FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Use-Case Conscious Trust Mechanisms for Low-Latency, Secure 5G and Beyond Connectivity

Pedro Osswald Claro

Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Ana Aguiar

Second Supervisor: Peter Steenkiste

July 31, 2024

Abstract

Emerging technologies, ranging from energy-efficient devices designed to operate on battery power for extended periods to ultra-low latency and highly reliable systems for mission-critical uses, are becoming increasingly prevalent, driving the need for network capabilities that meet a wide range of use case requirements. 5G cellular networks stand out as a technology aiming to fulfill these diverse needs, promising to provide adaptive access to services across different scenarios. To achieve these requirements, ongoing development is focused on minimizing latency and energy costs while ensuring robust security.

Security mechanisms employed by 5G technology are a contributor to the temporal and energetic overhead in connections through 5G technology. This study aims to therefore identify possible trade-offs between security and latency. A systematic characterization and analysis of the trust mechanisms within 5G communications is conducted to identify potential redundancies and trade-offs that are favorable for varied applications. To achieve this goal, this study utilizes a theoretical analysis of 5G protocols and algorithms, particularly focusing on authentication, with a secondary attention to confidentiality and integrity. By compiling the security technical specifications of 5G, this study discusses the need to view the 5G ecosystem as a cohesive whole rather than a collection of independent components.

The findings suggest that higher transparency between operators and end-consumers when choosing trust mechanisms can lead to improvements in latency for specific use cases. A categorization model is proposed to differentiate use cases by security-related factors, guiding the discussion on potential security trade-offs. The practical insights are provided by deploying a non-public 5G network using open-source software solutions. Through this deployment, the research quantifies the temporal overheads of various security measures and employs a mathematical model to study the impact of propagation delay variations on the experimental results.

This study aims to inform more efficient and use-case-appropriate security implementations in 5G networks, contributing to the broader objective of achieving low-latency requirements in modern applications. The specific contributions of this research are an analysis of the security options provided by 5G standards, a model to support choices in trust mechanisms based on use cases, and a quantification of the time overhead caused by these trust mechanisms.

Acknowledgements

I would first like to express my deepest gratitude to my Supervisor, Prof. Ana Aguiar, for the guidance provided to me over the last one and a half years. I hope to carry on the enthusiasm for research that your mentorship has fostered in me. I will always look back fondly on my time at the Shannon Lab as the routinely enriching, occasionally challenging, but always exciting period of my lifetime that it was.

I'm most thankful to my family for their support throughout this journey. To my siblings, Maria, Ana, and Daniel, thank you for lifting my spirits each and every day. To my parents, Teresa and João, for being my role models and showing me the importance of working hard and striving to do things right.

To my friends Zé, Sofia, and Maria, I am so very thankful that I got to write my thesis at the same time you did. You always made me feel understood, turning what could have otherwise been an at-times intimidating experience into a truly enjoyable one.

I sincerely thank my classmates - Marija and Filipe - with whom I shared the experimental setup during this project. I am lucky to have had such supportive people by my side. I am also grateful to Sofia Martins and José Rodrigues, as it was only upon their work and guidance that this project could be completed.

I am truly thankful for my co-supervisor, Prof. Peter Steenkiste, for his guidance in shaping the direction of my research and helping me identify the right questions to pursue.

This work was financially supported by IT (B-0113-23) under the scope of the project Route25 (C645463824-00000063), supported by the European Union / Next Generation EU, through Programa de Recuperação e Resiliência (PRR).

Pedro Osswald Claro

Contents

List of Figures

List of Tables

Abbreviations

Acronym	Definition
AES	Advanced Encryption Standard
AKA	Authentication and Key Agreement
AR	Augmented Reality
AV	Authentication Vector
AUTN	Authentication Token
AUSF	Authentication Server Function
CK	Cipher Key
CU	Central Unit
DTLS	Datagram Transport Layer Security
DU	Distributed Unit
eMBB	Enhanced Mobile Broadband
EAP	Extensible Authentication Protocol
GPRS	General Packet Radio Service
GUTI	Globally Unique Temporary Identity
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IK	Integrity Key
IMSI	International Mobile Subscriber Identity
IMEISV	International Mobile Station Equipment Identity Software Version
IoT	Internet of Things
LSAA	Lightweight and Secure Access Authentication
MAC	Message Authentication Code
mMTC	Massive Machine-Type Communications
NAI	Network Access Identifier
NAS	Non-Access Stratum
NTP	Network Time Protocol
NEA	New Radio Encryption Algorithm
NIA	New Radio Integrity Algorithm
OAI	OpenAirInterface
OPC	Operator Code
PDCP	Packet Data Convergence Protocol
PSK	Pre-Shared Key
RAN	Radio Access Network
RRC	Radio Resource Control
RU	Radio Unit
SIB1	System Information Block 1
SRBs	Signaling Radio Bearers

SEAF	Serving Authentication Function
SDG	Sustainable Development Goals
SIDF	Subscription Identifier De-concealing Function
SIM	Subscriber Identity Module
SMF	Session Management Function
SCTP	Stream Control Transport Protocol
SUPI	Subscription Permanent Identifier
SUCI	Subscription Concealed Identifier
TCP	Transmission Control Protocol
TMSI	Temporary Mobile Subscriber Identity
TLS	Transport Layer Security
UE	User Equipment
UPF	User Plane Function
URLLC	Ultra Reliable Low Latency Communications
XRES	Expected Response
gNB	gNodeB
3GPP	3rd Generation Partnership Project
5GC	5G Core

Chapter 1

Introduction

1.1 Context and Motivation

Emerging pervasive and interactive computer applications, such as Augmented Reality and Autonomous Vehicles, have raised the demand for services with high computational complexity and low latency requirements. A great effort has been dedicated to achieving these objectives. In particular, implementations utilizing 5G cellular networks promise to lead the advancements in the progress towards widely available, low-latency access to computing services for such applications. Previous research has focused on lowering the latency without altering or considering alternatives for the already existing trust mechanisms in this architecture. Implementations utilize different nodes, connected by diverse communication channels with specific protocols specifically designed for their conditions, that have each been designed to guarantee some form of trust between each other and with the user equipment, with several independent trust mechanisms. This segregating approach to implementing security measures may lead to energy and latency demands for trust mechanisms that don't align with modern low-latency goals. A lack of consideration of the different security requirements in common 5G use cases could lead to security solutions that cost more energy and time than necessary to achieve an adequate trust level.

1.2 Research Proposal

To address this issue, the thesis aims to provide a systematic analysis and characterization of the different trust mechanisms used in communications with services in a 5G environment. The goal is to identify redundancies and benefits in using different combinations of available trust mechanisms in different components and connections of a mobile cellular network. By understanding the security and overhead provided by different trust mechanisms, it will be possible to consider trade-offs for different use cases typically associated with 5G, which have different requirements for security, latency, traffic overhead, and energy consumption.

The research plan involves the theoretical analysis of the different protocols and algorithms that can be utilized in a 5G framework and their parametrizations, categorizing the security they

provide, focusing on the authentication procedure, with a secondary focus on confidentiality and integrity. After this study is completed, selected protocols and algorithms will be tested for performance and associated risk. The thesis will propose a categorization model to distinguish use cases by relevant factors to security, thereby allowing for a discussion on potential security trade-offs to favor different use cases. A non-public 5G cellular network and mathematical models are used to quantify the impacts of the different security solutions on the time overhead of connections.

By viewing the security system in 5G as a whole, instead of considering each component independently, and considering different solutions for the varied applications commonly associated with computing services, this thesis aims to explore trade-offs to aid in the progress towards meeting low-latency requirements.

1.3 United Nations Sustainable Development Goals

The Sustainable Development Goals (SDGs) are a set of global objectives established by the United Nations to address urgent social, economic, and environmental challenges. They aim to achieve a better and more sustainable future for all by 2030. This research is conducted in alignment with these goals. [14]

SDG 9 encompasses "Industry, Innovation, and Infrastructure." This thesis categorizes different industrial and healthcare use cases for 5G technology based on security factors. Various options for minimizing temporal overhead are discussed, particularly relevant for use cases such as automated driving, which rely on fast authentication procedures. SDG 12 focuses on "Responsible Consumption and Production." One aspect of this goal is the responsible consumption of energy by devices utilizing 5G technology. By aiming to employ only appropriate trust mechanisms, this thesis seeks to reduce the energy overhead of these devices.

1.4 Dissertation Structure

This dissertation is structured in eight chapters. Chapter 2 presents the state-of-the-art and background. It aims to introduce the 5G network architecture from a security perspective and explore the typical 5G use cases. Chapter 3 details the methodology used in the research. It states the exact problem under analysis, explains the motivations for the experimental setup, and categorizes the use cases based on security factors. A description of the 5G non-public network setup utilized is provided in chapter 4, with a focus on the open software utilized and the deployment process. The trust mechanisms are examined in this setup, covering a practical analysis of the security achieved in the authentication procedure, while focusing on identifying points of vulnerability and possible redundancies in security mechanisms. Chapter 5 presents the temporal analysis of this procedure, quantifying the time overhead of the different mechanisms at play, and utilizing a mathemtical model to study its variations due to propagation delays. Chapter 6 then utilizes state-of-the-art research, supported by the risk and temporal analysis, to draw conclusions on possible use-case-conscious trade-offs and solutions to minimize the overhead caused by security

mechanisms. Chapter 7 concludes the dissertation by summarizing the key findings, discussing the implications of the research, and suggesting directions for future work. References and an appendix are provided at the end of the dissertation. The appendix includes all scripts utilized during chapter 5.

Chapter 2

Background

2.1 Ecosystem Overview of 5G and Beyond

2.1.1 Foundations and Objectives

As the successor of 4G, 5G aims to overhaul the mobile network architecture to meet demands of rapidly evolving industries. Compared to previous generations, 5G promises to provide more support for a versatile network, capable of providing a diverse range of applications and services. The objective is not to mainly support smartphones connecting to the Internet, but to support a wide variety of use cases, from automated driving to inventory tracking to virtual reality.

According to [10], besides promising a tenfold increase in data rates when compared to 4G, 5G sets different requirements for its three main application areas:

- Ultra Reliable Low Latency Communications (URLLC): This category focuses on utilizing 5G for mission-critical applications. As such, it requires uninterrupted, real-time connections. 5G aims at up to 99.999% availability, latencies as low as 1 ms, at velocities of up to 100km/h.
- Massive Machine-Type Communications (mMTC): This category is designed to support a vast number of connected devices, particularly in Internet of Things (IoT) ecosystems. mMTC aims to provide connectivity for devices with diverse requirements, including ultra-low energy consumption, aiming at 10 or more years of battery life and high-density deployments of up to 1 million devices per square kilometer.
- Enhanced Mobile Broadband (eMBB): This category focuses on significantly increasing data rates. It aims to provideup to multi-gigabit speeds, enabling high-definition video streaming, virtual and augmented reality applications, and other data-intensive services.

The targets set for 5G serve as performance indicators and objectives in the ongoing development of mobile networks during the 5G generation. [10]

2.1.2 Key Use Cases

5G technology aims to satisfy differing requirements for industry-specific use cases. The diverse application areas of 5G are created to serve diverse use cases. Common use cases in 5G include manufacturing, healthcare, utilities, autonomous vehicles, and video games. A list of common use cases based on [1] is presented:

2.1.2.1 Manufacturing

As the manufacturing sector continues a digital transformation, companies actively explore 5G technology to enhance automation, monitor inventories, and manage resources. A few concrete examples are:

- Robotics: Automation can be achieved through production robots, that benefict from 5G networks for improved performance and flexibility in manufacturing processes.
- Inventory Tracking: Real-time tracking through IoT sensors connected via 5G networks can automate this process, leading to more reliable inventory tracking, that can help reduce counting errors and subsequent downtime.
- Predictive Maintenance: Sensors connected to 5G networks can be built on equipment to predict and prevent failure, possibly also before they occur, reducing unplanned downtime and maintenance costs. The use of artificial intelligence for this prediction can be done through edge computing to save resources.

2.1.2.2 Healthcare

The healthcare sector also aims to utilize 5G technology to improve patient care, with the objective of monitoring in real-time, performing remote surgeries and allowing for online consultations, improving access to healthcare.

- Online Consultations: The need for high-definition video consultations was raised during the COVID-19 pandemic. This use case becomes relevant for the consultation of experts from different areas, and in cases of safety concerns, such as pandemics.
- Remote Surgeries: By utilizing 5G's promised low-latency, robotic surgeries could allow specialists to operate on patients from different locations.
- Monitoring: The possibility of monitoring health information of a patient in real-time, transmitting its data to healthcare providers, and possibly processing it to detect anomalies, can allow for faster response times.

2.1.2.3 Utilities

Utilities are related to use cases in the public sector aiming to collect and use metrics to allow for the management of assets, such as distributed energy resources.

- Automated Metering: Smart meters can collect real-time metrics via 5G to help in the management of energy distribution and billing.
- Environmental Monitoring: Sensors connected via 5G can be used to monitor air quality, noise levels, and other environmental parameters.

2.1.2.4 Autonomous Vehicles

5G technology is essential for the deployment of autonomous vehicles, aiming to achieve the reliability and low-latency requirements necessary for vehicle-to-everything (V2X) interactions.

- Real-Time Data Exchange: Autonomous vehicles aim to utilize 5G technology to exchange time-sensitive data with other vehicles and infrastructures reliably.
- Remote Control and Monitoring: Fleets of autonomous delivery vehicles and drones can benefit from 5G technology for remote monitoring and control purposes.

2.1.2.5 Video Games

5G technology can enhance user experience, by providing lower latency and edge computing options.

- Cloud Gaming: Computing services can allow for high-quality games to users without the necessary hardware, for example for mobile gaming.
- Virtual Reality: Advancements in games that utilize virtual reality can require higher bandwidth and lower latency than other types of games, therefore benefiting from 5G technologies.

2.1.3 Standardization

Achieving compatibility across different devices and networks in the 5G environment requires standardization. The 3rd Generation Partnership Project (3GPP) is instrumental in this effort. It has been responsible for defining standards across multiple generations since 3G. Its Release 15 marked the transition between 4G and 5G in 2022 [10]. These standards present technical specifications and protocols essential for devices to function consistently across different 5G networks. The shift towards “Beyond 5G” indicates an evolution in standards.

2.2 Cellular Network Architectural Overview

Understanding the trust mechanisms within a 5G framework involves examining three key components: the User Equipment (UE), the Radio Access Network (RAN), and the 5G Mobile Core (5GC). This section breaks down these components, following the 5G architecture specifications from the 3rd Generation Partnership Project (3GPP), as outlined in [10].

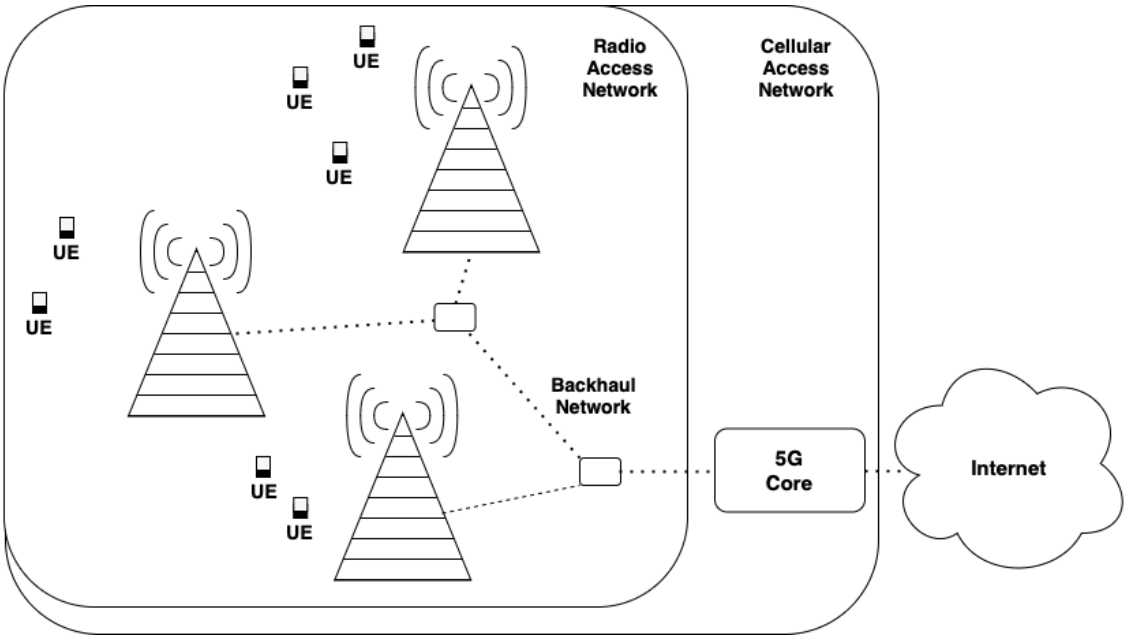

Figure 2.1: Mobile Cellular Network Architecture [6][10]

Figure 2.1 illustrates how the different components of a mobile cellular network are interconnected. The RAN is comprised of base stations, which in 5G are designated gNodeBs (gNBs). User equipments connect to gNBs, changing between them as they move to new locations. This process is known as handover and is coordinated by each gNB through direct communication with neighboring gNBs [7]. The Backhaul Network is used to connect the different gNBs and the 5GC. The 5GC connects the RAN, which covers a limited geographic area, to the global IP-based Internet and plays a defining role in the control plane by managing essential functions such as mobility, session management, and authentication. 5G can be deployed partially by integrating components into an existing 4G architecture. This is referred to as non-standalone 5G and requires additional security measures that are not necessary for a standalone 5G framework [7]. Private 5G networks are a deployment option that allows for the establishment of dedicated 5G infrastructures. These networks offer enhanced control for the specific needs of industries that opt to use them, providing a complementary option to traditional public 5G networks [10].

In a 5G network, there is a division between the control plane and the user plane. The control plane is responsible for managing network's operations, such as mobility, session management, authentication, and overall network security. The user plane is responsible for the actual transmission of application data. [10] The protocols implemented at each plane, and the security mechanisms implemented to protect their confidentiality and integrity, are distinct [7].

