People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
University of Mohamed Khider – BISKRA
Faculty of Exact Sciences, Science of Nature and Life
Computer Science Department

Order number : GLSD07/M2/2021

THESIS

Submitted in fulfilment of the requirements for the Masters degree
in Computer Science
Option : Software Engineering and Distributed Systems

Title

Specification and Verification of 5G protocols using mCRL2

Presented by
HAFIDI HOSSEM EDDINE

Defended in : dd/mm/2021

Defended in front of a jury composed of :

FULL NAME	GRADE	PRESIDENT
Laid Kahloul	Professor	Supervisor
FULL NAME	GRADE	PRESIDENT

Session 2021

This page left empty intentionally.

Acknowledgment

First and foremost, praises and thanks to Allah, the Almighty, for His showers of blessings throughout my research work to complete the research successfully.

I am extremely grateful to my parents for their love, prayers, caring and sacrifices for educating and preparing me for my future. Also I express my thanks to my sisters and brothers in law Yousra, Mohcen and Selsabil for their support and valuable prayers.

I would like to express my deep and sincere gratitude to my MSc supervisor, Dr. L. Kahloul M.Sc., Ph.D., Professor and Head, Centre for Linfi laboratory of Computer Science, Mohamed Kheidar University, Biskra, Algeria, for providing invaluable guidance throughout this thesis. His dynamism, vision, sincerity and motivation have deeply inspired me. He has taught me the methodology to carry out the MSc and to present the thesis works as clearly as possible. It was a great privilege and honor to work and study under his guidance. I am extremely grateful for what he has offered me. I would also like to thank him for his friendship, empathy, and great sense of humor.

I would like to thank all my professors during my studying time for their honest work and efforts in providing such a valuable courses where I am grateful to be apart of it.

I am very much thankful to my friends and MSc colleagues Sebti Med Riad, Mamen Abd El-Karim and Guerri Mohammed Fdel for staying in touch to support each others during our MSc graduation time.

Finally, my thanks go to all the people who have supported me to complete the MSc work directly or indirectly.

This page left empty intentionally.

Contents

List of Figures

Abstract

Fifth generation (5G) standard is the last telecommunication technology, widely considered to be the most important characteristics to implement in future network industry. The 5G system infrastructure contains three principle interfaces, each one follows a set of protocols defined by The 3rd Generation Partnership Project group (3GPP). For the next generation network, 3GPP specified two authentication methods systematized in two protocols namely 5G Authentication and Key Agreement (5G-AKA) and Extensible Authentication Protocol (EAP). Such protocols are provided to ensure the authentication between system's entities. Researchers have addressed an additional difficulties in security of wireless communication systems whereas system interfaces are connected in a logical topology (air transmission). The mCRL2 language and its associated toolset are formal tools used for modelling, validation and verification of concurrent systems and protocols. Our present work aims to formally re-examines as first time 5G-AKA protocol using mCRL2 language to verify such a security protocols i.e., 5G-AKA protocol. The authentication protocol model is built using Algebra of Communication Process (ACP), its properties are build with Modal mu-Calculus and properties validation exploits formal verification Model-Checking method. We designed an innovative mCRL2 model of 3GPP specification considering 5G-AKA protocol and specified some properties that represents its promised requirements to finally evaluate this protocol.

General Introduction

Context: 5G is a very important technology that provides the world new technology with a lower latency, higher bandwidth and massive connections [1] better than recent generations which makes an evolution in future network applications. However, it becomes a visible target to create complicated attacks and security threats discovery [2], [3]. This new technology improves Enhanced Mobile Broadband (eMBB), Critical Communications (CC) that require high security, Ultra Reliable and Low Latency Communications (URLLC), Massive Internet of Things (mIoT) and Flexible network operations [18] which is provided by the Standalone Architecture (SA) [4]. The SA architecture beside the Non Stand Alone (NSA) represents the full 5G system which includes a 5G Access Network (AN) entity (sometimes called interface) and a 5G Home Network (HN) interface named respectively Next Generation Radio Access Network (NG-RAN) and 5G Core Network (5GN) [4]. The interfaces are a set of interconnected network functionalities based on Service Based Architecture (SBA) framework [5], where the SBA elements are defined as Network Functions (NF) that were not existing in previous generations (including the 4G).

