

[20] Hammad Kabir, Muhammad Hassaan Bin Mohsin, and Raimo Kantola. "Implementing a security policy management for 5g customer edge nodes". In: *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium.* IEEE. 2020, pp. 1–8.

[21] Oguz Sunay Larry Peterson and Bruce Davie. *Private 5G: A Systems Approach.* URL: `https://github.com/SystemsApproach/private5g`.

[22] Madhusanka Liyanage et al. "5G Privacy: Scenarios and Solutions". In: *2018 IEEE 5G World Forum (5GWF).* 2018, pp. 197–203. DOI: `10.1109/5GWF.2018.8516981`.

[23] Theo Lynn et al. "A Preliminary Review of Enterprise Serverless Cloud Computing (Function-as-a-Service) Platforms". In: *2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom).* 2017, pp. 162–169. DOI: `10.1109/CloudCom.2017.15`.

[24] Olli Mämmelä et al. "Towards micro-segmentation in 5G network security". In: *European Conference on Networks and Communications (EuCNC 2016) Workshop on Network Management, Quality of Service and Security for 5G Networks.* 2016.

[25] B Maxwell et al. "Analysis of CRC methods and potential data integrity exploits". In: *International Conference on Emerging Technologies.* Citeseer. 2003, pp. 25–26.

[26] *P4 Over Linux TC Scripting Away At P4.* URL: `https://opennetworking.org/wp-content/uploads/2022/05/Jamal-Salim-Final-Slide-Deck.pdf`.

[27] Francesco Paolucci et al. “Enhancing 5G SDN/NFV Edge with P4 Data Plane Programmability”. In: *IEEE Network* 35.3 (2021), pp. 154–160. DOI: `10.1109/MNET.021.1900599`.

[28] Larry Peterson et al. *Software-Defined Networks: A Systems Approach.* URL: `https://github.com/SystemsApproach/SDN`.

[29] Paul Quinn, Uri Elzur, and Carlos Pignataro. *Network Service Header (NSH).* RFC 8300. Jan. 2018. DOI: `10.17487/RFC8300`. URL: `https://www.rfc-editor.org/info/rfc8300`.

[30] Paul Quinn and Jim Guichard. “Service Function Chaining: Creating a Service Plane via Network Service Headers”. In: *Computer* 47.11 (2014), pp. 38–44. DOI: `10.1109/MC.2014.328`.

[31] Rodrigo Roman, Javier Lopez, and Masahiro Mambo. “Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges”. In: *Future Generation Computer Systems* 78 (2018), pp. 680–698. URL: `https://doi.org/10.1016/j.future.2016.11.009`.

[32] *Scapy website.* URL: `https://scapy.net/`.

[33] Hamta Sedghani et al. “Advancing design and runtime management of AI applications with AI-SPRINT (position paper)”. In: *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC).* IEEE. 2021, pp. 1455–1462.

[34] Harrison J. Son. *7 Deployment Scenarios of Private 5G Networks.* URL: `https://www.netmanias.com/en/?m=view&id=blog&no=14500&xtag=5g-edge-kt-sk-telecom&xref=7-deployment-scenarios-of-private-5g-networks`.

[35] *tcpdump man pages.* URL: `https://www.tcpdump.org/manpages/tcpdump.1.html`.

[36] *Vagrant website.* URL: `https://www.vagrantup.com/`.

[37] Qigui Yao et al. “Dynamic Access Control and Authorization System Based on Zero-Trust Architecture”. In: *Proceedings of the 2020 1st International Conference on Control, Robotics and Intelligent System.* CCRIS ’20. Xiamen, China: Association for Computing Machinery, 2021, pp. 123–127. ISBN: 9781450388054. DOI: `10.1145/3437802.3437824`. URL: `https://doi.org/10.1145/3437802.3437824`.

[38] Pamela Zave and Jennifer Rexford. “Patterns and interactions in network security”. In: *ACM Computing Surveys (CSUR)* 53.6 (2020), pp. 1–37.

List of Figures

List of Tables

List of Algorithms

Acknowledgements

Giunto alla fine del mio lavoro di tesi per la laurea magistrale, vorrei ringraziare e menzionare tutte le persone che mi hanno sostenuto ed accompagnato durante il mio percorso di studi e supportato nello sviluppo e stesura della tesi.

Ringrazio il prof. Giacomo Verticale che in funzione di relatore mi ha guidato nello sviluppo di questo progetto e l'ing. Daniele Moro ed il prof. Gianni Antichi per i preziosi consigli su P4.

Ringrazio la mia famiglia: mia mamma Rosa, mio papà Raffaele e mio fratello Gabriele per tutto l'amore incondizionato che mi hanno dato e mi continuano a donare, per il sostegno perpetuo e per i consigli suggeritimi quando ne ho avuto bisogno.

Ringrazio Laura, che prima come amica e poi come compagna mi è stata vicina prima e durante la tesi, motivandomi in questi mesi con la sua intraprendenza e volontà a non mollare mai di fronte alle difficoltà, bensì ad affrontarle.

Ringrazio Alessio, Davide, Eleonora e Mattia per le giornate trascorse insieme dentro e fuori dall'università.

Ringrazio i "Muffini" Cristiano, Enka, Fabio, Francesco...ehm volevo dire Muff, Hakim, Matteo M., Meri, Priscilla, Veronica per il tempo condiviso insieme in questi anni.

Ringrazio Dario e Matteo F., per l'amicizia creatasi fin dal primo anno di università nonostante la distanza (si Crocetta è lontana dalla palestra).

Ringrazio Erica ed Oscar, per essermi stati vicino durante i periodi bui e per aver condiviso gran parte del tempo con loro durante questi ultimi due anni.

Ringrazio i gym bro, Danilo, Leyla e Loris per i pomeriggi (...in realtà serate...) trascorsi in palestra che, tra un'alzata e l'altra, hanno contribuito a rendere più leggero questo periodo intenso.

Ringrazio Roland, per le passeggiate percorse insieme di tanto in tanto per spezzare i periodi intensi.

Desidero fare una menzione d'onore a mia zia Maria Teresa ed a mia nonna Adriana. Alle quali, putroppo, è stata sottratta la possibilità di gioire con me, i miei amici ed i parenti, durante questo giorno speciale per via della caducità della vita e del destino che spesso si porta via i nostri cari che meno lo meriterebbero.

Infine, un ultimo ringraziamento a me stesso, per aver tenuto duro in questi anni spesso visti facili, perché "non si sta lavorando" ma che, in realtà, nascondono difficoltà intrinseche dovute alla continua pressione del dover finire e, soprattutto, alla pressione che noi stessi ci autoinduciamo.