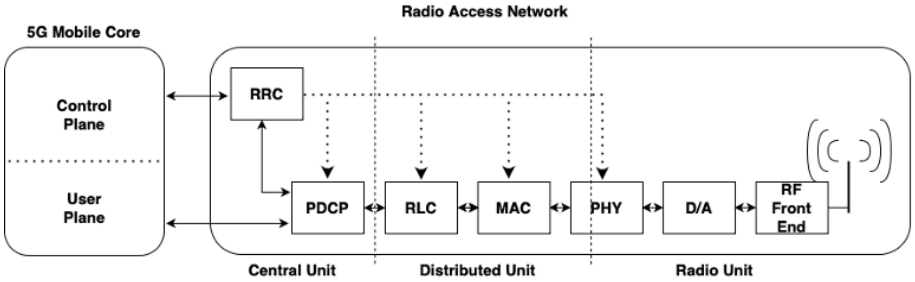

Figure 2.2: Radio Access Network [10]

2.3 Radio Access Network

2.3.1 Split Radio Access Network Model

The key stages involved in the RAN pipeline are presented in figure 2.2. The Split RAN pipeline model separates the stages between the control and user planes. There is also a separation between the Central Unit (CU), Distributed Unit (DU), and Radio Unit (RU), which provide possible points of separation of the services that a RAN provides within a gNB. [10]

Radio Resource Control (RRC) is a protocol used in the communication between the gNB and the UE. RRC is responsible for the trust mechanisms between the UE and gNB. It handles authentication procedures, encryption and integrity agreements, and handles the key management of the gNB. [7]

The Packet Data Convergence Protocol (PDCP) is utilized to communicate with the UE, handling control and user plane data. PDCP handles the ciphering of this data, conform indicated by the RRC.[7] The next three components, Radio Link Control, Media Access Control, and the Physical Layer, are responsible for the segmentation, scheduling, and transmission of data, respectively. [10]

2.3.2 Radio Resource Control Authentication Procedure

As the protocol responsible for the allocation of radio resources across all users, RRC attributes states to the connections between the gNB and UEs. Each UE changes between RRC states, depending on their specific needs at different times. When a UE is not actively using the network, it does not need a dedicated connection and only periodically listens to paging messages without occupying resources, staying in an 'RRC_IDLE' state. When a need for a connection arises, for example, by attempting to initiate a call, this state must change to become 'RRC_CONNECTED'. This transition requires an information exchange between the UE and the gNB, adding to a temporal overhead at the beginning of a connection. Establishing this RRC connection requires an exchange divided into three main steps: 'RRCSetupRequest', 'RRCSetup', and 'RRCSetupComplete'. [5]

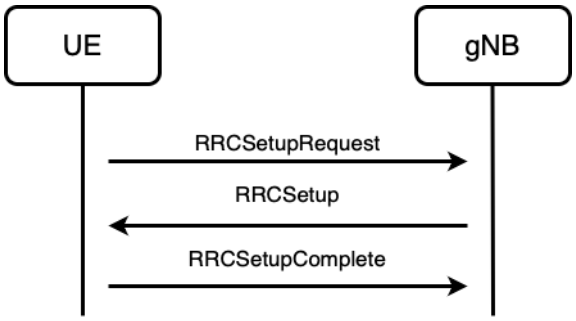

Figure 2.3: Radio Resource Control Connection Establishment [5]

The connection begins with the UE sending a message of intention, 'RRCSetupRequest'. It contains two pieces of relevant information. First, an identifier of the UE will be sent. If a Temporary Mobile Subscriber Identity (TMSI) has already been attributed by the gNB, as a the first part of the identifier, otherwise as a random value, which will be used as the UE-Identity forwardly. More information on UE identifiers is presented in section 2.5. Second, the UE indicates an 'establishmentCause', which the gNB, based on a priority list, will utilize to decide whether to create a connection when it's already running at capacity, possibly at the cost of other less important connections. This step becomes particularly relevant in cases where the temporal overhead of a connection start must be minimized and consistent. As 3GPP has left spare options on the 'establishmentCause', and the organization allows the operator to decide on the priority list between causes [5], the introduction of a specific slot for such use-cases with critical needs would not change already existing protocol:

```
1 EstablishmentCause ::= ENUMERATED {emergency, highPriorityAccess, mt-Access, mo-
    Signalling, mo-Data, mo-VoiceCall, mo-VideoCall, mo-SMS, mps-PriorityAccess,
    mcs-PriorityAccess, spare6, spare5, spare4, spare3, spare2, spare1}
```

The gNB will then respond with an 'RRCSetup' message. To address the identity of the UE, if a '5G-S-TMSI-Part1' was shared, the Part2 of the identifier will be utilized, ensuring further security by splitting the identifier in the up-and-downlink so the full exchange can't be traced as easily. Otherwise, the randomly generated number will be used. This response carries a 'radioBearerConfig', which manages the signaling (SRBs) and data radio bearers (DRBs). It can optionally also share the security algorithm configuration and key to ensure confidentiality or integrity.[5]

The final 'RRCSetupComplete' message by the UE signals that it has accepted the connection conditions, which are now agreed upon by both parties.

The security algorithm that has been decided on is effective only on non-access stratum (NAS) messaging, used for signaling. User plane messages utilize a different setup for security, which is decided on after the UE is authenticated to the 5GC, described in flowchart 2.4.

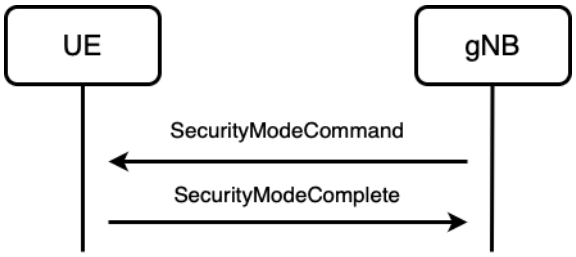

Figure 2.4: Radio Resource Control Access Stratum Security Mode [5]

2.4 5G Mobile Core

2.4.1 Internal Service-Based Structure Overview

The 5GC is designed based on a service-based architecture principle. This means that the components of the 5GC are separated by their function, i.e., the service they provide. Each of these components is a network function (NF). [10] NFs typically communicate with each other utilizing Transport Layer Security (TLS), by 3GPP's recommendation, unless the operator has chosen another form of network protection, or the interface is already physically protected. [7]

The 5GC separates its network functions (NFs) utilizing the control plane and user plane split model: The NFs in the control plane handle the establishment, maintenance, and termination of connections. The NFs in the user plane are responsible for data transfer between the User Equipment (UE) and the network. The bridge between both planes is handled by the Session Management Function (SMF), which controls each UE session by managing IP addresses, routing, and defining the quality of service [10]. The following figure 2.5 presents the connections between the NFs detailed in this section:

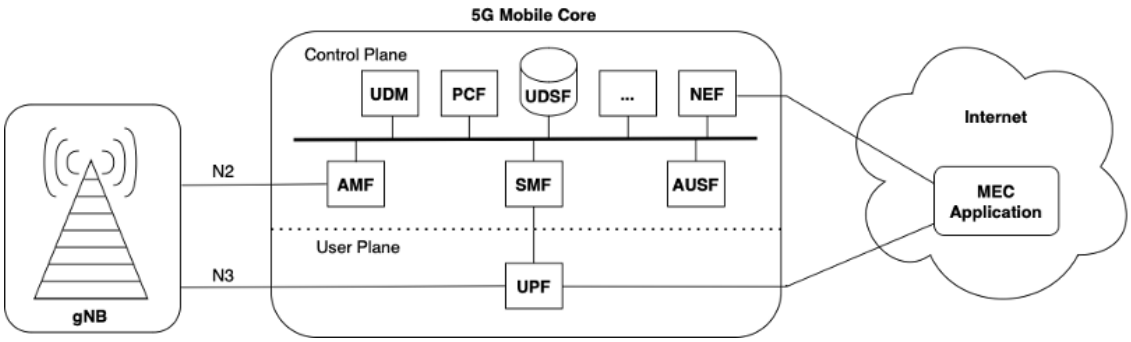

Figure 2.5: Mobile Core Architecture [6][10]

2.4.2 Access and Mobility Management Function

The N2 interface links the gNB to the Access and Mobility Management Function (AMF) [10]. It is protected by the same trust mechanism, IPsec, as the interface N3. To further increase security, it also supports the use of the Datagram Transport Layer Security (DTLS) protocol, which provides an additional layer of confidentiality, integrity, and replay protection, similarly to IPsec.

[7] The connection utilizes the Stream Control Transport Protocol (SCTP). Unlike the GTP, the SCTP is a reliable transport option, making it more suitable for signaling messages. The AMF is responsible for authorizing access and connecting a UE to the 5GC. It then tracks the UE as it connects to different gNBs. [10] The AMF additionally hosts the Security Anchor Functionality (SEAF), which is a main element in authentication, reducing latency overheads in the case of roaming. [6] Together with the SEAF, the Authentication Server Function (AUSF) and Unified Data Management (UDM) are responsible for the process of authenticating a UE [7]. The AUSF keeps track of authentications by creating and managing authentication vectors while the UDM holds the subscription and authentication information required to identify the UE. [10] Two different options are proposed for the authentication of UE: EAP – AKA' and 5G AKA. In the case of non-standalone architectures, additional procedures are utilized to ensure a secure authentication process. [7]

2.4.3 Authentication Procedure

The authentication procedure starts with the UE sending the SUCI or GUTI to the 5GC. The flow diagram 2.6 illustrates the path the identifier takes from the UE to the UDM/ARPF in the 5GC. This network function will then de-conceal the SUPI if it is concealed in the SUCI form. After identifying the user equipment it's connecting to, the 5GC chooses an authentication method. In the past, "IMSI catchers" have been used to steal the identifier of a UE by eavesdropping messages sent in the open air. In 5G. The option of using a concealed identifier, SUCI, can be used to minimize this risk, as a SUCI can only be linked to the identifiers through the de-concealing function belonging to the 5GC. This means that an attacker listening to the authentication process in the open air or impersonating the 5GC won't receive a plaintext version of the identifier. However, the option of sending the GUTI, for example, an IMSI, is also available. This leads to a potential need to encrypt the signaling in open air or risk impersonation and tracking. [7]

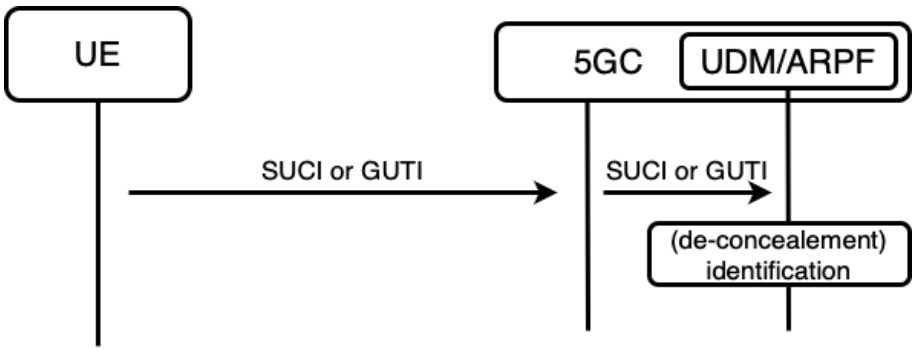

Figure 2.6: Authentication Initiation Communication Flowchart [7]

Once the UDM is aware that the UE seeks to authenticate, the UDM generates an Authentication Vector, which includes several critical elements such as the random challenge (RAND) and authentication token (AUTN).

It sends this information to the AUSt, which produces an authentication challenge for the UE, incorporating the RAND and AUTN. The SEAF then delivers the challenge to the UE. Using its stored credentials, the UE processes the challenge and generates a response (RES), which it sends back to the SEAF. The SEAF forwards this response to the AUSF for verification. The AUSF compares the received RES with the expected XRES obtained from the UDM. If the RES matches the XRES, the AUSF confirms successful authentication to the SEAF. [7] The process can be visualized in figure 2.7.

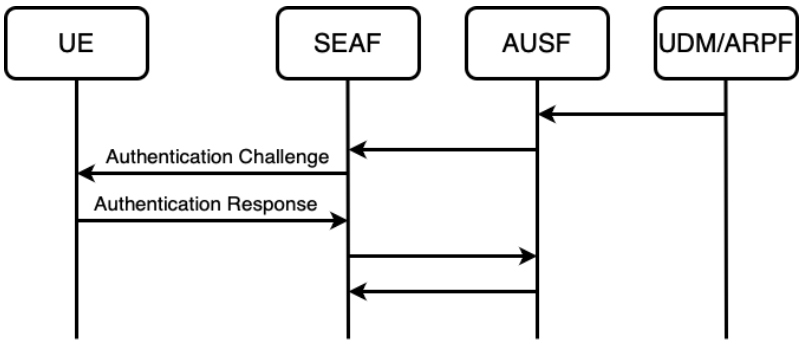

Figure 2.7: Authentication Procedure between UE and 5GC

In 5G networks, two primary authentication methods can be employed: 5G Authentication and Key Agreement (5G AKA) and Extensible Authentication Protocol-AKA (EAP-AKA). The 5G AKA method is particularly suitable for mobility and roaming scenarios due to its implementation of home and serving network interoperability. The Lightweight and Secure Access Authentication (LSAA) Scheme is a recently introduced proposal for an authentification method. It focuses on the need for mMTC devices for authentification methods with higher performance than the two already existing methods. [11]

2.4.4 User Plane Function

The User Plane Function (UPF) is the only NF in the user plane. Its main responsibility is to forward, measure, and police traffic between the RAN and the Internet. [10] The connection between the UPF and the base station, through the N3 interface, is secured with Internet Protocol Security (IPsec), with the option to ensure authentication, confidentiality, integrity, and replay protection. The specific cryptographic solutions used to protect the N3 interface are chosen by the operator. [7] The communication in this interface normally utilizes tunnels implemented through the General Packet Radio Service (GPRS) Tunneling Protocol (GTP). GTP is composed of two sub-protocols, GTP-U for transferring user data and GTP-C for controlling the tunnels.[10]

2.5 Identifiers in 5G

Each UE is identified globally by its International Subscription Permanent Identifier (SUPI). This identifier is stored in a Subscriber Identity Module (SIM) card and can take the form of either an

International Mobile Subscriber Identity (IMSI) or a Network Access Identifier (NAI). The SUPI is assigned by the operator and remains static for each UE. The UE is linked to its subscription information by the User Data Management (UDM) using the SUPI, so the 5GC can provide it with the subscriber's services. However, to avoid impersonation, the UE cannot send the SUPI in plaintext to the RAN. Instead, to authenticate itself, it encrypts the SUPI into a Subscription Concealed Identifier (SUCI) using a public key. The UDM can retrieve the SUPI from the SUCI using the Subscription Identifier De-concealing Function (SIDF). After authentication, the Access and Mobility Management Function (AMF) allocates a Globally Unique Temporary Identity (GUTI) to the UE. The GUTI is used for temporary identification by the UE without the risk of revealing the SUPI.[7] The Temporary Mobile Subscriber Identity (TMSI) is part of the GUTI and is used with the gNB to protect the identity of the UE from being stolen or tracked. It acts as a temporary identifier assigned to the UE by the network. The TMSI is used for preventing unauthorized tracking and interception of mobile subscribers. Using the TMSI, the 5G network ensures that a permanent identifier is not transmitted over the air frequently, thereby reducing the risk of identifier catchers and other privacy attacks. As the TMSI is updated frequently, it is difficult for attackers to track the UE based on their identifier. [10]

2.6 Transport Layer Security

Transport Layer Security (TLS) is widely utilized to secure communication over computer networks. It provides both confidentiality and integrity and is widely used, for example in protocols like HTTPS. TLS provides encryption, which can be tailored through different cipher suites that both parts negotiate on. For example, in some cases, only symmetric key algorithms are utilized, which could be beneficial as public-key operations can be costly. Other cipher suites might for example require preshared keys (PSK), which are relevant when mutual authentication is required. Integrity can be achieved through hash functions calculating Message Authentication Codes (MAC). [13]

2.6.1 Datagram Transport Layer Security

Datagram Transport Layer Security (DTLS) is a specific subtype of TLS to secure communication over unreliable transport protocols, such as UDP (User Datagram Protocol). For battery-powered devices, DTLS proves to be a better option, as utilizing the Transmission Control Protocol (TCP) leads to a high overhead which has been shown to lead to poor results in IoT devices.[8] The consideration of DTLS as an alternative to TLS in use-cases where unreliable communication should therefore be considered.

2.6.2 Transport Layer Security Handshake

In establishing secure communication between the user equipment and a server, the TLS handshake process is a critical mechanism that exists independently of the 5G authentication mecha-

nisms, further increasing the temporal overhead at the beginning of a connection. It involves a trade of messages between the UE and the end server to authenticate the identities of both entities. This step confirms the server's identity through a certificate and allows for the decision on the encryption method for application data.

A certificate in this context contains a public key and information about the server's identity, such as the website domain name. It is priorly signed by a third party, which is trusted by the UE and the end server. [3]

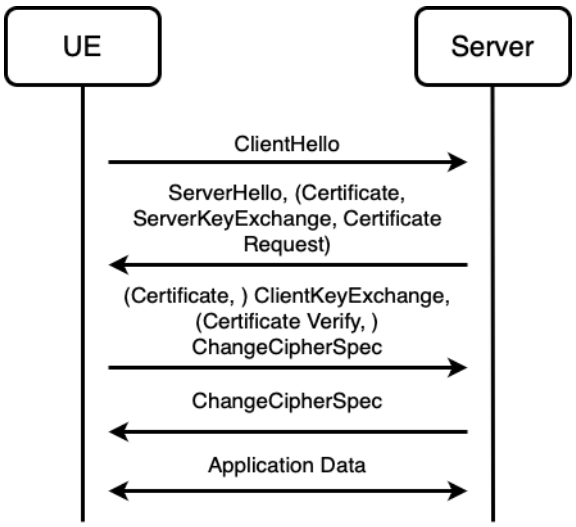

Figure 2.8: TLS Handshake Flow Chart [3]

The handshake is divided into three sections, starting with the 'clientHello' message sent by the UE. This initial contact indicates the intent to form a connection while containing a Session ID, which the server can use to resume the parameters of a past connection.

If the server can establish or resume a connection, it sends a 'serverHello' message. This message may include the server's certificate, primarily used by the UE to verify the server's identity. The server may also transmit a server key exchange message, especially if the certificate is not present or used only for identification purposes. The server can additionally request a certificate from the UE, ensuring both parties are mutually authenticated.

The UE then answers by sending the client key and, if requested, the certificate. Additionally, the UE sends a 'change cipher spec' message, indicating the encryption settings that will be used. The server then answers with its own 'change cipher spec' message, completing the handshake.

If the server holds data from a previous session in its cache and is presented by its session ID, it can skip the authentication phase by directly resuming the past connection parameters without needing to renegotiate them, enhancing the efficiency of the handshake and reducing the connection temporal overhead. In concrete terms, this method allows the connection to avoid the time of travel for one final message from the server, the processing time of verifying the certificate and calculating keys, and the energy costs attached to both. The flow diagram 2.9 presented below illustrates the shortened process. [3]

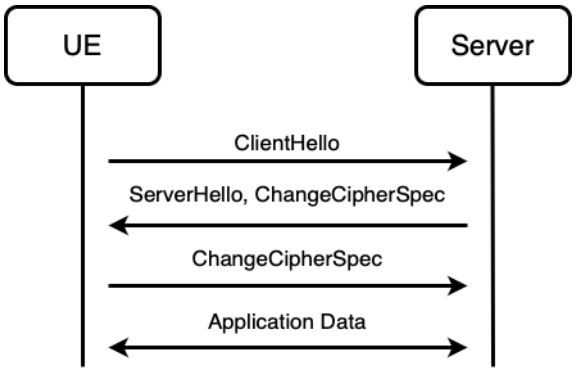

Figure 2.9: TLS Handshake Resuming Previous Session Flow Chart [3]

2.7 Encryption and Integrity Algorithms in 5G

3GPP recommends a set of algorithms to secure the connection between nodes inside the 5G framework, the New Radio Encryption Algorithms (NEAs) and the New Radio Integrity Algorithms (NIAs). These algorithms are designed to ensure the confidentiality and integrity of data. [7]

2.7.1 New Radio Encryption Algorithm

The New Radio Encryption Algorithm (NEA) is responsible for encrypting data. According to the 3GPP technical specifications, several encryption algorithms are available for NEA, including but not limited to:

Algorithm	Description
NEA0	No encryption
NEA1	Based on SNOW 3G, a stream cipher
NEA2	Based on Advanced Encryption Standard (AES), a widely used block cipher, adapted to be a stream cipher
NEA3	Based on ZUC, a stream cipher designed for LTE and 5G

Table 2.1: New Radio Encryption Algorithms

NEA0 specifies that no encryption is applied. When specified in the technical specifications that an encryption method must be chosen, this option is included, meaning that encryption is not required by 3GPP directly.

2.7.2 New Radio Integrity Algorithm

To ensure the integrity of the data, NIA utilizes message authentication codes (MACs), which involve sending a cryptographic hash of a message to guarantee its integrity. Using a secret key to generate this value, the sender creates a MAC that is transmitted with the original message. Upon receipt, the receiver uses the same secret key to generate a new MAC from the received message. The receiver then compares this newly generated MAC with the received MAC. If the two values

match, it confirms that the message has not been altered and verifies the sender's authenticity. If the message had been altered or the sender's identity had been impersonated, the newly generated MAC would no longer match the received MAC, indicating a failure in integrity verification.

Algorithm	**Description**
NIA0	No Integrity
NIA1	Based on SNOW 3G
NIA2	Based on AES-CMAC, which uses AES in conjunction with the Cipher-based Message Authentication Code
NIA3	Based on ZUC

Table 2.2: New Radio Integrity Algorithms

Similar to encryption, NIA0 specifies that no integrity protection is applied.

2.8 Related Work

2.8.1 Authentication

The paper "A Formal Analysis of 5G Authentication" [2] provided an in-depth overview of the 5G AKA protocol, including the role of the identifiers SUPI and SUCI in authentication. Compared to this work, the paper also focuses on the distinction between home networks and serving networks and how they might affect security during authentication depending on which one the UE is connecting to. The study details 5G AKA steps, to utilize a protocol analysis tool to access its security. The paper found that the protocol might be susceptible to attacks such as replay and man-in-the-middle attacks. The findings are relevant to this work, as they show that there is still an inherent security risk in the protocols used in 5G, and present a structured analysis of one of the security mechanisms considered during this research project.

2.8.2 Encryption and Integrity

A survey on lightweight cryptographic algorithms [11] presents 11 alternatives to the used cryptographic algorithms, comparing them on performance and security levels. These algorithms offer faster encryption with less power consumption, although offering weaker security than the most used encryption methods. Although some of these methods can fall to sophisticated cryptographic attacks, they resist many common attacks and protect against casual eavesdropping. One of the examples, CLEFIA, has been widely studied and is considered secure against differential, linear, and related-key cryptanalysis. While highly confidential information will not be guaranteed protection through lightweight encryption algorithms, this information is relevant in the context of this thesis, in cases where mitigating the risk of casual eavesdropping and protecting against attackers who are eavesdropping on many connections simultaneously is the priority.