The entity Home Network (HN) [5] provides various functions, such as (i) Access and Mobility management Function (AMF) [14] to manage all the signalling related to mobility and security, (ii) The Session Management Function (SMF) which takes care of User Data traffic such as session establishment, and (iii) the User Plane Function (UPF) that performs the handling of user data. On the Access Network (AN) side, there exist a gNB (G Node B) entity which represents all the main tasks of AN (e.g., Radio Resource Management (RRM)). The AMF function that has responsibility to guarantee the primary Authentication Access task for each User Equipment (UE) that connects to 5G system must choose 5G Authentication and Key Agreement (5G-AKA) or an Extensible Authentication Protocol (EAP)-AKA [7]. 5G-AKA is a challenge-and-response protocol based on asymmetric cryptography where there is a shared key between User Equipment (UE) and its associated Home Network (HN). The Serving Network (SN) can authenticate a UE only if the challenge encryption is equal to the challenge response sent from UE. The encryption in both sides is based on a one way Key

Derivation Function (KDF) where inputs can't be extracted from outputs. Indeed, in order to users benefit from 5G system,those users have to success in the authentication phase imposed by 5G-AKA.

Networking protocols in general and authentication protocols like the ones used in 5G, require to be reliable. In order to insure this reliability, software engineering in verification of communication systems introduced different approaches where the most ambitious are Formal Methods [26], using different formal languages. One can find several works in which formal methods were used to specify, verify and evaluate networking and communication protocols ([27], [36], [37], [38], [39], [40], [41], [42] and [43]). However, there are few works which have exploited formal methods to analyse 5G authentication protocols such as [45], [11] [52], [46] and [13] where verification of 5G-AKA analyses faultiness of the protocol by applying cryptographic attacks, using ProVerif [35] and Scyther [44]. In these two works, data specification structures was not considered. For the best of our knowledge, no previous work has used micro Common Representation Language 2 (mCRL2) language on 5G-AKA protocol verification. The previous works have focused on verification of correctness or on faultiness of the protocols, however in this current work, we are interested to deal with the two aspects. Moreover, the language mCRL2 provides a very rich formula language named Modal Mu-Calculus to specify properties. The mCRL2 provides an associated state-of-the-art toolset with a friendly graphical user interface (mCRL2 GUI) ready for generating automatic solutions (e.g., Boolean solver) and graphs (e.g., 2D and 3D ltsgraph generator) in different platforms.

Objective: The Objective of our work is to obtain a verifiable model of 5G-AKA protocol. Where our formal specification, verification and validation is obtained using micro Common Representation Language 2 (mCRL2). Our 5G-AKA protocol specification includes basically three kinds of components which represent the 5G standard entities (User Equipment, Access Network and Home Network). The ACP definition obtained in mCRL2 model is a set of actions that formally specify 5G-AKA behavior. The synchronization of actions in ACP definition specify the communication between the three processes of entities included in the protocol. The extended data specify the encrypted messages exchanged between interfaces. The properties to be checked in the

5G-AKA protocol are specified as Modal mu-Calculus [28] formulas. The verification of those properties is achieved using model checking through mCRL2 Parametrized Boolean Equation System (PBES) solver [19], [29].

Outlines: After this general introduction, this manuscript is composed of three chapters:

1. Chapter 1: 5G Background, where we will introduce 5G system architecture and infrastructure that provide numerous advantages in networking with focusing more on 5G authentication side using 5G-AKA protocol.

2. Chapter 2: Formal Methods, which introduces the formal methods using mCRL2 specification and verification language.

3. Chapter 3: Specification and Verification, which includes the obtained model using mCRL2 with specification and verification of 5G-AKA protocol.

These three chapters are followed with a general conclusion which presents the results obtained and gives a set of perspectives.

Chapter 1

5G Background

1.1 Introduction

This chapter details 5G architecture and infrastructure including protocols used in authentication task. The chapter is composed of 7 sections including conclusion. The main objective of this chapter is to investigate the 5G system from the general architecture into deeply inside the protocols to make it possible for highlighting authentication protocols i.e., Authentication & Key Agreement (5G-AKA) and Extensible Authentication Protocol (EAP)-AKA. The 5G-AKA protocol represents one of two methods of authenticating users in 5G. Another goal is to introduce the different features provided in 5G systems represented in the 5G New Radio (5G-NR) and new 5G Core (5GC) and their new technologies implemented. The descriptions of this chapter are mostly derived from 5G phase 1 - release 15 specification of the 3rd Generation Partnership Project (3GPP) standard [5].