Chapter 3

Methodology

3.1 Research Approach

3.1.1 Problem Statement

In the pursuit of enabling new use cases with stricter requirements, 5G looks to introduce more flexibility into mobile networks, differentiating, for example, between services aimed at mMTC to maximize battery life at the cost of lower data rates and higher latency, and URLLC, to meet requirements of extremely low latency and high reliability. Security in 5G has also been improved compared to its predecessor, with increased options for identifiers and securing of signaling and application data between every possible entity, from the UE to gNBs, 5GCs, and end servers. To secure communication, integrity and encryption options are being negotiated between the different entities. The freedom to choose from different security measures, as introduced by the 3GPP standards, presents a unique set of opportunities and challenges. This thesis considers these by identifying how far each security measure is relevant for different use cases and how temporal and energy overheads can be minimized by choosing appropriate levels of security. Considerations are presented on how the different actors involved in 5G - end-users, operators, open-source contributors such as OpenAirInterface (OAI), and 3GPP — can take action to ensure appropriate levels of security is being achieved. The focus is on utilizing the flexibility in 5G security, and at certain points considering modifications to the methods in which it is implemented, so it can aid in the goal of creating a use case appropriate approach to mobile communications, conscious of diversity in energy, temporal, and security needs.

3.1.2 Proposed Approach

To analyse this problem, a work plan divided into three work objectives was set up: a theoretical analysis of trust mechanisms in 5G; a deployment of a non-public 5G network and the assessment of its security; and a temporal analysis of the impact of the different trust mecanisms in the temporal overhead of a connection. The first step involved analying a set of sources to understand the various security options of 5G.

In order to achieve this, an experimental assessment of security mechanisms, as detailed in chapter 4, guided the analysis. Specific points of encryption and integrity are detailed separately in the technical specifications of their respective protocols. As such, a parallel approach between an experimental analysis and a theoretical analysis aided in identifying and understanding these mechanisms. The background chapter presents the results of this task. The research was centered around the technical specifications of the security architecture for 5G [7], published by 3GPP, which details the authentication processes with the 5GC, the interfaces connecting the nodes and their security options, alongside the approved encryption and integrity methods. Additional sources were needed to explain the remaining aspects of the authentication procedure in 5G. Information was gathered on identifiers used within the 5GC and their purpose[10]. The technical specification of the RRC protocol [5] was utilized to detail authentication procedures with the gNB. An analysis of the TLS handshake was considered to understand the security mechanisms involved in this type of connection commonly used in 5G environments. The result allowed for a transition away from having the information on security mechanisms presented in a segregated approach, with security measures specifications originally separated between the RAN, 5GC, and the end server. This approach allows for the proposed analysis of the security options available of the 5G framework as a whole, and a discussion on the impacts of the current 5G ecosystem, from 3GPP to operators and end-consumers, on the implementation of these options.

This preliminary study pointed to evidence of possible redundancies between choices of security mechanisms involved in connections over 5G. 3GPP opens options of implementing encryption betweeen the UE and gNB, and between the gNB and 5GC, for both the data plane and the user plane, which might, in certain use cases, become redundant when encryption between the UE and 5GC, and between the UE and end server, is also available. The problem becomes exacerbated through the separation of choices related to security into the operator and end consumer, as the end consumer does not have knowledge of the choice of encryption in the tunnelling between gNBs and 5GCs. This could cause the UE to choose to encrypt for example signalling data both to the gNB and 5GC, leading to redundancy if encryption between the gNB and 5GC has already been established. The extent of this redundancies, and its impact in the negotations for security measures, depends on the temporal overhead that each of them imply, and, as such, this thesis aims to review these timings on a deployment of a 5G network. In order to do that, a timeline of all security mechanisms negotiated during the authentication phase, and tunnelling options that result from them, is presented in section 3.2.

To complement this analysis, a categorization, presented in section 6.1, splits the use cases by core factors. This categorization provides a structured approach to analyze the choice of security mechanisms, tailoring it to specific use case requirements. It helps in understanding the diverse needs and constraints of different applications, thus guiding the selection of appropriate security measures.

In order to quantify and understand the impact of these trust mechanisms in the time overhead of a connection, the timings of the authentication phase were measured, and its resuts presented in chapter 5. A mathematical model is used to consider three different scenarios of possible delays:

between the UE and gNB, between the gNB and 5GC, and between the 5GC and an end server. This allows for a contextualization of how diverse environments of the network can influence the impact of the trust mechanisms. The following section 3.2 presents the different steps timed during the authentication phase, and their purpose and structure.

3.2 Trust Mechanisms Established in Authentication Phase

The authentication procedures and establishment of encryption and integrity algorithms for the different interfaces can be analysed through the experiment. Their order is presented in the flow diagram 3.1. As the UE starts in an RRC_IDLE state, it must first (a) form a connection to the gNB through the RRC Connection Setup. At this stage, the encryption and integrity algorithms for the signalling between the UE and gNB are chosen. The 5GC then (b) requests the UE to identify itself and (c) to prove its authentication through an authentication challenge. At this point, (d) the encryption and integrity algorithms for the signalling messages between the 5GC and the UE are chosen. After the 5GC internally processes the request for a connection, if approved, it signals the gNB (e) to define the encryption and integrity algorithms for the data plane with the UE. (f) The gNB and UE then configure the radio bearers for the connection. The authentication procedure to the 5GC ends with the UE (g) as it signals that the NAS registration is complete. After the authentication with the 5G network entities is complete, (h) the TLS handshake procedure occurs with the end server.

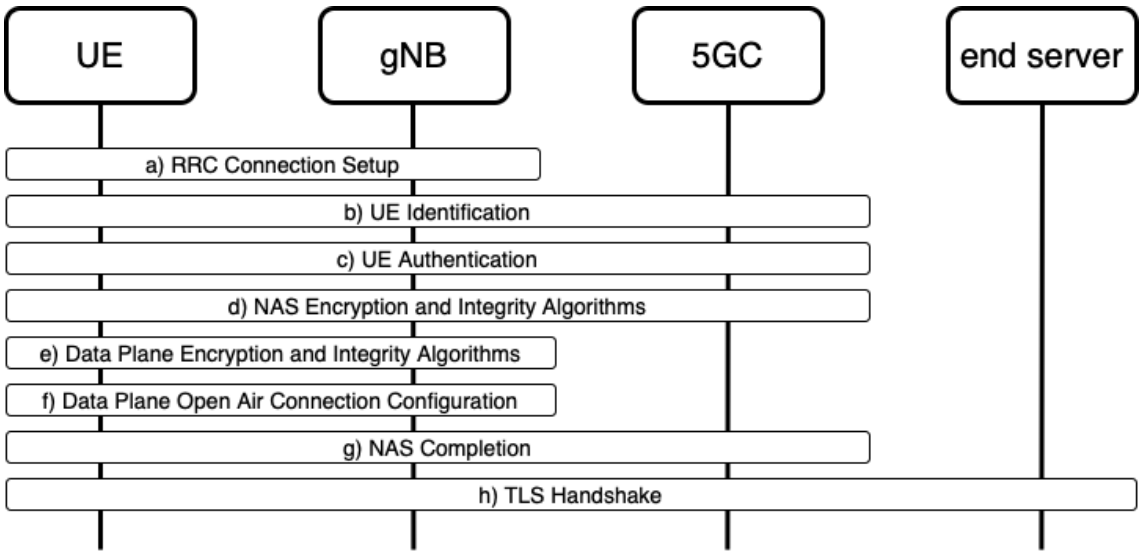

Figure 3.1: Authentication Procedure Order in Flow Chart

Figure 3.2 illustrates the encryption and integrity channels that are negotiated during the UE initial authentication process. The upper side of the diagram shows the control plane, comprised of signaling messages. In this plane, the UE has the option of setting up encryption and integrity mechanisms in communication to the 5GC and gNB. In addition, the SCTP channel connecting the gNB and 5GC may also have encryption or integrity mechanisms, although unknown to the

UE. The activation of encryption in all of these connections could lead to double encryption to signaling messages destined at the 5GC. In the data plane, the UE can decide on encryption to the gNB and to the end server, established through the TLS handshake. Encryption on the GTP tunnel between the gNB and 5GC is also possible, although not decided on by the UE. The possibility of information being encrypted twice during the open air and backhaul connections is therefore also possible for the data plane.

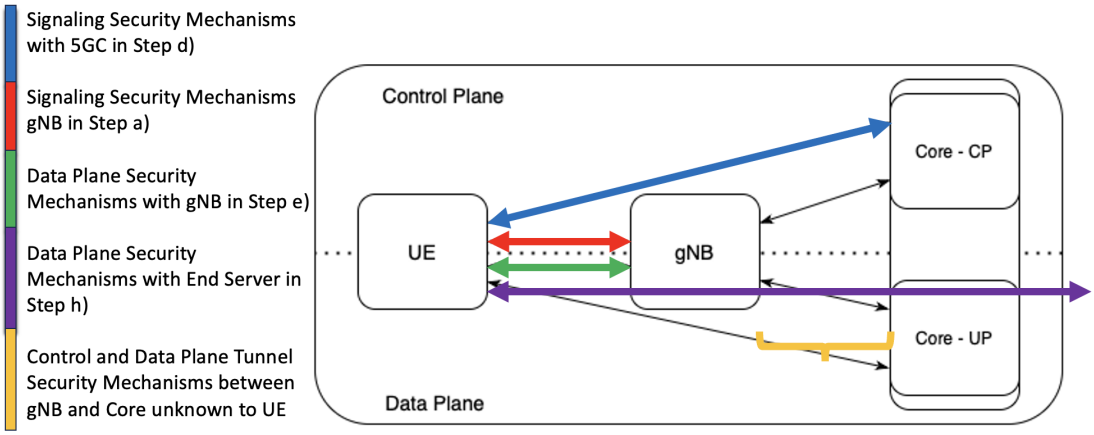

Figure 3.2: Encryption and Integrity Mechanisms Setup During Authentication by UE

3.2.1 Start and Stop Points for Timing Measurements

Step	**Start Point**	**End Point**
A	RRC Setup Request sent from UE	RRC Setup Complete received by CU
B and C	Identity Information sent from CU	Authentication Response received by 5GC
D	NAS Security Mode Command sent from 5GC	NAS Security Mode Complete received by 5GC
Step E	Security Mode Command sent from CU	Security Mode Complete received by CU

Table 3.1: Start and End Points for Authentication Procedure Timings

The first step, A, responsible for creating the connection utilized in signalling messages between the UE and both the gNB and 5GC, is measured as the time from when the UE sends the request to when the CU receives the confirmation that the setup is complete.

Step B and C are responsible for the identification and authentication of the UE with the 5GC. Since the purpose of both of these steps are related, they are measured together. The prior analysis

shows that, deviating from the identity request and identity response exchange, the identity information is sent by the CU to the 5GC, as it is received by the CU from the 'RRC Setup Complete' packet. The timing is therefore measured between this registration request sent by the CU, and the reception of the authentication response by the 5GC.

Step D is responsible for the decision on encryption and integrity mechanisms for signalling messages between the UE and the 5GC. It is measured in the 5GC, as the time difference between the NAS security mode command packet is sent, and its answer received.

Likewise, Step E is responsible for the decision on encryption and integrity mechanisms for the data plane in open air. It is measured in the CU, as the time difference between the NAS security mode command packet is sent, and its answer received.

All steps are measured at the arrival of the packets through wireshark.

Chapter 4

Experimental Assessment of Security Mechanisms

4.1 Open-Source Software Utilization

To analyze the deployment of a 5G network and focus on its security aspects, this master's thesis relies on the utilization of open-source software solutions. The choice to use open-source software is rooted in their significance in 5G: Ideally, the transparency and accessibility of open-source platforms are a source for rapid implementation and testing, which is crucial to addressing the evolving demands of 5G use cases. However, the open-source software available for 5G is still largely under development and, therefore, limited in concrete also in the choice of encryption, integrity, and authentication mechanisms that have already been implemented.

In concrete, the deployment utilizes the following setup, which follows a widely used format combining two complementing softwares:

- OpenAirInterface (OAI), used to simulate the user equipment, and the gNB, divided into the DU and CU.
- Open5GS, which implements the 5GC in the service-based architecture approach, with different network functions contained separated and implemented in separate dockers.

4.2 OpenAirInterface Structure

The OpenAirInterface (OAI) codebase is structured in directories which divide the software, each serving specific functions for the development, deployment, and execution of the 5G network components. Understanding this structure is essential for understanding the results and modifying code. Below is an official description of the OAI directory structure [1]:

[1] Source: README of the OpenAirInterface repository, found in https://gitlab.eurecom.fr/oai/openairinterface5g/-/blob/master/README.md, last checked on June 26, 2024.

Directory	Description
charts	
ci-scripts	Meta-scripts used by the OSA CI process. Contains also configuration files used day-to-day by CI.
CMakeLists.txt	Top-level CMakeLists.txt for building
cmake_targets	Build utilities to compile (simulation, emulation, and real-time platforms), and generated build files.
common	Some common OAI utilities, some other tools can be found at openair2_UTILS.
doc	Documentation
docker	Dockerfiles to build for Ubuntu and RHEL
executables	Top-level executable source files (gNB, eNB, ...)
maketags	Script to generate emacs tags.
nfapi	(n)FAPI code for MAC-PHY interface
openair1	3GPP LTE Rel-10_12 PHY layer / 3GPP NR Rel-15 layer. A local Readme file provides more details.
openair2	3GPP LTE Rel-10 RLC_MAC_PDCP_RRC_X2AP + LTE Rel-14 M2AP implementation. Also 3GPP NR Rel-15 RLC_MAC_PDCP_RRC_X2AP.
openair3	3GPP LTE Rel-10 for S1AP, NAS GTPV1-U for both ENB and UE.
openshift	OpenShift helm charts for some deployment options of OAI
radio	Drivers for various radios such as USRP, AW2S, RFsim, ...
targets	Some configuration files; only historical relevance, and might be deleted in the future

Table 4.1: Directory Structure and Descriptions

4.2.1 OAI Structure Separated by OSI layers

Linking the directories of the OAI software to their corresponding OSI layers allows for a better understanding of how each of them is implemented, aiding in the analysis of the security mechanisms and in identifying the origin of errors and warnings in the logs of different layers.

The physical layer is implemented in ‘openair1’, based on the release 15 of 3GPP, which covers the standalone version of 5G, and by ‘radio’, which covers the drivers that enable the usage of hardware radio equipment.

The data link layer is implemented in ’openair2’ as well as ’nfapi’. ’Openair2’ contains the protocols utilized in this layer, such as Radio Resource Control (RRC). Radio Link Control (RLC) for the control and data plane between the UE and gNB, as well as Packet Data Convergence Protocol (PDCP) and Medium Access Control (MAC). The directory ’nfapi’ implements the interface that connections this layer to the physical layer.

The network layer is implemented in ’openair3’. Of note, it manages two crucial protocols in the authentication procedure, the S1 Application Protocol (S1AP) and Non-Access Stratum (NAS). The directory additionally contains the tunneling protocols.

The application layer is present in ’openshift’, containing the deployment options for OAI,

and the 'executables' directory, which contains the source files for the various network elements. Of note are the executables 'nr-softmodem', used to run the gNB subdivided in the DU and CU, as well as the 'nr-uesoftmodem', running the user equipment simulator utilized in this project.

4.3 Deployment Map

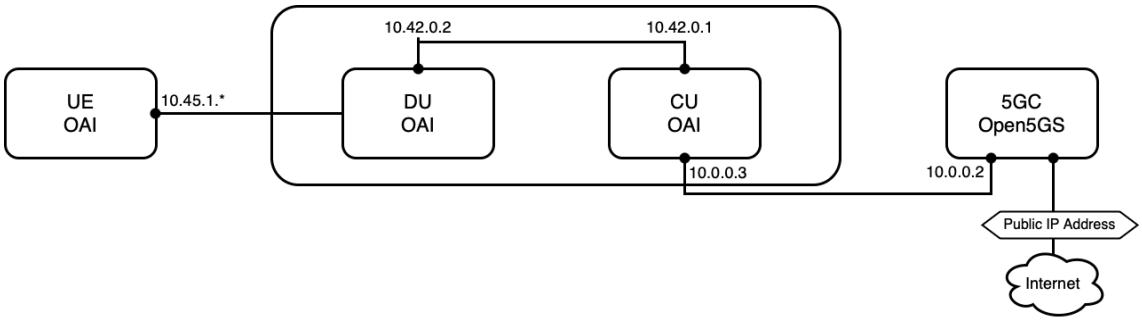

Figure 4.1: Deployment Setup

The deployment setup, as illustrated in Figure 4.1, presents the interconnections between the different entities (UE, DU, CU, and 5GC) and their corresponding IP addresses, providing context for the results and analysis of packets. The interface connecting the UE to the DU is created through the authentication procedure on demand by the UE, at which point a random IP address in the range between 10.45.1.0 and 10.45.1.255. Between the DU and CU, the IP addresses 10.42.0.2 and 10.42.0.1 are used, both for the control and data planes. Between the CU and the 5GC, similarly, the control and data planes, in the form of the SCTP and GTP tunnels, both utilize the 10.0.0.3 and 10.0.0.2 IP address combination.

4.4 Building Process

The DU, CU, and UE are installed on Ubuntu 22.04. The executables utilized were built using the following commands, executed in the 'openairinterface/cmake_targets' directory for all three entities. A rebuild is necessary for changes in code, however it is not for changes in the configuration files.

```
./build_oai -w USRP --gNB
./build_oai -w USRP --nrUE
```

The first command compiles the necessary binaries for the CU and DU, the second one for the UE.

4.4.1 Centralized Unit (CU)

The CU is executed using the following command from the openairinterface/cmake_targets/ran_build/build directory:

```
sudo ./nr-softmodem -O ../../../targets/PROJECTS/GENERIC-NR-5GC/CONF/
cu_gnb.conf --sa -E --continuous-tx
```

The command specifies that the CU should be initiated according to the configuration file 'cu_gnb.conf'. The –sa flag indicates standalone mode, -E –continuous-tx ensures continuous transmission.

The 'cu_gnb.conf' configuration file includes identifying information of the gNB, as well as the toggling options for logs from different protocols and layers, which are then used in the analysis. Of particular interest are also the following security settings:

```
security = {
  # preferred ciphering algorithms
  # the first one of the list that an UE supports is chosen
  # valid values: nea0, nea1, nea2, nea3
  ciphering_algorithms = ( "nea0", "nea2" );

  # preferred integrity algorithms
  # the first one of the list that an UE supports is chosen
  # valid values: nia0, nia1, nia2, nia3
  integrity_algorithms = ( "nia2", "nia0" );

  # setting 'drb_ciphering' to "no" disables ciphering for DRBs, no matter
  # what 'ciphering_algorithms' configures; same thing for 'drb_integrity'
  drb_ciphering = "no";
  drb_integrity = "yes";
};
```

The priority in ciphering algorithms and integrity algorithms, from left to right, will then be compared with the UE capabilities in the RRC setup phase, until the first match available is found, to protect signaling data. The 'drb_ciphering' and 'drb_integrity' options allow for the same protocols to be used in the data connection as it travels in open air between the UE and the gNB, as described in section 2.3.2.

4.4.2 Distributed Unit (DU)

The DU was executed using the following command from the openairinterface/cmake_targets/ran_build/build directory:

```
sudo ./nr-softmodem -O ../../../targets/PROJECTS/GENERIC-NR-5GC/
CONF/du_gnb.conf --sa -E --continuous-tx --T_stdout 2 --T_nowait
-q
```

This command specifies that the DU should be executed using the configuration file 'du_gnb .conf'. The –sa flag indicates standalone mode, -E –continuous-tx ensures continuous transmission, –T_stdout, –T_nowait disables wait times in transmission. The configuration file defines the

identifying values of the DU, available logs, as well as physical specifications for the radio. No encryption or integrity options are provided in this configuration file.

4.4.3 User Equipment (UE)

The UE was executed using the following command from the openairinterface/cmake_targets/ran_build/build directory:

```
sudo ./nr-uesoftmodem -O ../../../targets/PROJECTS/GENERIC-NR-5GC/
CONF/ue.conf -r 106 --numerology 1 --band 78 -C 3619200000 --ue-fo
-compensation --sa -E --ue-rxgain 80
```

The command specifies that the UE should be initialized using the configuration file 'ue.conf'. Specific information for the physical layer is defined, such as the bandwidth, numerology value, frequency band, downlink carrier frequency, frequency offset compensation, and receiver gain. The UE runs in standalone mode (–sa).

The 'ue.conf' file includes relevant information for security. In particular, it contains the IMSI, key and OPC parameter combination associated with the UE and shared with the 5GC for identification and authentication purposes through the 5G-AKA process. Additionally it also holds the International Mobile Station Equipment Identity Software Version (IMEISV), to identify the model and software of the UE.

4.4.4 5G Core (5GC)

The 5GC was implemented using the Open5GS software and continuously operated to manage core network functions. The relevant configuration for the 5GC security parameters is in the `amf.yaml` file, with the following security settings:

```
1 integrity_order: [ NIA2, NIA1, NIA0 ]
2 ciphering_order: [ NEA0, NEA1, NEA2 ]
```

4.5 Specifications

This subsection presents hardware specifications for the equipment used in our experimental setup, divided into the DU and CU running in containers inside the same system, the UE, and the 5GC. By detailing these specifications, the aim is to provide a clear context for the results of the project, ensuring that the performance and outcomes can be appropriately understood and evaluated.

4.5.1 Distributed Unit and Centralized Unit Specifications

The DU and CU are hosted in two distinct computers with the following specifications:

- **CPU:** 1x Intel Core i9 13900K 24-Core (2.2GHz-5.8GHz)
- **Memory (RAM):** 32 GB
- **Motherboard:** Asus TUF Z790-Plus Gaming D4
- **Graphics Card:** Gigabyte GeForce® GTX 1650 D6 OC Rev.2 4G
- **Network Interface:** 2x WL-PCI Express TP-Link TX401 | 10GB PCI Ethernet

4.5.2 User Equipment Specifications

The UE in this experiment is implemented in a high-performance laptop with the following specifications:

- **Model:** Legion Pro 5 16IRX9
- **CPU:** 14th Gen Intel® Core™ i9-14900HX
- **Graphics Card:** NVIDIA® GeForce RTX™ 4070

4.5.3 5G Core Specifications

The core network functions are hosted on a dedicated server with the following specifications:

- **Model:** Dell PowerEdge R710

4.6 Experiment Design

4.6.1 Data Collection

To initiate the experiment, the DU, CU and 5GC were started as described in 4.6.2. Once the starting procedures for all software were completed, as confirmed by their respective logs, and the tunnels were set up, the UE was started.

The 'tcpdump' command was used to monitor packets in the UE, DU, CU, and 5GC. Wireshark and the packet structures detailed in the 3GPP technical specifications [5] [7] were utilized to analyze security mechanisms and their setup points. This allowed for a deduction of security risks associated with a network associated with this deployment.

By observing the UE logs, the completion of the UE authentication process was confirmed. In this case, a connection to an end server, 'www.google.com,' was manually initiated with the following command, which establishes a connection to download the Google logo:

```
1 wget --bind-address=10.45.1.8 https://www.google.com/images/branding/googlelogo/1x/
    googlelogo_color_272x92dp.png
```

4.6.2 Deployment Procedure

The deployment of the CU, DU, UE, and 5GC was carried out as follows:

1. **Start the 5GC:** The core network was initiated first to ensure it was ready to handle connections from the CU and UE.
2. **Run the CU and DU:** The CU and DU were started simultaneously. The logs were monitored to ensure both components initialized correctly and established a connection.
3. **Start the UE:** After confirming that the CU and DU were fully initialized and connected, the UE was started. The UE configuration was set up to authenticate with the core network.

4.7 Analysis of Security Mechanisms

4.7.1 RRC Connection Setup

The RRC Connection Setup initiates several elements necessary for the connection between the UE and gNB. Of note, the method of encryption and integrity for open-air signaling messages is defined here. The indication of an 'establishment cause' is utilized to define the priority of the connection and, therefore, relevant to reducing the connection temporal overhead in time-critical use cases. Lastly, the option of identifier used for the gNB, either the TMSI or a random value, is also present here.

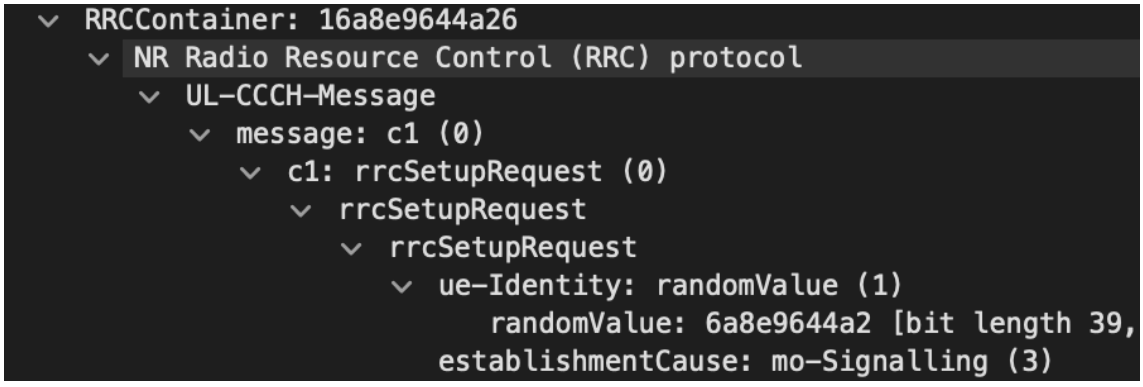

Figure 4.2: RRC Setup Request Wireshark Capture

The RRCSetupRequest, presented in figure 4.2, shows that the establishment cause of the connection is defined as 'mo-signalling', used, amongst other cases, when a UE is newly registering to an unknown gNB. 3GPP does not specifically define how the priority between different causes is handled. The identifier utilized a random value, as a TMSI has not yet been attributed.

The RRCSetup message has the main purpose of configuring the radio bearers. The radioBearerConfig item of the packet, shown in 4.3, is relevant for this analysis. Typically, it carries the 'securityConfig' value, which is then used to decide the encryption and integrity options for the

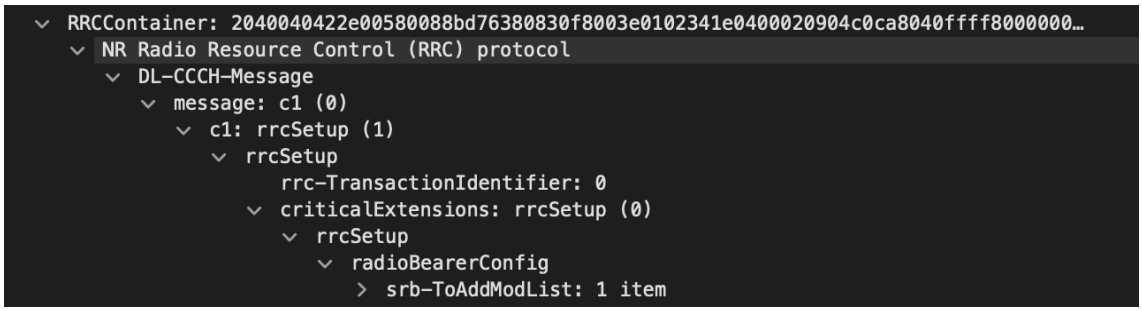

Figure 4.3: RRC Setup Wireshark Capture

Figure 4.4: RRC Setup Complete Wireshark Capture

signalling messages in open air. The lack of the optional value in the experiment indicates that no security option is being used, and that the connection is not secured in this interface.

The RRCSetupComplete message, 4.4, appears as expected completing the sequence. The MAC value, defined as 'MAC=0x00000000', indicates that, as expected by the lack of security mechanisms chosen in the prior message, integrity is not being guaranteed in the default state of this OAI setup. This happens as the encryption mechanism chosen for the RRC connection is not yet implemented in the UE side 4.8.

4.7.2 UE Identification

In the implementation of OAI, the identifier is sent directly with the RRC Setup Complete message to save time. The Centralized Unit CU then forwards this to the 5GC. Figure 4.5 show the identifier being transmitted unencrypted over the air in the message. At this stage, the details of the encryption and integrity methods that the UE can use are also sent, as shown in 4.6.

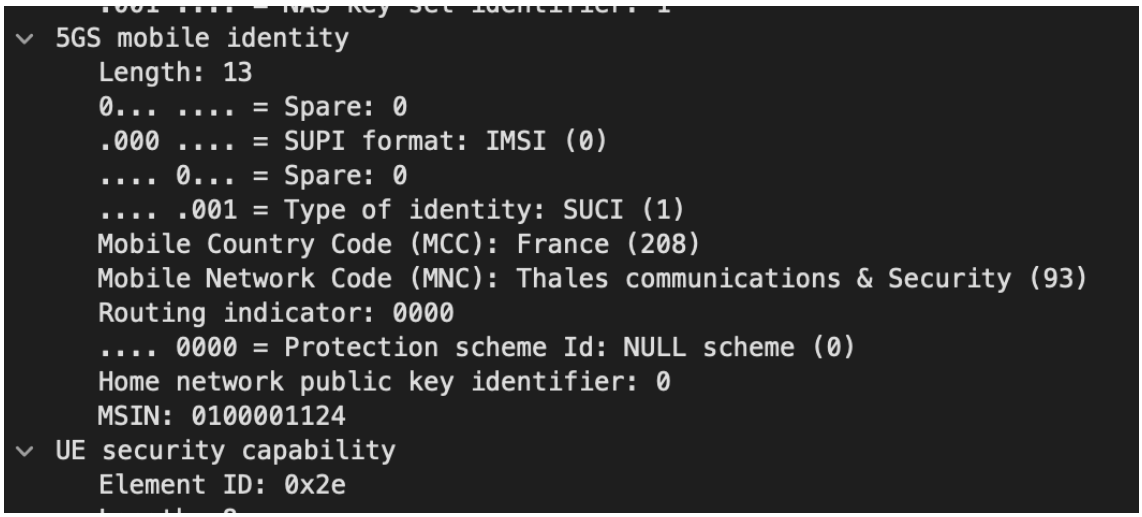

Figure 4.5: Identifier Details Sent Unencrypted shown in Wireshark Capture

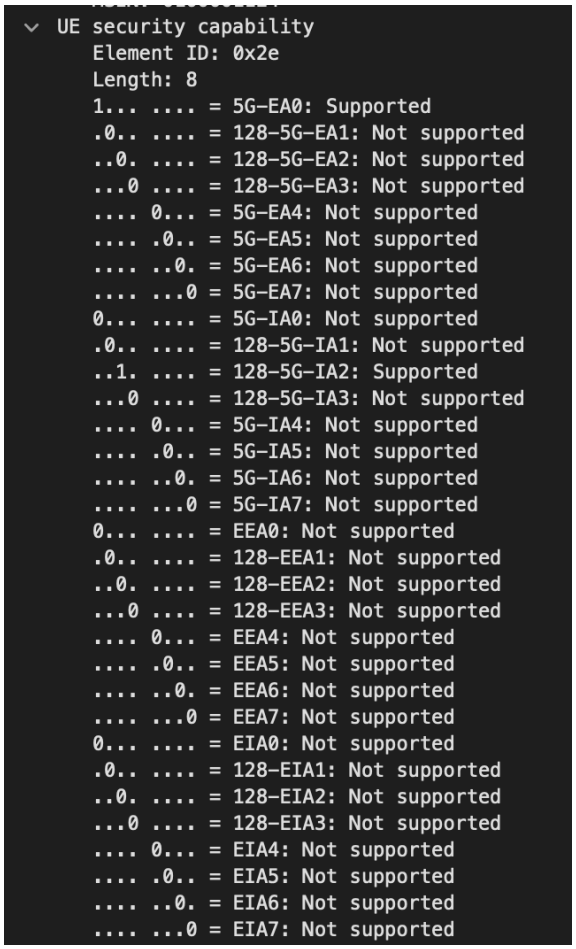

Figure 4.6: User Equipment Reported Encryption and Integrity Capabilities

4.7.3 UE Authentication

After the UE announces its identity, the 5GC sends an authentication request packet, with an authentication challenge. The authentication challenge is composed of two parameters, as shown in 4.7: a random value and an authentication token. The UE utilizes these to calculate the response parameter, which it sends to complete the authentication in 4.8.

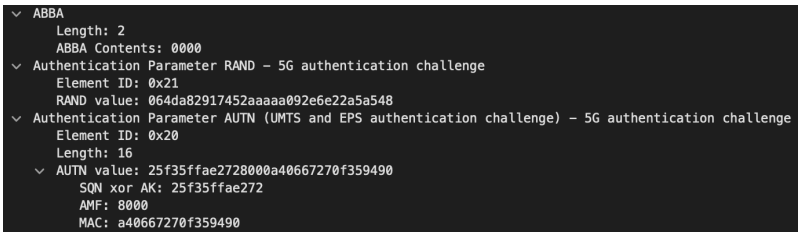

Figure 4.7: Authentication Challenge in Authentication Request Wireshark Capture

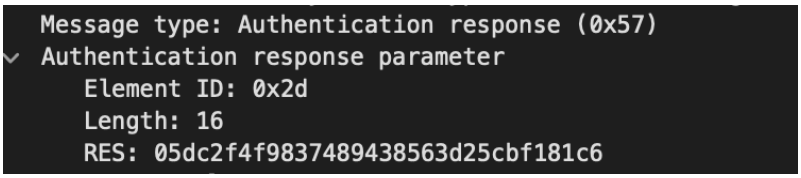

Figure 4.8: Authentication Challenge Result in Authentication Response Wireshark Capture

4.7.4 NAS Encryption and Integrity Algorithms

Upon having accepted the challenge and therefore confirmed the identity of the UE, the encryption and integrity algorithms for NAS are now decided on. As the 5GC sends the decision on those algorithms, it attaches a copy of the UE security capability to ensure that the UE capabilities it received haven't been altered. In the connection using the default setup by OAI, only integrity is secured through NIA2, whereas no encryption method is utilized.

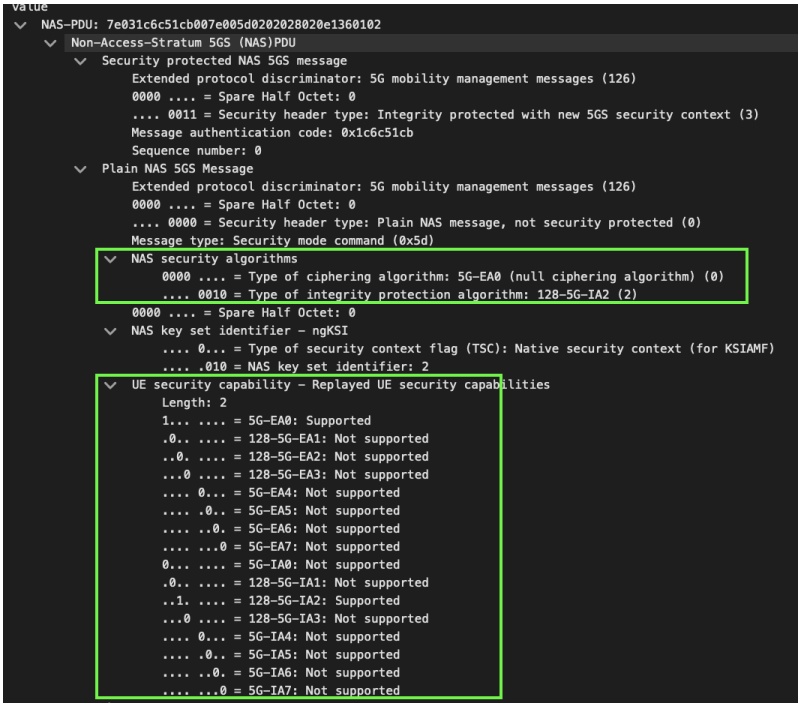

Figure 4.9: NAS Security Mode Command Wireshark Capture

The UE confirms this by sending back an NAS Security Mode Complete packet.

4.7.5 Data Plane Encryption and Integrity Algorithms

The Data Plane Encryption and Integrity Algorithms negotation starts in parallel with the NAS Encryption and Integrity Algorithms negotiation. The UE processes and answers the RRC only after answering the negotation for signalling with the 5GC, as displayed in the following figure 4.10:

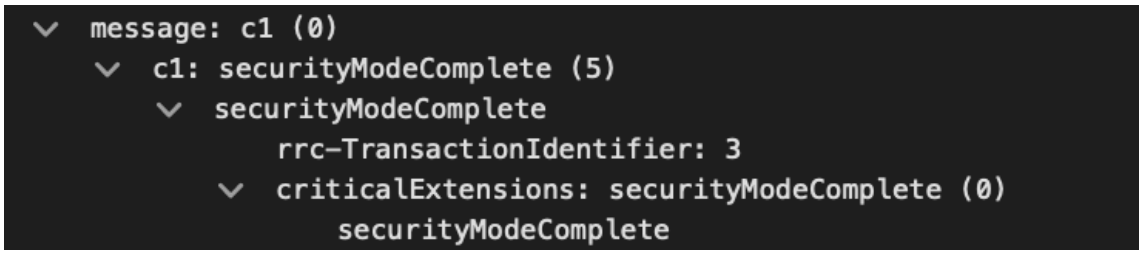

Figure 4.10: Security Mode Complete in Wireshark Capture

4.7.6 Data Plane Open Air Connection Configuration

With the encryption and integrity options established, the gNB and UE proceed to agree on the new connection for the data plane, through an RRC Configuration and RRC Configuration Complete packet exchange.

Figure 4.11: RRC Reconfiguration and RRC Configuration Complete in Wireshark Capture

4.7.7 NAS Completion

The UE acknowledges to the 5GC that the data plane connection is established, completing the authentication proccess.

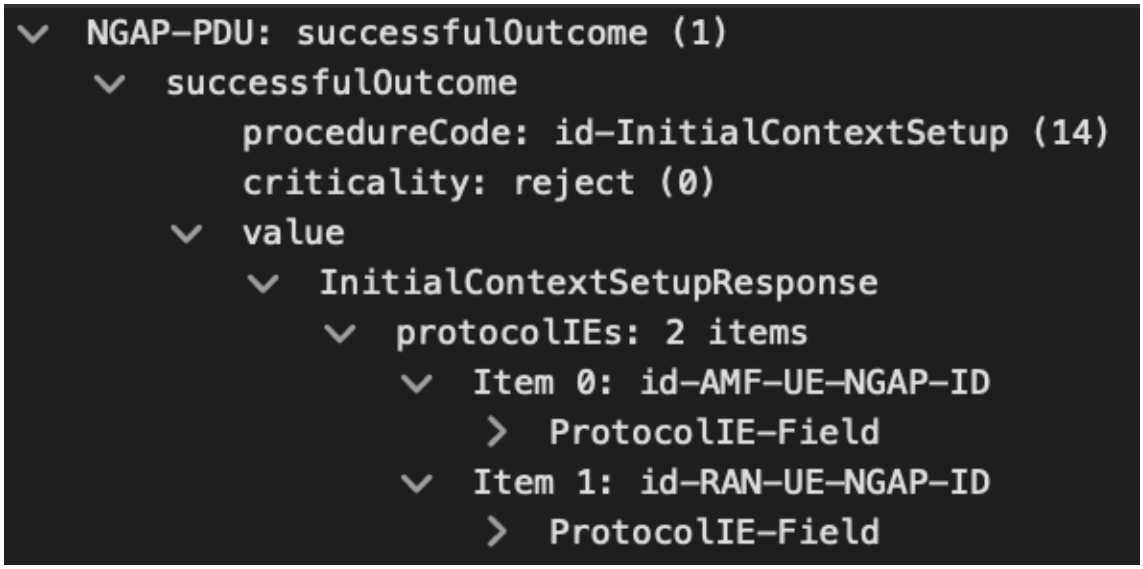

Figure 4.12: UE confirms Connection to 5GC, ending the Authentication Phase

4.8 New Version of the UE

Since April, the latest release of the UE software in the main branch was not functioning correctly in the used setup. The process of identifying this issue began with an error indicating that the send queue was full. Initially, it was suspected that this was due to changes in the physical layer, which had been restructured compared to the running version. However, logs showed that the error always occurred at the same stage of the authentication process, indicating that the problem was connected to it.

```
1 [NR_MAC]   Got NACK on NR-BCCH-DL-SCH-Message (SIB1)\ \ Assertion (i< 20) failed!\
      In writerEnqueue()
2 /home/ue/openairinterface5g/radio/COMMON/common_lib.c:195\Write queue full\
```

Further investigation revealed that the issue was specifically related to the connection with the 5GC. A wireshark capture in the 5GC (Figure 4.13) shows that the 5GC attempted to send a securityModeCommand message to negotiate NAS security three times. The responses (figure 4.14) are to be sent encrypted, as defined in figure 4.15. This encryption is not fully implemented in the OAI software, and, as such, the payload is being sent unencrypted. The 5GC, expecting to receive encrypted data, does not understand the contents of the message being received, leading to repeated attempts in the negotation and the sequent and registration reject message.