1.1.1 History

The first generation (1G) was invented in 1970. The engineers started to investigate more in this telecommunication system when the interest was based on maximizing bandwidth and decreasing latency. It was deployed in Tokyo in 1979, from 1970 to 2015, 4 more generations were already specified by the 3rd Generation Partnership Project where each new generation enhances the previous generation's drawbacks.

The second generation (2G) was created in 1991; calls and digital voice were encrypted, and the telecommunications are no longer the only focus in this system. It introduced the digital features with an average bandwidth of 10 $kbit/sec$. As a result, text messages (SMS), picture messages and multimedia (MMS) were provided alongside the new technologies used; for example, Time-Division Multiple Access (TDMA) which divided time to provide many users a single frequency, and Digital cellular public land mobile telecommunication systems (DCPLMTS) which was able to transmit data and higher voice that has better quality than the 1G.

The third generation (3G) was invented in 2001; the goal depicted in this generation was to standardize networking. It used the same received and sent packets in all providers around the world to facilitate the mobility of clients. The new generation

used Code-division Multiple Access (CDMA) technology which divided code instead of time (time division used in 2G) where bandwidth was up to 2 *Mbps*.

The forth generation (4G) created in 2009; it has been provided as a standard named Long Term Evolution (LTE). It used Worldwide Interoperability for Microwave Access (WiMAX) technology and WiFi technology to supply 200 *Mbps* better speed than the 3G.

Finally the fifth generation (5G) where all the requirements of the current network industries can be achieved. It uses Orthogonal frequency-division multiplexing (OFDM) and Beam forming technology beside numerous technologies to reach the goal of holding a high number of users in the same area (the major drawback of 4G) with a high bandwidth more than 1 Gbps. The requirements that the massive Internet of Things (mIoT) will benefit and to be considered the most related one for 5G services.

1.1.2 Most concerned 5G features

The 5G came up basically to fit the needs of the current network industry requirements with providing new enhanced platforms and better performance. Highlights depicted in the following main enhanced 5G services:

- Massive Internet of Things (mIoT), where the capability to handle hundreds to billions of connected devices is possible (e.g., Smart cities). Also, the benefits of lower energy cost, cheaper hardware and wider signal coverage for challenging places are all related to mIoT requirements.

- Enhanced Mobile Broadband (eMBB), where different scenarios that need higher data-rates, mobility and connection traffic are considered (e.g., Higher video streaming quality).

- Critical Communications (CC) and Ultra Reliable and Low Latency Communications (URLLC), where lower latency is critically dependent (e.g., Media and entertainment, Medical usage). This service provides the needed requirements of such a communication with a very high availability and security.

The architectures that provide such features and services are described in the next section where all 5G entities such as users, radio and data center exchange tasks to form a 5G Stand Alone and 5G Non Stand Alone architectures.

1.2 5G architecture

1.2.1 Introduction

The fifth generation (5G) cellular network architecture (Figure 1.1) is composed of three main interfaces where every interface contains information (e.g., ID, Shared Keys) and responsible of different tasks (can be found in [5]).

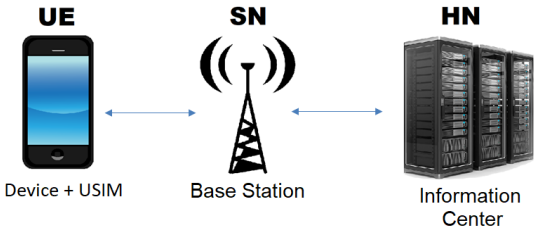

Figure 1.1: 5G Architecture.

1.2.2 Three major components of 5G

User Equipment (UE) in 5G, it is the one that connects into 5G and benefits from a combination of four information and a device that supports 5G features and services. Typically, a smart phone or a sensor with a unique Universal Subscriber Identity Module (USIM) information. The USIM is a combination of a mobile device that can be a smart phone or a sensor. 5G requires for users to contain both application and hardware equipment. Application is the Universal Subscriber Identity Module (USIM) and the hardware as Universal Integrated Circuit Card (UICC). The USIM is one of the applications that reside in UICC and composes of:

- Subscription Permanent Identifier (SUPI): a string of 15 or 16 decimal digits; it has the Mobile Country Code (MCC) in the first three digits, the Mobile Network Code (MNC) in two or three digits and a Mobile Subscriber Identification Number (MSIN) in the remaining digits.
- Public asymmetric key pkHN which is related to the associated HN.
- Long-term symmetric key K, a shared secret key between subscribers and their corresponding HN.
- Sequence number SQN which is a counter that is used to check synchronization between subscribers and their corresponding HN during authentication process.