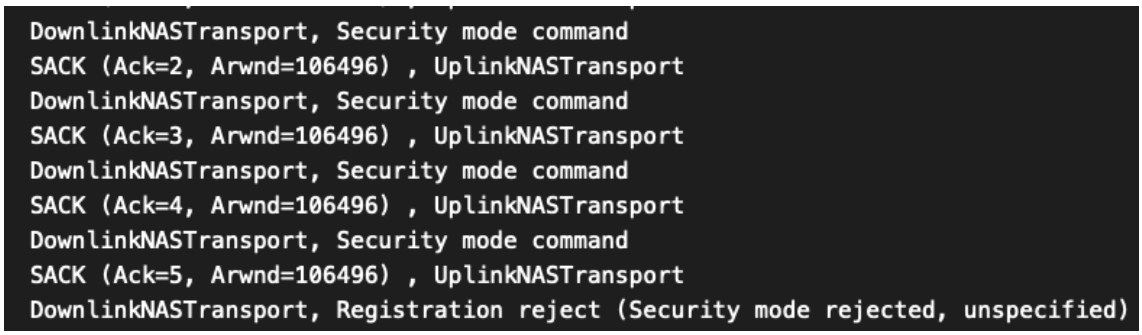

Figure 4.13: 5GC rejects connection to UE, ending the Authentication Phase

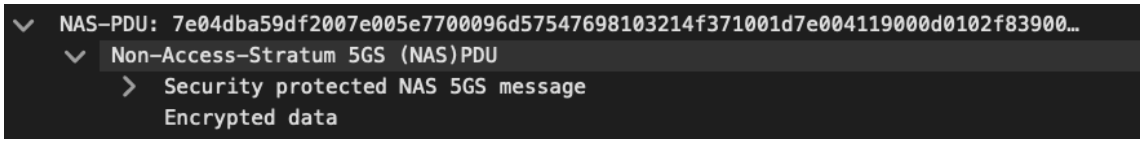

Figure 4.14: 5GC rejects connection to UE, ending the Authentication Phase

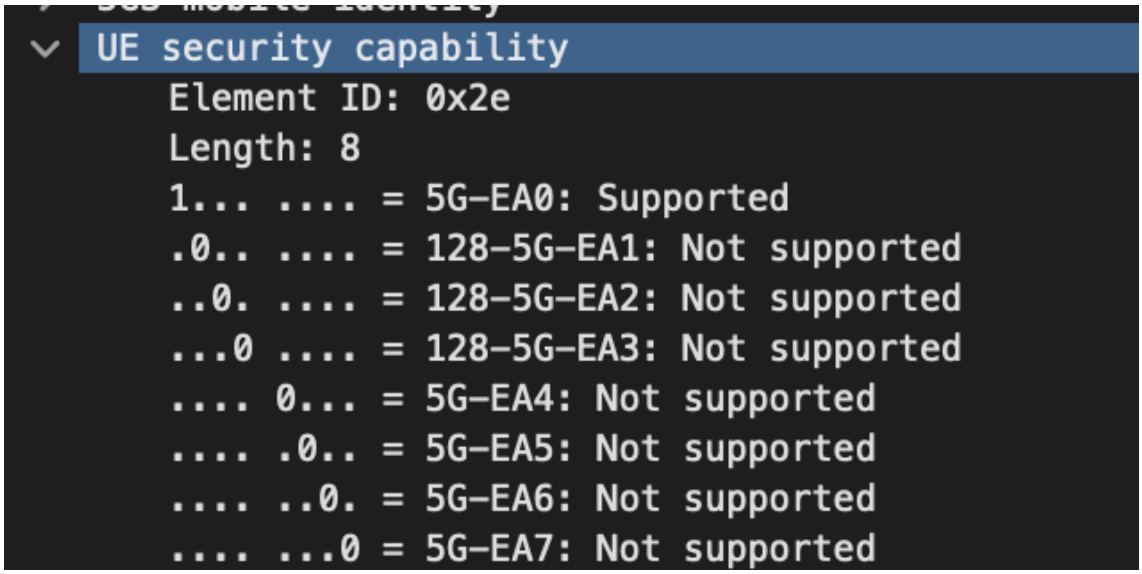

Figure 4.15: Encryption Announced by UE

The key to resolving the issue was changing the bit that indicates the UE's ability to encrypt to 0. This change is done in the file managing NAS in the UE, nr_nas_msg_sim.c, found in openair3/NAS/NR_UE. The new value for the encryption was 0x80, indicating the only option for encryption as NEA0, meaning no encryption, as shown in 4.16. It allowed the new version of the UE to function correctly, as the 5GC accepts NEA0 and expects unencrypted messages. After contacting the OAI team, they tested the issue and confirmed that they plan to fix it.

```
registration_request.presencemask |= REGISTRATION_REQUEST_
registration_request.nruesecuritycapability.iei = REGISTRA
registration_request.nruesecuritycapability.length = 8;
registration_request.nruesecuritycapability.fg_EA = 0x80;
registration_request.nruesecuritycapability.fg_IA = 0x20;
registration_request.nruesecuritycapability.EEA = 0;
registration_request.nruesecuritycapability.EIA = 0;
10;
```

Figure 4.16: Encryption Announced in NAS in OAI code

Chapter 5

Quantifying Time Overhead

5.1 Tools to Measure Time Overhead

In order to measure the timings for the various stages of the authentication procedure in the 5G network setup in ten repetitions of the experiment, the python library Pyshark was utilized to extract the timestamps of specific messages at different nodes. This tool allows for the analysis of packet captures in a programmable manner. To compare the timings of different stages, it is important to collect timestamps from different 5G entities, as the processes start with messages sent and received at different points. Therefore, the time is calculated as the difference from timestamps taken from different devices, in order to most accurately analyse the duration of each stage.

To verify the synchronization between the nodes, a script was implemented to process the ping results captured during the experiments. The script, presented in appendix B, utilizes the pyshark library to read the packet capture files and extract the timestamps of ICMP packets. A more in-depth explanation of how to utilize the library is provided in the section 5.2.3.1. The midpoints of the ping request and reply times are calculated for each pair of nodes (CU-DU, Core-CU, and UE-Core). The differences between these midpoints are then computed to assess the time offset between the nodes. This method does not consider asymmetry between propagation times, and can be influenced by its variation. However, it serves as a baseline to assess the result of the synchronization from the protocol.

5.2 Synchronization

5.2.1 Network Time Protocol

The Network Time Protocol (NTP) provides a mechanism for the time alignment of devices in a network by transmitting time-stamped messages between clients and servers. This interaction allows a device to adjust its internal clock to mirror the time kept by another device, the server, utilizing several iterations to correct for variables like network latency and jitter. Through this

process, time synchronization can be achieved, which is vital for analysing timings across different computers.

The latest specifications of the protocol [4] allow for different configurations in the role devices play within the network. These configurations can be symmetric, where a machine can be both client and server, unicast, either client or server, or broadcast, as either client or server. NTP clients are designed to query servers at intervals that can range from every 16 seconds to every 36 hours. The repeated process is necessary, as, due to clock drift, synchronized clocks do not run at the exact same rate, and can deviate from each other.

A study [12] concluded that NTP achieves comparable results with other leading protocols, such as Chrony, which utilizes GPS information for synchronization purposes, specifically when used in local servers with relatively close nodes. By testing under networks without traffic it has found the difference of error between both protocols to be in the scale of microseconds. At the one-hour mark, without any protocol running, it found the difference between clocks to be at around 4 milliseconds.

5.2.2 Implementation in Linux

To implement NTP, the Linux package "ntp" was utilized. The package allows for the set up of NTP servers and clients, configuring specific IP addresses for synchronization, regulating query intervals, and running queries at specified times.

Upon installing the package in a linux machine, through sudo apt-get install ntp, a file is created in the etc directory, named ntp.conf. This file can be used to decide the association modes in which NTP is used, and their restrictions.

A client can be set up with a line with the format server [server ip address] [burst option] minpoll [minpoll] maxpoll [maxpoll]. The following examples show possible configurations:

```
1 server 10.0.0.1 iburst
2 server 10.0.0.2 burst minpoll 4 maxpoll 14
```

The first example connects to a server with the IP address 10.0.0.1, utilizing the iburst mode. This mode sends a burst of packets instead of the usual one when the server is unreachable, leading to a faster start. As no poll interval is set, the default will be set at 64 seconds, increasing to a maximum of 1024 seconds when the clock discipline has stabilized. The second example utilizes the method burst, which sends bursts of packets instead of just one when the server is reachable, which is recommended as an option when the maxpoll is greater than 10, as the device does not synchronize as often. The max and min poll are calculated as powers of 2, such that in this option the interval between polls would start at $2^4 = 16$ seconds and increase to $2^{14} = 16384$ seconds.

The setup employed, presented in figure 5.1, utilizes the closest cabled connection available between nodes to synchronize between them. This criterion is applied to attempt to minimize variation and asymmetry between propagation times. The CU has the closest connections to the DU and 5GC and acts as a server for both. The UE can connect to the DU only when the connection

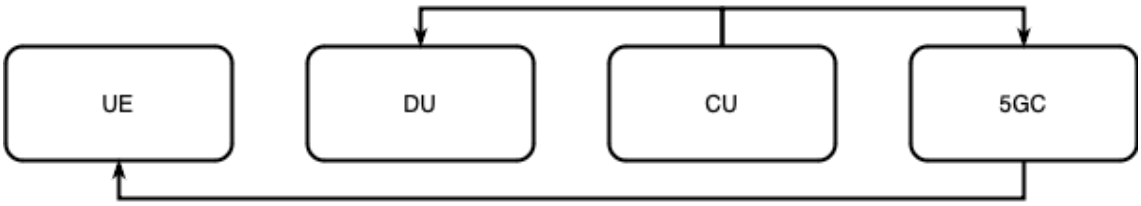

Figure 5.1: NTP Synchronization Architectural Setup

is established through the experimental setup. To ensure a stable connection to one of the other nodes, the shortest path is connecting to the 5GC through its public IP. For the configurations, the iburst mode is utilized, as sending only one packet during experimentation at higher relative intervals will not generate traffic that could impact the network significantly. The configurations of all four nodes are present in annex A.

5.2.3 Synchronization Verification

The mean and the 95% confidence interval for the time differences of 10 pings are calculated to measure synchronization accuracy. The 95% confidence interval is derived using the standard error of the mean and the t-distribution.

Node Pair	**Mean (μs)**	**95% Confidence Interval (μs)**
DU to CU	13.18	[10.96, 15.41]
Core to CU	-68.57	[-78.85, -58.29]
UE to Core	2904.44	[2707.44, 3101.44]

Table 5.1: Synchronization Verification Results: Mean and 95% Confidence Intervals in Microseconds

The following results were obtained from the verification: The differences between CU and DU exhibit a mean of approximately 1.318×10^{-5} seconds, with a 95% confidence interval ranging from 1.096×10^{-5} to 1.541×10^{-5} seconds, indicating a consistent offset. The differences between 5GC and CU present a mean of approximately -6.857×10^{-5} seconds, with a 95% confidence interval from -7.885×10^{-5} to -5.829×10^{-5} seconds, suggesting a slightly larger, but still relatively small offset when compared to the measurements of the security mechanisms, in the order of 10^{-2} seconds. The UE to 5GC differences show a more significant mean difference of approximately 2.904×10^{-3} seconds, with a 95% confidence interval between 2.707×10^{-3} and 3.101×10^{-3} seconds, reflecting a noticeable latency due to the longer connection path. These results demonstrate that the CU and DU are closely synchronized with minimal time variation, while the synchronization between the Core and CU is slightly less precise yet still within an acceptable range, still falling below the order expected when measuring the security. The NTP protocol is expected to be more exact than this verification method, as it employs statistical filtering and network delay compensation to account for variability and asymmetry in propagation delays. As such, the

synchronization method is shown to have a precision at least an order of magnitude lower than what is expected from the measurements.

Figure 5.2 shows the development of the calculated time differences over time. The time interval between each ping was 1 second. Visually, the graph does not imply any significant trend. A linear regression was performed, revealing a trend of $0.712365815\,\mu$s/s for the connection between CU and DU, a trend of $-2.24546953\,\mu$s/s between CU and 5GC, and a trend of $-46.2517594\,\mu$s/s between UE and 5GC. The trend is assumed to result from both clock drift and, more prominently, variations in channel propagation times. Due to the order of magnitude of the trends and the frequency of synchronization through NTP, it is concluded that clock drift will not affect the results.

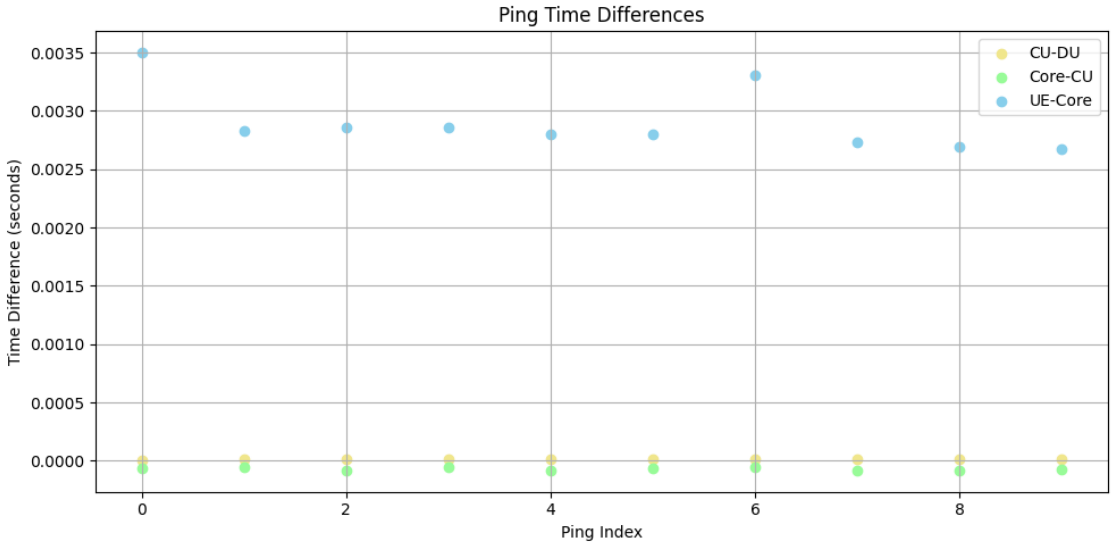

Figure 5.2: Synchronization Verification Results

5.2.3.1 Analysis Script

The Python script developed for this analysis, displayed in appendix C, utilizes PyShark to read and analyze packet capture files. PyShark processes one packet at a time, presenting each as a tree data structure. The packet's protocol can therefore quickly be checked utilizing the function hasattr(). For example, to find signalling messages between DU and CU, checking for the protocol 'f1ap' would filter these:

```
hasattr(packet, 'f1ap'):
```

As such, the parsing time of the .pcap files can be reduced, such that most packets can be excluded and only a shorter selection of packets related to signalling need to be analysed in depth. This is relevant as all packets need to be analysed for each of 10 iterations in three different entities. To ensure that all timestamps are collected in each iteration, a state machine was implemented, which controls the message, and in the capture of which entity, it is being searched for. In particular, signalling messages between the UE and CU can be checked by filtering through RRC,

or through the F1AP Protocol, which encapsules these messages between the CU and DU. The signalling between the UE and the 5GC can similarly be filtered by checking for the NGAP protocol. Two timestamps to be collected at the UE, the beginning of connection and sending of the RRC Setup Request message, are not captured through tcpdump. To circumvent this issue, the software was altered to print timestamps at both of these points, found at executables/nr-uesoftmodem.c and openair2/RRC/NR_UE/rrc_UE.c.

Utilizing the Python libraries 'numpy' and 'matplotlib', the statistics and visualisation of the data is processed. The timings of each step, calculated by the direct difference between both timestamps, are aggregated, are then visualized using a bar chart to show the proportion of time spent in each step. Scatter plots are utilized to display the distribution of timings for each step, in order to visualise the variability between iterations of the experiment.

5.3 Results

The first point of analysis is the relative value between the times consumed in each task, presented in figure 5.3.

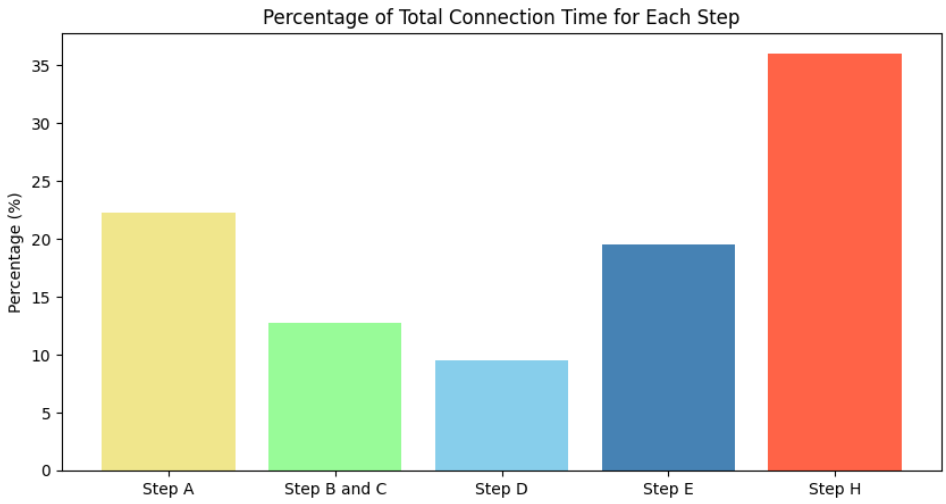

Figure 5.3: Authentication Phase Security Mechanisms Time Distribution Bar

Figure 5.3 indicates the lowest time for step D, responsible for the NAS signalling encryption decision, at 9.5%, around 15.17ms. It is closely followed by step B and C, responsible for identification and authentication to the 5GC, occupying a relative time of 12.8% of the connection, at an average of aproximately 20.35ms. Step A, responsible for the the RRC signalling connection, takes 22.27% of the connection time, at an average of 35.50ms. Step E, responsible for the data plane encryption in open air, takes up a similar value of 19.5%, an average of 31.06ms. The longest step, the TLS handshake, occupies 36.0% of the total time, at an average of 57.32ms.

The scatter plot displayed in figure 5.4 shows low variability in the time, represented in seconds, between different experiments for all steps, to the exception of the TLS handshake. As opposed to the TLS handshakes, all other steps are in a controlled environement without traffic generated from other nodes. In accessing the server through the internet, it is expected for the results to have a higher variation.

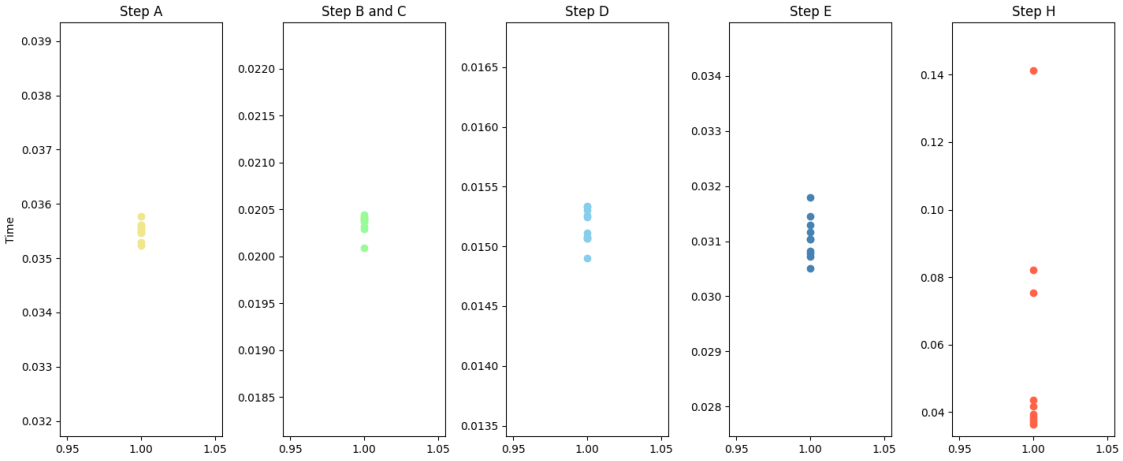

Figure 5.4: Authentication Steps Time Distribution in Scatter Plot

As the processes are not completely sequent, and at some points overlap, the following graph 5.5 shows the start and end of each step, to further contextualize its timings.

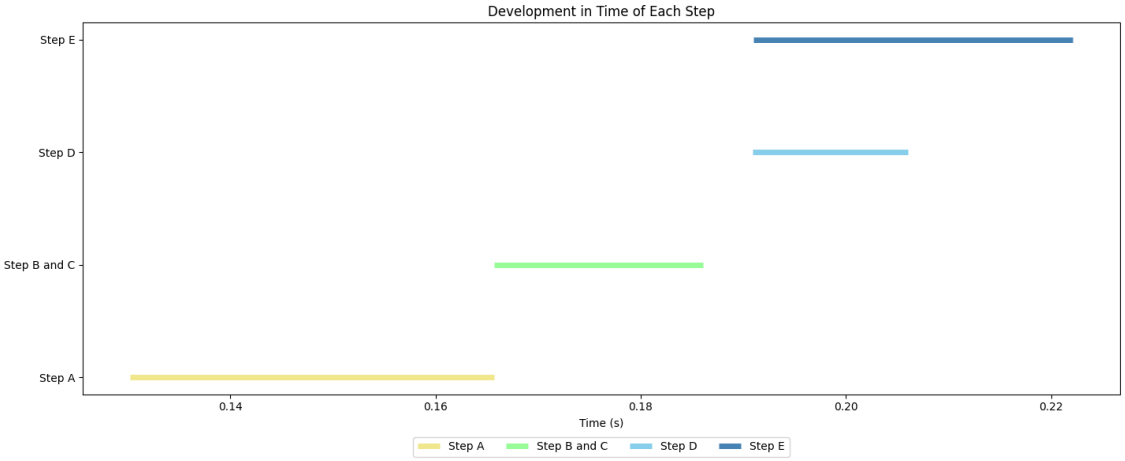

Figure 5.5: Authentication Steps Time Development

5.4 Delay Model

5.4.1 Model Design

So far in this analysis, the propagation delays between UE, gNB, 5GC, and the server have been constant. The performance of these mechanisms can, however, be influenced by the propagation

delays between these elements. This section aims to analyze how changes in these propagation delays affect the percentual time contribution of each security mechanism, and therefore, the impact of potentially removing one of them.

The first variation analyzed is in the propagation delay between UE and gNB. In a private setup with a cabled connection, the propagation delay becomes minimized. To simulate a real environment with a longer physical distance, delays from signal processing, and delays from the competition for time slots, the propagation delay is varied up to more 20ms. Changes in this propagation delay don't impact all steps equally:

- Step A requires 3 messages that go through the UE-gNB connection, RRCSetupRequest, RRCSetup and RRCSetupComplete. As such, it changes linearly with a factor of 3 by the change in propagation delay.
- Step B and C require only the authentication request and authentication response messages from this connection. The identifier is sent from gNB to 5GC directly, so it doesn't involve the UE-gNB interface. Their scaling factor is, therefore, 2.
- Step D and E both require a SecurityModeCommand and SecurityModeComplete message, both having to go through the UE-gNB interface. Their scaling factor is, therefore, 2.
- Step H requires 4 messages, ClientHello, ServerHello and ChangeCipherSpec, and the ChangeCipherSpec answer from the UE. However, as the serverhello and ChangeCipherSpec are sent sequentially by the server, the impact on the total time of the propagation change will only be counted once, leading to a scaling factor of 3.

The next change analysed was the propagation delay between gNB and 5GC, simulating a situation where the backhaul would be a bottleneck of the network. Originally, the propagation time between both during the connection was on average 0.3ms.

- Step A and E both have scaling factors of 0, as the messages utilize only the UE-gNB connection.
- Step B and C are now increased to a scaling factor of 3, as the identification message from gNB to 5GC is now affected, as well as both authentication messages mentioned before.
- Step D and H remain the same.

The last change in propagation delay analyzed is between 5GC and the server. This change allows for an analysis of how the proximity of the server impacts the weight of each security mechanism. The interval of time was adjusted to be tested between -8.5ms and 16.5ms, as the time difference between 5GC and UE is already at 9ms, measured through an average of 10 pings to the server.

- Step A, B, C, and D are unaffected by changes between the 5GC and the server.
- Step H remains influenced with the same scaling factor of 3.

5.4.2 Model Implementation

The script, presented in appendix D, simulates how changes in propagation delays between different network elements in a 5G authentication process affect the percentual time contribution of various security steps. The primary goal is to visualize how the propagation delays impact each step, allowing for analysis of their significance under different conditions. The model considers three scenarios: delays between UE and gNB, gNB and 5GC, and 5GC and the server, each with different scaling factors that reflect the impact of propagation delays on specific steps.

The mathematical model used in the script calculates the scaled times for each step based on the initial average times and scaling factors. For a given time interval t, the scaled time for each step i is computed as

$$\text{scaled\_time}_i(t) = \text{initial\_time}_i + \text{scaling\_factor}_i \cdot t$$

The total time for all steps is then

$$\text{total\_time}(t) = \sum_i \text{scaled\_time}_i(t)$$

The percentage contribution of each step i at time t is given by

$$\text{percentage}_i(t) = \left(\frac{\text{scaled\_time}_i(t)}{\text{total\_time}(t)} \right) \times 100$$

The script generates plots showing these percentages over time, illustrating the relative impact of each step as propagation delays vary.

5.4.3 Results of Propagation Delay Variation between UE and gNB

Figure 5.6 depicts the percentage changes for the scenario between the UE and gNB. As the added propagation time increases, the percentage contribution of Step H decreases, while Steps A, B and C, and D, become more significant. Despite having the highest scaling factor, Step H sees its relative importance diminish in longer propagation times. Step E remains relatively constant in time. It is expected that the percentages converge to the relative value of the scaling factors as the impact of the propagation time grows in relation to the fixed time taken during computation.

For comparison, figure 5.7 shows the absolute value changes over time. It is clear from both figures that as the propagation delay becomes higher, the relative impact of the TLS handshake decreases, while the impact on the connection time of the authentication with 5GC and NAS Encryption and Integrity Negotiation increases slightly.

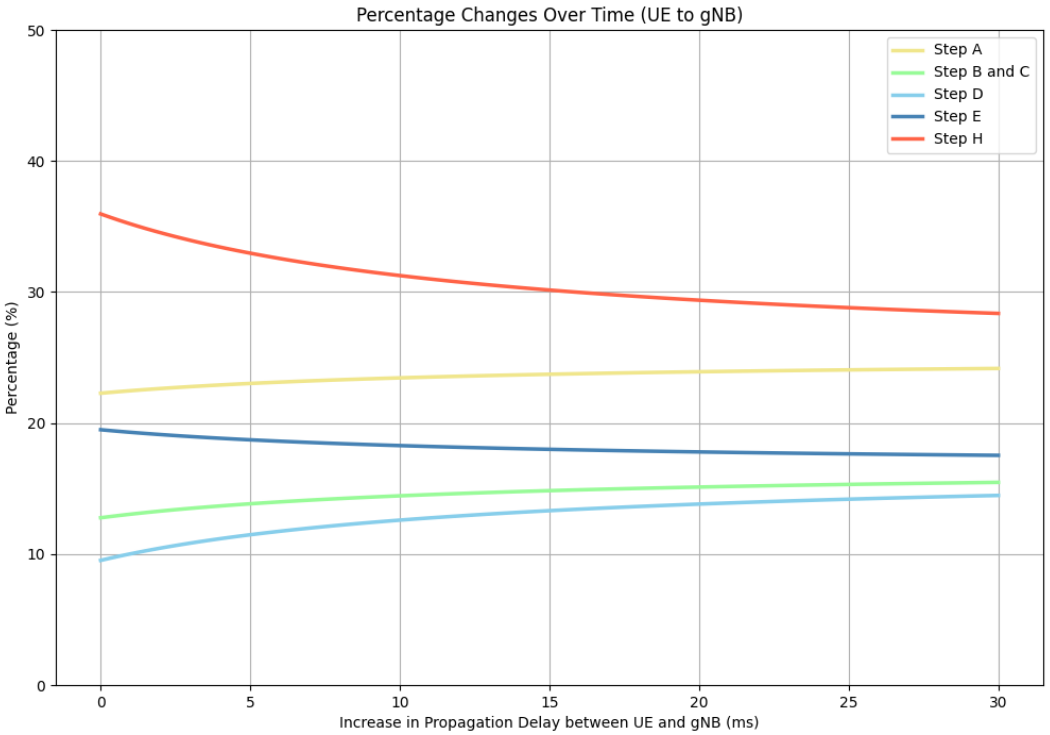

Figure 5.6: Step Percentages at different Propagation Time Changes (UE to gNB)

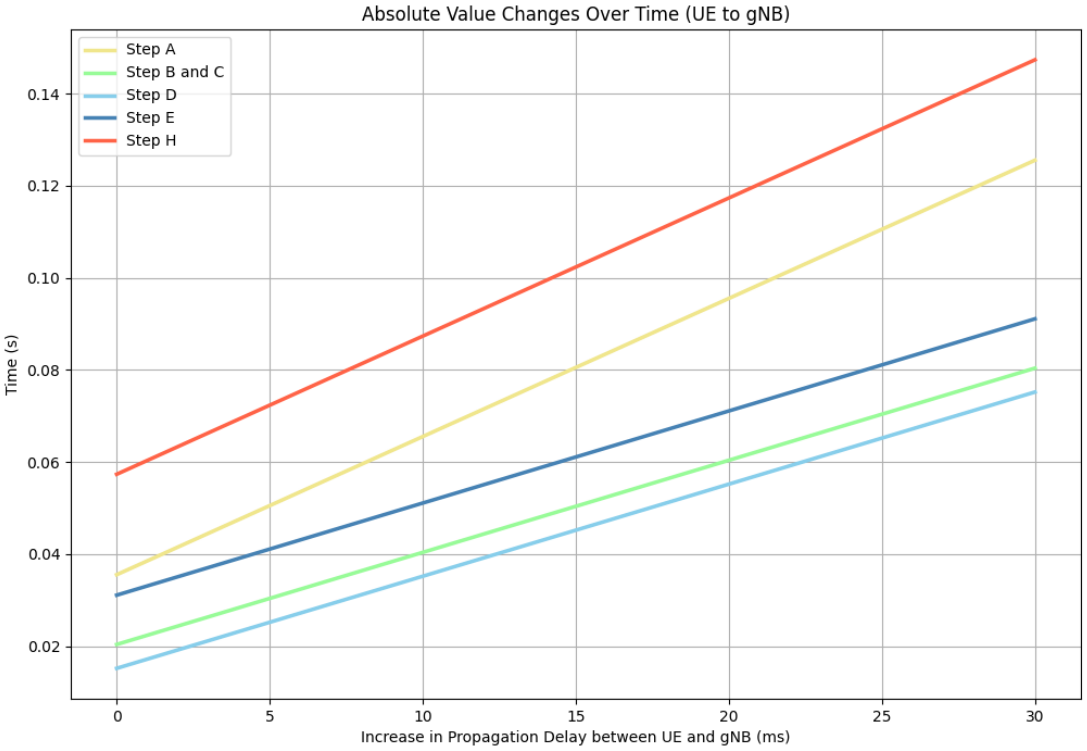

Figure 5.7: Absolute Value Changes at different Propagation Times (UE to gNB)

5.4.4 Results of Propagation Delay Variation between gNB and 5GC

Figure 5.8 shows the percentage changes for the scenario between the gNB and 5GC. while Step H remains the highest influence in the overall authentication time, the rest of the hierarchy changes. The most pronounced change is in Steps B and C, which become more dominant due to their high scaling factor, while the contributions of Steps A and E slightly decrease. Step D shows a slight increase, however not as pronounced, due to the lower scaling factor.

Figure 5.9 shows the absolute value changes over time for the scenario between the gNB and 5GC. Both figures show that, in a situation where the bottleneck of a connection is between the gNB and 5GC, the relative influence of the security mechanisms of the RRC Setup and Open Air Encryption and Integrity diminishes. The impact of the TLS Handshake remains approximately constant, while the Identification and Authentication with 5GC phase sees the highest relative increase.

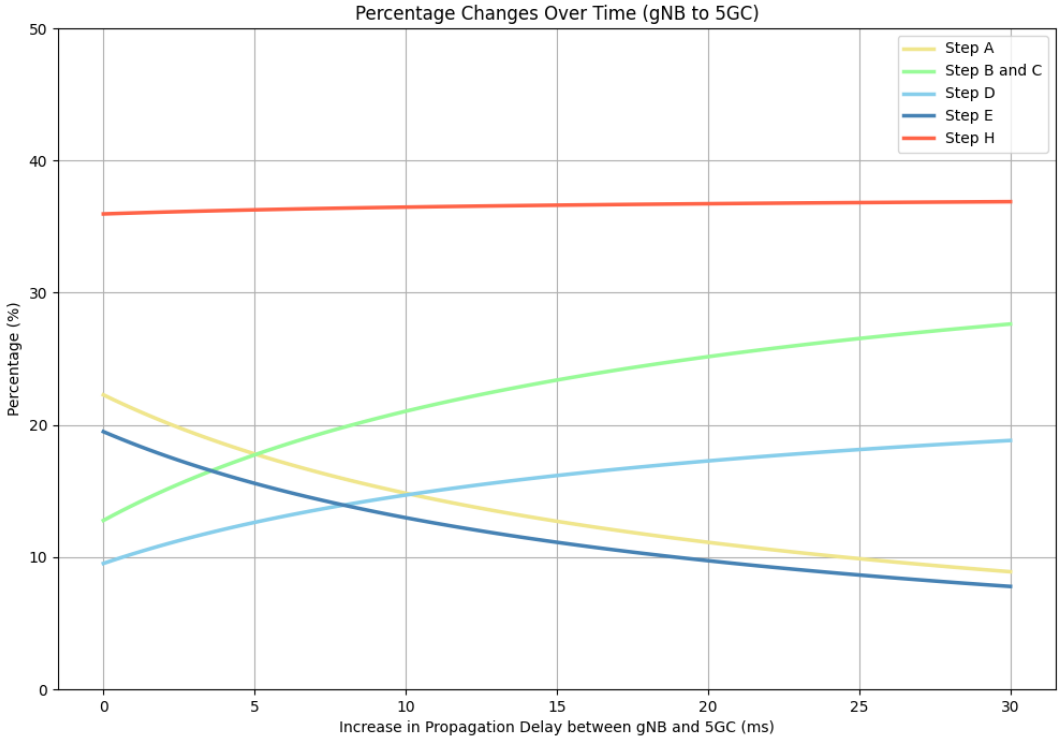

Figure 5.8: Step Percentages at different Propagation Time Changes (gNB to 5GC)

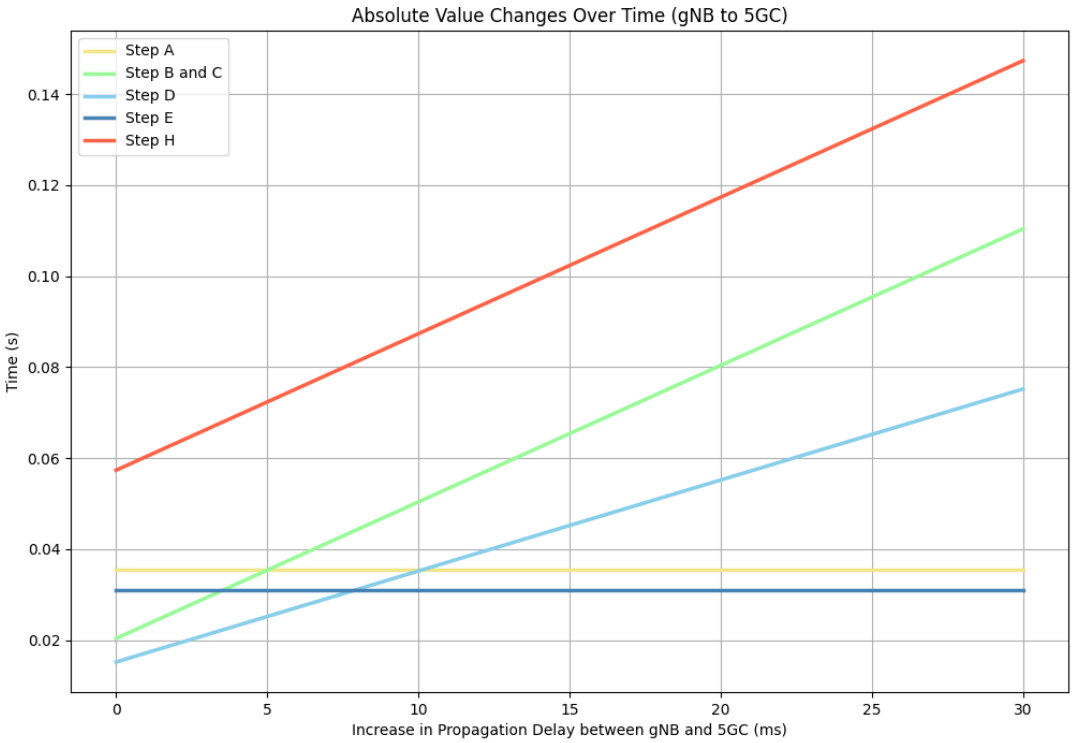

Figure 5.9: Absolute Value Changes at different Propagation Times (gNB to 5GC)

5.4.5 Results of Propagation Delay Variation between 5GC and server

Figure 5.10 depicts the percentage changes for the scenario between the 5GC and end server. As propagation time increases, the percentage contribution of Step H becomes more and more dominant, ending over 50%.

Figure 5.11 shows the absolute value changes over time for the scenario between the 5GC and server. It is clear that the dominance of the TLS handshake in the impact on the authentication time is vastly dependent on the propagation time. In closer proximity, such as in edge computing, the RRC Setup seems to have a higher impact than the TLS handshake, which stays similar to the open-air encryption step at around 23%.