Access Network (AN), the Radio access network is the interface responsible for sending and receiving data from UE to HN. It is responsible to handle the signalling of the whole system which can be for example authentication requests of users. It is possible for a 5G system to accept a non-3gpp access network rigorously under 3rd Generation Partnership Project (3GPP) defined standard. The 5G-NR is the new radio access network of 5G system; it is responsible mainly for signalling provided with new enhancements better than fourth generation long-term evolution (4G LTE). This base station is provided with a physical multiple antenna structure which allow it to conduct signals with a beam shape. This technology works with a coordination based on user's locations.

Home Network(HN), also called 5G Core Network (5GC), it is composed of Network Functions (NFs) that are equivalent to the entities in old generations (including 4G) (Figure 1.5) where these NFs rely on a so-called Service Based Architecture (SBA) to respond for the different system requests.

The NFs related protocols are categorized with their tasks; for example, the Access and Mobility Management Function (AMF) where that last provides Reachability, Connection, Mobility, Registration managements and different tasks in the authorization security can be found in the release 15 summary [5]. This work is related to one of the tasks provided by AMF function where we can find authentication protocols.

1.2.3 SA architecture

The Stand Alone architecture (SA) in Figure 1.2 found in [5] is the full 5G system composed of 5G core/access network only without including 4G. SA is the only architecture where all 5G services can be implemented because the three entities exist (User Equipment, Access network and Home Network).

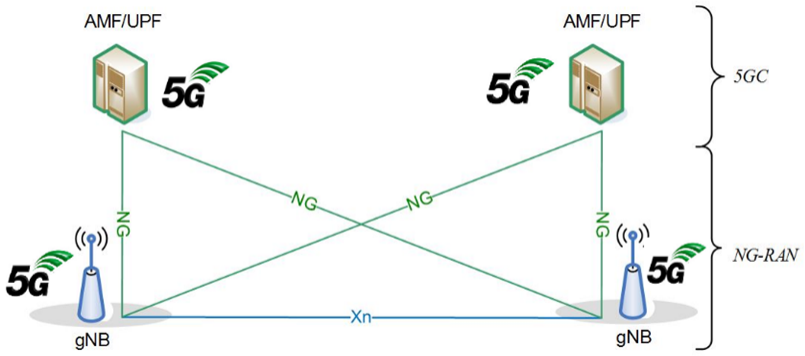

Figure 1.2: 5G-AKA, Stand alone architecture.

1.2.4 NSA architecture

The Non Stand Alone architecture (Figure 1.3) [5] is used basically as a solution to use 5G radio features earlier in 4G system. The addition enhances 4G system with a 5G access network base station. The idea is to provide all 5G services that rely on 5G Radio and does not rely on the Home Network tasks. The enhancement can be seen in latency and bandwidth in general. In other words, the services of 5G Home Network are missing and it is impossible to appear in NSA.

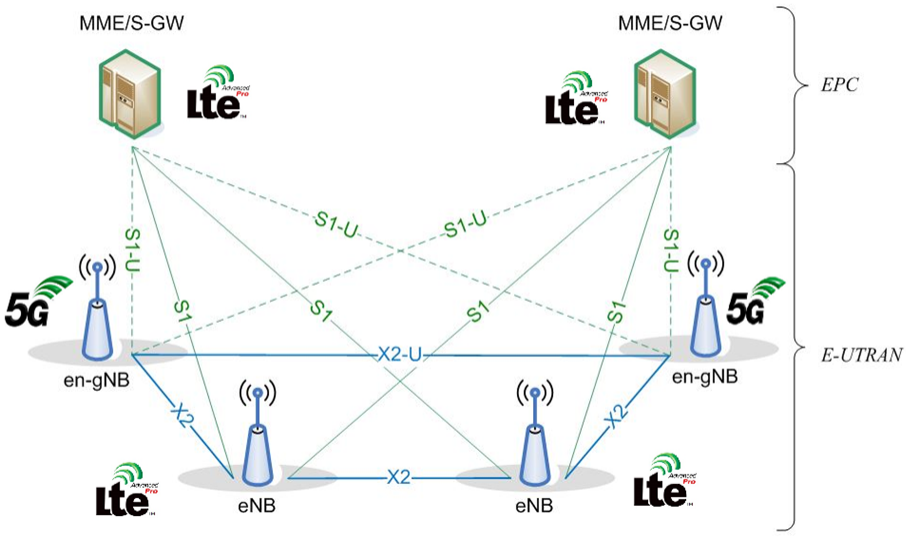

Figure 1.3: 5G-AKA, Non stand alone architecture.