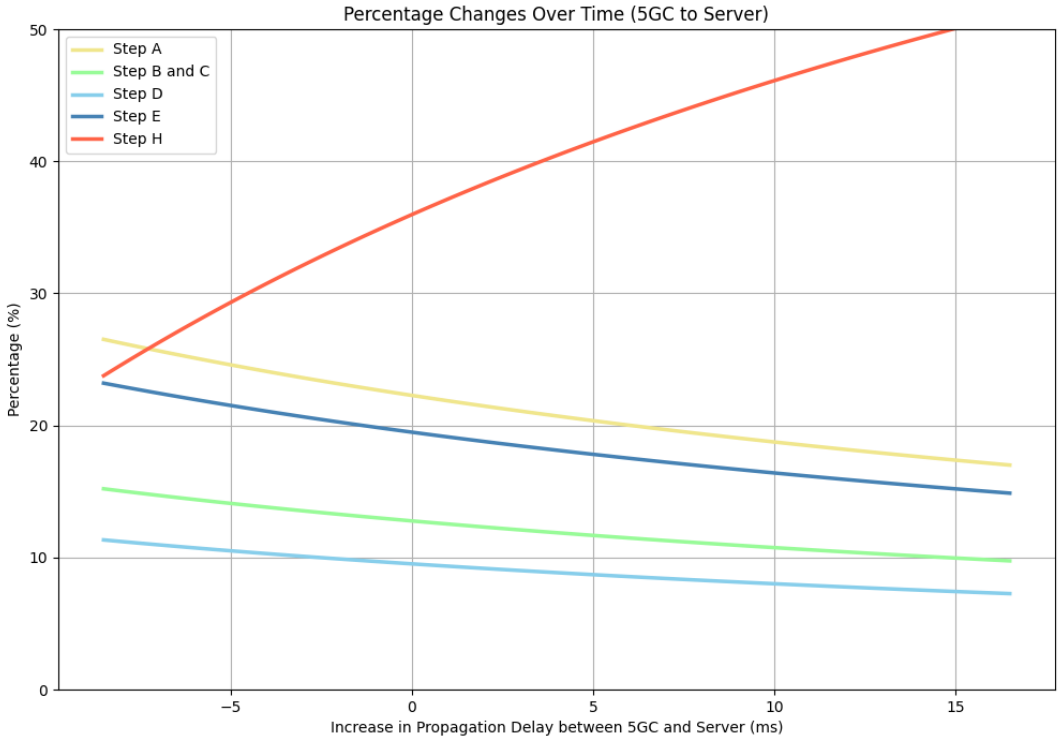

Figure 5.10: Step Percentages at different Propagation Time Changes (5GC to Server)

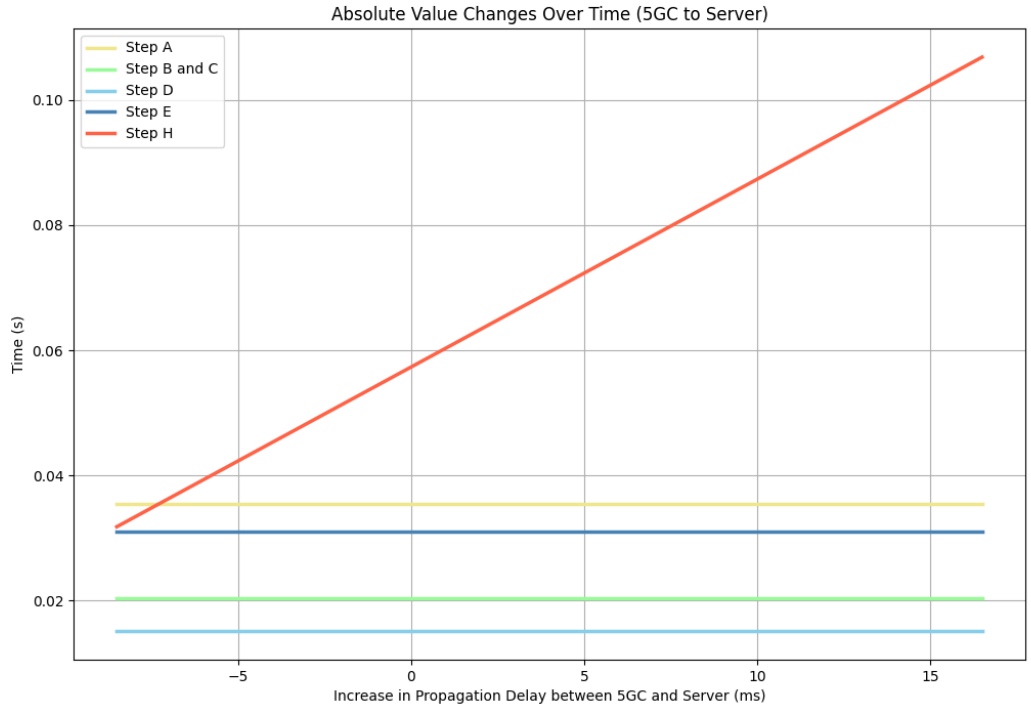

Figure 5.11: Absolute Value Changes at different Propagation Times (5GC to Server)

Chapter 6

Discussion

6.1 Categorization of Use Cases

During the theoretical and practical analysis, several options and choices between the trust mechanisms became apparent. To complement this flexibility inherent in 5G, a set of catefories based on specific metrics was developed during this research, compiling key factors that influence the choice of security mechanisms most favorable to the needs and constraints of each use case. In this section, concrete categories are utilized to structure these differences. Following the categories, a table structures all the use cases discussed in Section 2.1.2 according to these new differentiations. This approach ensures a base for a discussion on how concrete security mechanisms under analysis can be optimized for various 5G applications.

The proposed factors under analysis are:

1. Connection Duration
2. Device Mobility
3. Device Purpose
4. Data Sensitivity
5. Latency Requirements
6. Power Source

5G use cases differ in the frequency and duration of their connections. In spontaneous connections, such as sensors sending their measurements at a given frequency or an autonomous vehicle sending an alert to a neighbouring entity, the authentication procedure tends to play a more prominent role in the time and energy overhead, leading to an opportunity of minimization through a more use-case-conscious choice of trust mechanisms. In longer connections, the choices related to authentication are not as impactful, as the relative time and energy overhead caused by the authentication becomes smaller in relation to the overhead caused by confidentiality and integrity

mechanisms. Although both types of use cases can benefit from avoiding some of the encryption and integrity negotiations of the authentication, confidentiality and integrity mechanisms, this distinction helps identify the most critical component.

5G use cases can be distinguised between stationary and moving scenarios. Use cases with moving devices, such as connected vehicles or smartphones, must manage handovers more often than stationary devices. The choice in authentication of 5G AKA over EAP-AKA' can have a positive impact in minimizing time and energy overhead during handovers between different 5GCs, which is mostly relevant for moving devices. Stationary use cases, such as fixed humidity sensors or smart home devices, do not have the same need to be optimized for the risk of tracking through identifiers or hidden identifiers [9] when compared to moving devices. This factor can also restrict the choice of security mechanisms.

5G use cases can vary significantly between multi-purpose and single-purpose devices. Single-purpose devices, such as environmental sensors, typically connect only to one server. However, multi-purpose devices like smartphones connect to multiple services, leaving a trail of connections to different endpoints with varying frequencies. This connection pattern can be used to reliably identify a user and be used for tracking [9]. To mitigate this risk, multi-purpose devices might need to encrypt parts of their connections such as the destination address, ensuring that the usage patterns of services they connect to remain protected from potential tracking.

Use cases can also be separated based on wether the information they send is private or public. Devices handling private information need to employ encryption to protect sensitive data. For example, a smart home sensor might encrypt data to prevent revealing when a family is home. However, some use cases, devices like sensors measuring humidity used in agriculture, might not require from the same level of security. These devices could instead only use integrity mechanisms, with the possibility of lowering their temporal and energetic overhead.

Use cases in 5G also differ significantly betweeen those requiring low latency and those that are more delay tolerant. For low latency use cases, such as remote surgery or autonomous driving, it is imporative to minimize the overhead caused by the security mechanisms. In delay tolerant use cases, such as environmental monitoring for analytics, time is not the main concern. As such, these use cases can afford to prioritize a focus on the minimization of the energetic overhead instead.

5G use cases can differ significantly between battery-powered devices and those connected to a constant power source. Battery-powered devices, such as humidity sensors in agriculture, must focus on minimizing the energy consumption of their security mechanisms to ensure a long operational life without frequent recharging. These devices need to optimize for energy efficiency to remain self-sufficient for extended periods. In contrast, devices connected to a constant power source, such as devices related to robotics in an industrial context, have more flexibility to prioritize other optimizations, such as reducing temporal overhead or implementing different security measures without worrying about energy constraints.

		Delay Tolerant	Low-Latency Requirements	Public Data	Private Data	Multi-Purpose	Single-Purpose	Moving Devices	Stationary Devices	Continuous Connections	Spontaneous Connections
Manufactoring	Robotics		✓	(✓)	(✓)		✓		✓	✓	
	Inventory Tracking	✓		(✓)	(✓)		✓		✓		✓
	Predictive Maintenance	✓		(✓)	(✓)		✓		✓		✓
Healthcare	Online Consultations		✓		✓		✓		✓	✓	
	Remote Surgery		✓		✓		✓		✓	✓	
	Monitoring	(✓)	(✓)		✓		✓		✓	(✓)	(✓)
Utilities	Automated Metering	✓		(✓)	(✓)		✓		✓		✓
	Environmental Monitoring	✓		(✓)	(✓)		✓		✓		✓
Automated Driving	Real-Time Data Exchange		✓	(✓)	(✓)	(✓)	(✓)	✓		(✓)	(✓)
	Remote Control and Monitoring		✓	(✓)	(✓)		✓	✓		(✓)	(✓)
Video Games	Cloud-Gaming		✓	(✓)	(✓)		✓	(✓)	(✓)	(✓)	(✓)
	Virtual Reality		✓	(✓)	(✓)		✓	(✓)	(✓)	✓	
	Smartphones	(✓)	(✓)	(✓)	(✓)	✓		✓		(✓)	(✓)

Figure 6.1: Categorized Use Cases

6.2 Effect of Current Standards of Security Mechanisms in Trade-Off Opportunities

3GPP provides various options to enhance the flexibility of 5G networks, yet this flexibility comes at the cost of increased authentication and negotiation time for different encryption methods. The segmentation of 3GPP security into distinct documents without comprehensive integration makes an informed choice of security mechanisms more complicated for operators and end-consumers. While a technical specifications document dedicated to security in 5G exists [7], it does not thoroughly address the negotiations at different encryption points, such as RRC and NAS. As such, the absence of unified documentation explaining these various options and their purposes can undermine the effectiveness of 3GPP in ensuring security. 3GPP provides flexibility without assuming responsibility for the implemented security, leaving it to operators and end-users. This approach,

while understandable to a degree due to the need for flexibility, complicates the role of those responsible for security. One exception is in industries that plan to rely on private deployments of 5G, where total synchronization between the options chosen by the operator and end-consumer can be achieved.

6.2.1 Data Plane

In the data plane, encryption from the UE to the end server is standard, but 3GPP allows operators to choose whether to implement encryption from the gNB to 5GC, and allows end consumers and operators to negotiate whether to implement encryption from the UE to gNB. This can raise an issue of trust between the involved parties, as 3GPP does not indicate how the gNB to 5GC tunnel should be secured. In consequence, the UE could possibly need to account for the possibility that encryption from gNB to the 5GC might not be implemented, as assuming such encryption could lead to increased risks. A primary argument for open-air encryption use cases is to conceal the server being contacted. Research indicates that a user's connection pattern and frequency to various servers can be uniquely identified [9]. In cases like smartphone usage, this encryption is valuable. However, this encryption may not be necessary for single-purpose use cases connecting to only one server.

6.2.2 Control Plane

The control plane also faces the issue of double encryption, where the UE cannot be certain if there is encryption between the gNB and the 5GC. This uncertainty may lead to redundant encryption efforts by encrypting both from UE to gNB and UE to 5GC, highlighting the need for better coordination between the UE and the operator, which is currently lacking due to 3GPP's role in opening multiple options. An interesting case to examplify this dillema are devices utilizing non-encrypted identifiers, such as legacy equipment or the UE implementation by OAI. In these cases, the UE could risk being tracked or have its identifier stolen if the connection between gNB and 5GC is not ciphered, therefore possibly choosing to encrypt both to gNB and 5GC.

6.3 Cost of Flexibility in the Authentication Process

Flexibility itself implies costs. The existence of numerous options means each must be considered during authentication, impacting time and energy overheads. Reducing this time is vital when minimizing the time for the first message or energy costs is critical. A possible change to the authentication phase could be using parameters like the RRC establishmentCause to signal an intent to bypass certain encryption/integrity exchanges to optimize performance. Similar to how SOS calls are managed in 5G, this parameter could then be used to bypass certain steps to implement optional encryption or integrity methods, aligning with the specific needs of critical use cases.

The first step that could be considered to be skipped is the NAS Signaling Encryption and Integrity Negotiation, which had an impact of, on average, 15ms on the temporal overhead, at

a percentage of 9.5% of the total authentication procedure. The results from the delay model showed that this percentage tends to increase for longer propagation delays between UE and gNB, converging to closer to 15%. It is expected that in less controlled environments, this propagation delay should increase, due to increased traffic and physical distance between nodes. Propagation delay variations between the gNB and 5GC, simulating a bottleneck in this connection, lead to an increase, converging to closer to 20%. Propagation delay variations between the 5GC and the server do not influence this percentage as much. With a server connected directly to the 5GC, the impact of this step would only increase by 3%. Increasing the propagation time slowly leads the percentage to converge to around 8%. This result suggests that skipping this step leads to the most impact in cases of a bottleneck at the connections between gNB and 5GC, however, its removal would be always beneficial, as the impact does not seem to decrease past 8% in any of the scenarios. Skipping NAS signalling encryption is only an option in a limited amount of use cases, as it protects the UE from attackers stealing its identifier. In this case, the integrity would have to be maintained through RRC signaling between UE and gNB, as well as through tunneling between gNB and 5GC. To hide the identifier of the UE and to avoid tracking, encryption in both of those connections would be necessary. If the UE is not impacted by being tracked, as would be the case with some IoT devices, the encrypted identifier option available in 5G could be employed instead of the encryption of the signaling tunnels.

Removing the TLS Handshake could have the most relevant impact in diminishing temporal overheads. Experimentally, the TLS handshake occupied 36% of the total time in the authentication process. This value diminished in the propagation delay variation between UE and gNB, to slightly below 30%, while it remained constant in the variations between gNB and 5GC. In both cases, the impact was more prominent than from the other security mechanisms. Increasing the propagation time between the 5GC and server significantly increases the relative impact, to more than 50%, over an increase of 15ms. Unlike other propagation variation delays, the percentage will converge to 100% in this case, as no other security mechanism grows with the variation. It's important to note that, in cases of close server proximity, the impact of the TLS handshake is diminished, up to around 23%, the same as the impact of the data plane encryption and integrity algorithm negotiation. The removal of the data plane encryption and integrity algorithm negotiation step would also have a significant impact on the time of the authentication process. The experiment showed a 19.5% contribution to this time. In the case of an increase in the propagation delay between the UE and gNB, the impact did not change too significantly. However, in the case of a higher delay between gNB and 5GC, the impact of this step does seem to lower significantly, in 30ms lowering to around 8%. Use cases that require an exchange of private information cannot consider removing the TLS handshake. Multi-purpose devices such as smartphones that connect to several IP addresses and need to avoid tracking for safety concerns should not consider removing the open-air encryption between UE and gNB. Public 5G networks will always need to guarantee the integrity of the UE for legal and billing reasons, and as such, removal of this negotiation process would have to be paired with, for example, the usage of the integrity method from the RRC connection for the data plane.

6.4 The Role of Open-Source Solutions

In OAI, the focus on implementing 3GPP's flexibility mechanisms is apparent, but certain decisions, such as the lack of encryption in tunnels from the gNB to the 5GC, are already made. The use of the software shows that most security mechanisms are turned off by default, which can be a limiting factor when considering the distribution of security responsibility among end consumers and operators, which can both utilize this software. The UE software shows a lack of encryption mechanisms available, which, when paired with the incorrect advertisement of encryption in the UE capabilities in NAS, could lead to an incorrect understanding of the security it provides. This situation poses a risk of affecting performance studies using open-source software, due to the measuring of unrealistic scenarios, as seen with the unencrypted identifier in non-encrypted communication, which could impact results.

Chapter 7

Conclusion

This thesis explored possibilities of balancing flexibility, efficiency, and security in the context of 5G networks. Through analysis and characterization of the trust mechanisms employed within 5G communications, critical trade-offs and potential improvements to latency were identified and linked to diverse use cases. The investigation into 3GPP's security standards highlighted that while multiple encryption and authentication options provide valuable flexibility, they also introduce complexities and potential inefficiencies. The task of selecting and implementing appropriate security mechanisms for operators and end-consumers became clearer by unifying the segregated security documentation. The split of responsibility and choice of security mechanisms between the operators and end-consumers in the choice to encrypt connections between the UE and gNB and between the gNB and 5GC was identified as a potential cause of inefficiency in the data and control plane. While open-air encryption is essential to conceal server connections for privacy, it was found that for use cases like IoT sensors that connect to a single server, the encryption, and its negotiation procedures could be avoided to minimize temporal and energy overheads. The control plane faces the challenge of potential double encryption, where the UE cannot ascertain if there is encryption between the gNB and 5GC. This uncertainty can lead to redundant encryption efforts, highlighting the need for better coordination between network entities.

Flexibility in the authentication process incurs costs, as the multitude of options available results in increased time and energy overheads. Leveraging parameters such as the RRC establishmentCause to skip some security mechanisms for critical use cases could mitigate the temporal and energy overhead. Practical insights were gained from deploying a non-public 5G network using open-source software solutions. This deployment quantified the temporal overheads of various security measures and employed a delay mathematical model to study the impact of propagation delay variations on experimental results. The findings demonstrated how varying propagation delays and different security mechanisms impact overall connection times, emphasizing the need for use-case- and network-conscious trade-offs in security.

This research underscores the importance of a use-case-aware approach to implementing 5G security mechanisms, balancing the diverse requirements of different applications. By considering the specific needs of different use cases and viewing the 5G ecosystem as a cohesive whole,

it is possible to achieve robust security while enhancing performance. The proposed categorization model guides future implementations, ensuring that 5G networks can meet the demanding requirements of contemporary and emerging applications, contributing to the broader objective of achieving low-latency, efficient, and secure 5G connectivity.

References

[1] 5G Americas. 5G Vertical Use-Cases - A 5G Americas White Paper, 2021.

[2] David Basin, Jannik Dreier, Lucca Hirschi, Saša Radomirovic, Ralf Sasse, and Vincent Stettler. A formal analysis of 5g authentication. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 1383–1396, New York, NY, USA, 2018. Association for Computing Machinery.

[3] T Dierks. The TLS protocol. 1999.

[4] Tim Dierks and Eric Rescorla. The transport layer security (tls) protocol version 1.2. *RFC*, 5246:1–104, August 2008.

[5] ETSI. 5G; NR; Radio Resource Control (RRC); Protocol specification (3GPP TS 38.331 version 16.1.0 Release 16), 2020.

[6] ETSI. Multi-access Edge Computing (MEC) MEC 5G Integration, 2020.

[7] ETSI. Security architecture and procedures for 5G System (3GPP TS 33.501 version 16.3.0 Release 16), 2020.

[8] Thomas Kothmayr, Corinna Schmitt, Wen Hu, Michael Brünig, and Georg Carle. A DTLS based end-to-end security architecture for the Internet of Things with two-way authentication. In *37th Annual IEEE Conference on Local Computer Networks - Workshops*, pages 956–963, October 2012.

[9] Jeffrey Pang, Ben Greenstein, Ramakrishna Gummadi, Srinivasan Seshan, and David Wetherall. 802.11 user fingerprinting. In *Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking*, MobiCom '07, page 99–110, New York, NY, USA, 2007. Association for Computing Machinery.

[10] Larry Peterson, Oguz Sunay, and Bruce Davie. *Private 5G: A Systems Approach*. Systems Approach LLC, version 1.1-dev edition, 2022.

[11] Suzan Sallam and Babak D. Beheshti. A Survey on Lightweight Cryptographic Algorithms. In *TENCON 2018 - 2018 IEEE Region 10 Conference*, pages 1784–1789, October 2018. ISSN: 2159-3450.

[12] Eduardo Soares, Pedro Brandão, and Rui Prior. Analysis of timekeeping in experimentation. In *2020 12th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, pages 1–6, 2020.

[13] Sean Turner. Transport Layer Security. *IEEE Internet Computing*, 18(6):60–63, November 2014. Conference Name: IEEE Internet Computing.

[14] UNESCO, International Centre for Engineering Education. Engineering for sustainable development: executive summary, 2021.

Appendix A

Network Time Protocol Configurations

5GC Configurations:

```
# Synchronize to specific NTP servers
server 10.0.0.3 iburst
server [Public IP of UE] iburst

# Allow localhost
restrict 127.0.0.1
restrict ::1

# Enable logging
logfile /var/log/ntp.log
```

UE Configurations:

```
# Synchronize to specific NTP server
server [Public IP of 5GC] iburst

# Allow localhost
restrict 127.0.0.1
restrict ::1

# Enable logging
logfile /var/log/ntp.log
```

CU Configurations:

```
# Allow only specific clients
restrict 10.42.0.2 mask 255.255.255.255 nomodify notrap
restrict 10.0.0.2 mask 255.255.255.255 nomodify notrap

# Block all other clients
```

```
restrict default ignore

# Allow localhost
restrict 127.0.0.1
restrict ::1

# Enable logging
logfile /var/log/ntp.log

# Serve time to clients without syncing to an external server
server 127.127.1.0  # local clock
fudge 127.127.1.0 stratum 10
```

DU Configurations:

```
# Synchronize to specific NTP server
server 10.42.0.1 iburst

# Allow localhost
restrict 127.0.0.1
restrict ::1

# Enable logging
logfile /var/log/ntp.log
```

Appendix B

Script for Synchronization Verification

```
1
2  import pyshark
3  import matplotlib.pyplot as plt
4  import numpy as np
5  from scipy import stats
6
7  pingCUDUdifferences = np.zeros(10)
8  pingCoreCUdifferences = np.zeros(10)
9  pingUECoredifferences = np.zeros(10)
10
11 path = "/Users/pedro/Pictures/Tese/timingsLast/pedro-claro-experiments/"
12 du_sending = pyshark.FileCapture(path + "sending-du.pcap")
13 cu_receiving_du = pyshark.FileCapture(path + "cu-receives-du.pcap")
14 core_sending = pyshark.FileCapture(path + "core-sends.pcap")
15 cu_receiving_core = pyshark.FileCapture(path + "cu-receives-core.pcap")
16 ue_sending = pyshark.FileCapture(path + "ue-sending.pcap")
17 core_receiving_ue = pyshark.FileCapture(path + "core-receiving-ue.pcap")
18
19 def get_ping_timestamps(capture):
20     timestamps = []
21     for packet in capture:
22         if 'ICMP' in packet:
23             timestamps.append(float(packet.sniff_time.timestamp()))
24     return timestamps
25
26 timestamps_du_sending = get_ping_timestamps(du_sending)
27 timestamps_cu_receiving_du = get_ping_timestamps(cu_receiving_du)
28 timestamps_core_sending = get_ping_timestamps(core_sending)
29 timestamps_cu_receiving_core = get_ping_timestamps(cu_receiving_core)
30 timestamps_ue_sending = get_ping_timestamps(ue_sending)
31 timestamps_core_receiving_ue = get_ping_timestamps(core_receiving_ue)
32
33 if len(timestamps_du_sending) < 10 or len(timestamps_cu_receiving_du) < 10:
```

```
34      print("Error: Not enough ICMP packets in DU sending or CU receiving DU captures
            ")
35  if len(timestamps_core_sending) < 10 or len(timestamps_cu_receiving_core) < 10:
36      print("Error: Not enough ICMP packets in Core sending or CU receiving Core
            captures")
37  if len(timestamps_ue_sending) < 10 or len(timestamps_core_receiving_ue) < 10:
38      print("Error: Not enough ICMP packets in UE sending or Core receiving UE
            captures")
39
40  for i in range(10):
41      midpoint_du_sending = (timestamps_du_sending[2*i] + timestamps_du_sending[2*i +
             1]) / 2
42      midpoint_cu_receiving_du = (timestamps_cu_receiving_du[2*i] +
            timestamps_cu_receiving_du[2*i + 1]) / 2
43      pingCUDUdifferences[i] = midpoint_cu_receiving_du - midpoint_du_sending
44
45  for i in range(10):
46      midpoint_core_sending = (timestamps_core_sending[2*i] + timestamps_core_sending
            [2*i + 1]) / 2
47      midpoint_cu_receiving_core = (timestamps_cu_receiving_core[2*i] +
            timestamps_cu_receiving_core[2*i + 1]) / 2
48      pingCoreCUdifferences[i] = midpoint_cu_receiving_core - midpoint_core_sending
49
50  for i in range(10):
51      midpoint_ue_sending = (timestamps_ue_sending[2*i] + timestamps_ue_sending[2*i +
             1]) / 2
52      midpoint_core_receiving_ue = (timestamps_core_receiving_ue[2*i] +
            timestamps_core_receiving_ue[2*i + 1]) / 2
53      pingUECoredifferences[i] = midpoint_core_receiving_ue - midpoint_ue_sending
54
55  def calculate_confidence_interval(data, confidence=0.95):
56      n = len(data)
57      mean = np.mean(data)
58      sem = stats.sem(data)  # Standard Error of the Mean
59      margin_of_error = sem * stats.t.ppf((1 + confidence) / 2, n - 1)
60      return mean, mean - margin_of_error, mean + margin_of_error
61
62  cu_du_mean, cu_du_lower, cu_du_upper = calculate_confidence_interval(
        pingCUDUdifferences)
63  core_cu_mean, core_cu_lower, core_cu_upper = calculate_confidence_interval(
        pingCoreCUdifferences)
64  ue_core_mean, ue_core_lower, ue_core_upper = calculate_confidence_interval(
        pingUECoredifferences)
65
66  x = np.arange(10)
67  cu_du_trend = np.polyfit(x, pingCUDUdifferences, 1)
68  core_cu_trend = np.polyfit(x, pingCoreCUdifferences, 1)
69  ue_core_trend = np.polyfit(x, pingUECoredifferences, 1)
70
```

```
print("CU-DU Differences:")
print(f"Mean: {cu_du_mean}, 95% Confidence Interval: [{cu_du_lower}, {cu_du_upper}], trend: [{cu_du_trend}]")
print("Core-CU Differences:")
print(f"Mean: {core_cu_mean}, 95% Confidence Interval: [{core_cu_lower}, {core_cu_upper}], trend: [{core_cu_trend}]")
print("UE-Core Differences:")
print(f"Mean: {ue_core_mean}, 95% Confidence Interval: [{ue_core_lower}, {ue_core_upper}], trend: [{ue_core_trend}]")

colors_scatter = ['#F0E68C', '#98FB98', '#87CEEB', '#4682B4']

plt.figure(figsize=(10, 5))

plt.scatter(range(10), pingCUDUdifferences, color=colors_scatter[0], label='CU-DU')
plt.scatter(range(10), pingCoreCUdifferences, color=colors_scatter[1], label='Core-CU')
plt.scatter(range(10), pingUECoredifferences, color=colors_scatter[2], label='UE-Core')
plt.xlabel('Ping Index')
plt.ylabel('Time Difference (seconds)')
plt.title('Ping Time Differences')
plt.legend()
plt.grid(True)

plt.tight_layout()
plt.show()
```

Appendix C

Capture Parsing Script for Authentication Temporal Analysis

```
import pyshark
import matplotlib.pyplot as plt
import numpy as np
from datetime import datetime
from enum import Enum, auto

class State(Enum):
    RRCSetupComplete = auto()
    RegistrationRequest = auto()
    AuthenticationResponse = auto()
    NASSecurityMode = auto()
    NASSecurityModeComplete = auto()
    SecurityMode = auto()
    SecurityModeComplete = auto()
    Finished = auto()

connection_start_times = [
    "2024-06-19 17:54:30.230899",
    "2024-06-19 17:57:09.723686",
    "2024-06-19 17:59:41.550893",
    "2024-06-19 18:00:59.815347",
    "2024-06-19 18:02:06.934552",
    "2024-06-19 18:03:11.741160",
    "2024-06-19 18:04:19.837884",
    "2024-06-19 18:08:24.613133",
    "2024-06-19 18:12:04.751336",
    "2024-06-19 18:13:24.053948"
]

rrc_setup_request_times = [
    "2024-06-19 17:54:33.682347",
```

```
    "2024-06-19 17:57:13.645549",
    "2024-06-19 17:59:45.783293",
    "2024-06-19 18:01:02.837527",
    "2024-06-19 18:02:10.719592",
    "2024-06-19 18:03:15.691218",
    "2024-06-19 18:04:23.743044",
    "2024-06-19 18:08:28.712642",
    "2024-06-19 18:12:08.767838",
    "2024-06-19 18:13:27.719287"
]

def parse_high_precision_time(time_str):
    return datetime.strptime(time_str, '%Y-%m-%d %H:%M:%S.%f')

rrcSetupRequestTime = np.array([parse_high_precision_time(t).timestamp() for t in rrc_setup_request_times])
connectionStartTime = np.array([parse_high_precision_time(t).timestamp() for t in connection_start_times])

rrcSetupCompleteTime = np.zeros(10)
securityModeCommandTime = np.zeros(10)
securityModeCompleteTime = np.zeros(10)
authenticationResponseTime = np.zeros(10)
nasSecurityModeCommandTime = np.zeros(10)
nasSecurityModeCompleteTime = np.zeros(10)
stepHTime = np.zeros(10)

stepATime = np.zeros(10)
stepBCTime = np.zeros(10)
stepDTime = np.zeros(10)
stepETime = np.zeros(10)

for indexExperiment in range(0, 10):
    print(indexExperiment)
    path = "/Users/pedro/Pictures/Tese/times/"
    current_state = State.RRCSetupComplete
    next_state = State.RRCSetupComplete

    cap = pyshark.FileCapture(path + "timing" + str(indexExperiment + 1) + "CU.pcap")

    for packet in cap:
        current_state = next_state
        if current_state == State.AuthenticationResponse: break
        match current_state:
            case State.RRCSetupComplete:
                if hasattr(packet, 'f1ap'):
                    if 'rrcSetupComplete' in str(packet.f1ap):
```

```
77                          rrcSetupCompleteTime[indexExperiment] = packet.sniff_time.
                                timestamp()
78                          next_state = State.SecurityMode
79              case State.SecurityMode:
80                  if packet.ip.dst == "10.42.0.2":
81                      if hasattr(packet, 'f1ap'):
82                          if 'Security mode command' in str(packet.f1ap):
83                              securityModeCommandTime[indexExperiment] = packet.
                                    sniff_time.timestamp()
84                              next_state = State.SecurityModeComplete
85              case State.SecurityModeComplete:
86                  if hasattr(packet, 'f1ap'):
87                      if 'Response' in str(packet.f1ap):
88                          securityModeCompleteTime[indexExperiment] = packet.
                                sniff_time.timestamp()
89                          next_state = State.AuthenticationResponse
90
91      cap.close()
92
93      cap = pyshark.FileCapture(path + "timing" + str(indexExperiment + 1) + "core.
            pcap")
94
95      for packet in cap:
96          current_state = next_state
97          if current_state == State.Finished: break
98          match current_state:
99              case State.AuthenticationResponse:
100                 if hasattr(packet, 'ngap'):
101                     if 'Authentication response parameter' in str(packet.ngap):
102                         authenticationResponseTime[indexExperiment] = packet.
                                sniff_time.timestamp()
103                         next_state = State.NASSecurityMode
104             case State.NASSecurityMode:
105                 if hasattr(packet, 'ngap'):
106                     if 'Security mode command' in str(packet.ngap):
107                         nasSecurityModeCommandTime[indexExperiment] = packet.
                                sniff_time.timestamp()
108                         next_state = State.NASSecurityModeComplete
109             case State.NASSecurityModeComplete:
110                 if hasattr(packet, 'ngap'):
111                     nasSecurityModeCompleteTime[indexExperiment] = packet.
                            sniff_time.timestamp()
112                     next_state = State.Finished
113
114     cap.close()
115     if current_state != State.Finished: print(current_state)
116
117     cap = pyshark.FileCapture(path + "timing" + str(indexExperiment + 1) + "ue.pcap
            ")
```

```
118
119     client_hello_time = None
120     change_cipher_time = None
121
122     for packet in cap:
123         if hasattr(packet, 'ip'):
124             if packet.ip.dst == "142.250.184.4":
125                 if hasattr(packet, 'tls'):
126                     if 'Client Hello' in str(packet.tls):
127                         client_hello_time = packet.sniff_time.timestamp()
128                     elif 'Change Cipher Spec'  in str(packet.tls) and
                            client_hello_time is not None:
129                         change_cipher_time = packet.sniff_time.timestamp()
130                         break
131
132     cap.close()
133
134     if client_hello_time is not None and change_cipher_time is not None:
135         stepHTime[indexExperiment] = change_cipher_time - client_hello_time
136     else:
137         print(f"Error finding TLS packets in experiment {indexExperiment}")
138
139     stepATime[indexExperiment] = rrcSetupCompleteTime[indexExperiment] -
            rrcSetupRequestTime[indexExperiment]
140     stepBCTime[indexExperiment] = authenticationResponseTime[indexExperiment] -
            rrcSetupCompleteTime[indexExperiment]
141     stepDTime[indexExperiment] = nasSecurityModeCompleteTime[indexExperiment] -
            nasSecurityModeCommandTime[indexExperiment]
142     stepETime[indexExperiment] = securityModeCompleteTime[indexExperiment] -
            securityModeCommandTime[indexExperiment]
143
144 total_step_times = np.array([stepATime.sum(), stepBCTime.sum(), stepDTime.sum(),
        stepETime.sum(), stepHTime.sum()])
145
146 total_connection_times = [securityModeCompleteTime[i] - connectionStartTime[i] for
        i in range(10)]
147 average_connection_time = np.mean(total_connection_times)
148
149 labels = ['Step A', 'Step B and C', 'Step D', 'Step E', 'Step H']
150
151 colors_scatter = ['#F0E68C', '#98FB98', '#87CEEB', '#4682B4', '#FF6347']
152
153 percent_step_times = (total_step_times / total_step_times.sum()) * 100
154
155 print(f"Average connection time: {average_connection_time:.2f} seconds")
156
157 plt.figure(figsize=(10, 5))
158 plt.bar(labels, percent_step_times, color=colors_scatter)
159 plt.title('Percentage of Total Connection Time for Each Step')
```

```
plt.ylabel('Percentage (%)')

plt.figure(figsize=(15, 6))

plt.subplot(1, 5, 1)
plt.scatter(np.ones(len(stepATime)), stepATime, color=colors_scatter[0])
plt.title('Step A')
plt.ylabel('Time')
plt.ylim(min(stepATime) * 0.9, max(stepATime) * 1.1)

plt.subplot(1, 5, 2)
plt.scatter(np.ones(len(stepBCTime)), stepBCTime, color=colors_scatter[1])
plt.title('Step B and C')
plt.ylim(min(stepBCTime) * 0.9, max(stepBCTime) * 1.1)

plt.subplot(1, 5, 3)
plt.scatter(np.ones(len(stepDTime)), stepDTime, color=colors_scatter[2])
plt.title('Step D')
plt.ylim(min(stepDTime) * 0.9, max(stepDTime) * 1.1)

plt.subplot(1, 5, 4)
plt.scatter(np.ones(len(stepETime)), stepETime, color=colors_scatter[3])
plt.title('Step E')
plt.ylim(min(stepETime) * 0.9, max(stepETime) * 1.1)

plt.subplot(1, 5, 5)
plt.scatter(np.ones(len(stepHTime)), stepHTime, color=colors_scatter[4])
plt.title('Step H')
plt.ylim(min(stepHTime) * 0.9, max(stepHTime) * 1.1)

plt.tight_layout()
plt.show()

avg_start_times = [
    np.mean(rrcSetupRequestTime),
    np.mean(rrcSetupCompleteTime),
    np.mean(nasSecurityModeCommandTime),
    np.mean(securityModeCommandTime),
]

avg_end_times = [
    np.mean(rrcSetupCompleteTime),
    np.mean(authenticationResponseTime),
    np.mean(nasSecurityModeCompleteTime),
    np.mean(securityModeCompleteTime),
]

plt.figure(figsize=(15, 6))

```

```
labels5G = ['Step A', 'Step B and C', 'Step D', 'Step E']
for i in range(len(labels5G)):
    plt.hlines(y=i, xmin=avg_start_times[i], xmax=avg_end_times[i], color=
        colors_scatter[i], linewidth=5, label=labels5G[i])

plt.yticks(range(len(labels5G)), labels5G)
plt.xlabel('Time (s)')
plt.title('Development in Time of Each Step')
plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.1), ncol=5)
plt.tight_layout()
plt.show()

average_step_times = total_step_times / 10

data = {
    "total_step_times": total_step_times.tolist(),
    "average_step_times": average_step_times.tolist(),
    "percent_step_times": percent_step_times.tolist()
}

print("Data for further use:")
print(f"total_step_times = {data['total_step_times']}")
print(f"average_step_times = {data['average_step_times']}")
print(f"percent_step_times = {data['percent_step_times']}")
```

Appendix D

Script for Mathematical Model

```
1
2  import matplotlib.pyplot as plt
3  import numpy as np
4
5  # Initial average step times
6  initial_times = np.array([0.035503935813903806, 0.020353055000305174,
       0.015170145034790038, 0.031063008308410644, 0.05731589794158935])
7  labels = ['Step A', 'Step B and C', 'Step D', 'Step E', 'Step H']
8  colors = ['#F0E68C', '#98FB98', '#87CEEB', '#4682B4', '#FF6347']
9
10 # Scaling factors for UE to gNB, gNB to 5GC, and 5GC to Server
11 scaling_factors_ue_gnb = np.array([3, 2, 2, 2, 3])
12 scaling_factors_gnb_5gc = np.array([0, 3, 2, 0, 3])
13 scaling_factors_5gc_server = np.array([0, 0, 0, 0, 3])
14
15 # Time intervals from 0 to 30 ms
16 time_intervals_0_to_30 = np.linspace(0, 0.03, 100)
17
18 # Time intervals from -8.5 ms to +16.5 ms
19 time_intervals_custom = np.linspace(-0.0085, 0.0165, 100)
20
21 # Arrays to store percentage and absolute values for UE to gNB
22 percentages_over_time_ue_gnb = np.zeros((len(time_intervals_0_to_30), len(
       initial_times)))
23 absolute_times_over_time_ue_gnb = np.zeros((len(time_intervals_0_to_30), len(
       initial_times)))
24
25 for i, t in enumerate(time_intervals_0_to_30):
26     scaled_times = initial_times + scaling_factors_ue_gnb * t
27     total_time = scaled_times.sum()
28     percentages_over_time_ue_gnb[i, :] = (scaled_times / total_time) * 100
29     absolute_times_over_time_ue_gnb[i, :] = scaled_times
30
```

```
percentages_over_time_gnb_5gc = np.zeros((len(time_intervals_0_to_30), len(
    initial_times)))
absolute_times_over_time_gnb_5gc = np.zeros((len(time_intervals_0_to_30), len(
    initial_times)))

for i, t in enumerate(time_intervals_0_to_30):
    scaled_times = initial_times + scaling_factors_gnb_5gc * t
    total_time = scaled_times.sum()
    percentages_over_time_gnb_5gc[i, :] = (scaled_times / total_time) * 100
    absolute_times_over_time_gnb_5gc[i, :] = scaled_times

percentages_over_time_5gc_server = np.zeros((len(time_intervals_custom), len(
    initial_times)))
absolute_times_over_time_5gc_server = np.zeros((len(time_intervals_custom), len(
    initial_times)))

for i, t in enumerate(time_intervals_custom):
    scaled_times = initial_times + scaling_factors_5gc_server * t
    total_time = scaled_times.sum()
    percentages_over_time_5gc_server[i, :] = (scaled_times / total_time) * 100
    absolute_times_over_time_5gc_server[i, :] = scaled_times

# Plots
plt.figure(figsize=(12, 8))
for i in range(len(initial_times)):
    plt.plot(time_intervals_0_to_30 * 1000, percentages_over_time_ue_gnb[:, i],
        label=labels[i], color=colors[i], linewidth=2.5)

plt.title('Percentage Changes Over Time (UE to gNB)')
plt.xlabel('Propagation Time between UE and gNB (ms)')
plt.ylabel('Percentage (%)')
plt.ylim(0, 50)
plt.legend()
plt.grid(True)
plt.show()

plt.figure(figsize=(12, 8))
for i in range(len(initial_times)):
    plt.plot(time_intervals_0_to_30 * 1000, absolute_times_over_time_ue_gnb[:, i],
        label=labels[i], color=colors[i], linewidth=2.5)

plt.title('Absolute Value Changes Over Time (UE to gNB)')
plt.xlabel('Propagation Time between UE and gNB (ms)')
plt.ylabel('Time (s)')
plt.legend()
plt.grid(True)
plt.show()

plt.figure(figsize=(12, 8))
```

```
for i in range(len(initial_times)):
    plt.plot(time_intervals_0_to_30 * 1000, percentages_over_time_gnb_5gc[:, i],
        label=labels[i], color=colors[i], linewidth=2.5)

plt.title('Percentage Changes Over Time (gNB to 5GC)')
plt.xlabel('Propagation Time between gNB and 5GC (ms)')
plt.ylabel('Percentage (%)')
plt.ylim(0, 50)
plt.legend()
plt.grid(True)
plt.show()

plt.figure(figsize=(12, 8))
for i in range(len(initial_times)):
    plt.plot(time_intervals_0_to_30 * 1000, absolute_times_over_time_gnb_5gc[:, i],
         label=labels[i], color=colors[i], linewidth=2.5)

plt.title('Absolute Value Changes Over Time (gNB to 5GC)')
plt.xlabel('Propagation Time between gNB and 5GC (ms)')
plt.ylabel('Time (s)')
plt.legend()
plt.grid(True)
plt.show()

plt.figure(figsize=(12, 8))
for i in range(len(initial_times)):
    plt.plot(time_intervals_custom * 1000, percentages_over_time_5gc_server[:, i],
        label=labels[i], color=colors[i], linewidth=2.5)

plt.title('Percentage Changes Over Time (5GC to Server)')
plt.xlabel('Propagation Time between 5GC and Server (ms)')
plt.ylabel('Percentage (%)')
plt.ylim(0, 50)
plt.legend()
plt.grid(True)
plt.show()

plt.figure(figsize=(12, 8))
for i in range(len(initial_times)):
    plt.plot(time_intervals_custom * 1000, absolute_times_over_time_5gc_server[:, i
        ], label=labels[i], color=colors[i], linewidth=2.5)

plt.title('Absolute Value Changes Over Time (5GC to Server)')
plt.xlabel('Propagation Time between 5GC and Server (ms)')
plt.ylabel('Time (s)')
plt.legend()
plt.grid(True)
plt.show()
```