1.2.5 SA and NSA differences

The key difference in architecture between SA and NSA is shown in (Figure 1.4) found in [5]. Where NSA represents other important characteristics of 5G which provide the previous 4G system with a 5G Radio Access Network (5G-RAN) also known as gNodeB (gNB). However, the NSA was provided mainly to introduce an early 5G benefits for global needs effectively (e.g., Latency and Bandwidth) as long as 4G systems remain deployed until the full 5G deployment.

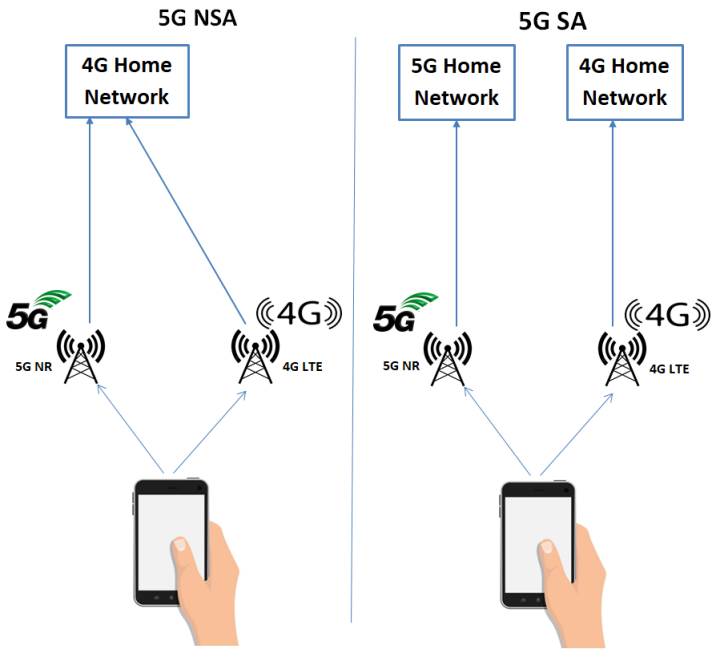

Figure 1.4: 5G SA and NSA architectures.

The services provided by 5G as new enhancements, especially those provided by Home Network (HN,) are not capable of being applied in the Non-Standalone Architecture (NSA). The purpose of providing NSA is the possibility to integrate 5G-RAN in 4G system alongside 4G-LTE.

1.3 Home Network and Related Services

The functions in HN differ depending on their role. For instance, some are related to network aspects i.e. Authentication Server Function (AUSF) (e.g., Authentication tasks) [5], Network Slice Selection Function (NSSF) (e.g.,Private Network tasks) [5] and the Policy Control Function (PCF) (e.g., Policy Rules tasks) [5]. Other functions are security related ones i.e. Security Anchor Function (SEAF), AUSF, Authentication credential Repository and Processing Function (ARPF) and Security Edge Protection Proxy (SEPP) [5]. Some functions are used to control information exchange in HN

i.e. Session Management Function (SMF), User-Plane Function (UPF) and Access Managment Funtion (AMF).

The authentication in 5G is established depending on different functions, mainly AMF function.

The AMF provides users with mobility management tasks related to different behavioral situations of User Equipment (UE); one of these tasks is the authentication. The system first needs to confirm the subordination of users to the system using access authentication, authorization, and roaming Rights. It checks network slicing support and also it chooses the Session Management Function (SMF) as long as the user authentication is successfully done.

The SMF in conjunction with the AMF function enables customized mobility management schemes such as Mobile Initiated Connection Only (MICO) or RRC Inactive state. As its name indicates, its primary tasks are configuration of routing, allocation of IP addresses, Session Management, Quality Of Service (QoS) control, downlink Data messages, UPF selection and control same as SMF selection task in AMF.

Note that The N1 is a transparent signalling support interface between UE and SN where N2-N6 are a support signalling between SN and HN to handle messages of related NFs tasks (e.g., UPF) [5].

The following figure 1.5 [47] represents the general architecture of 5G with its three components highlighting Network Functions (NFs) of Home Network (HN).

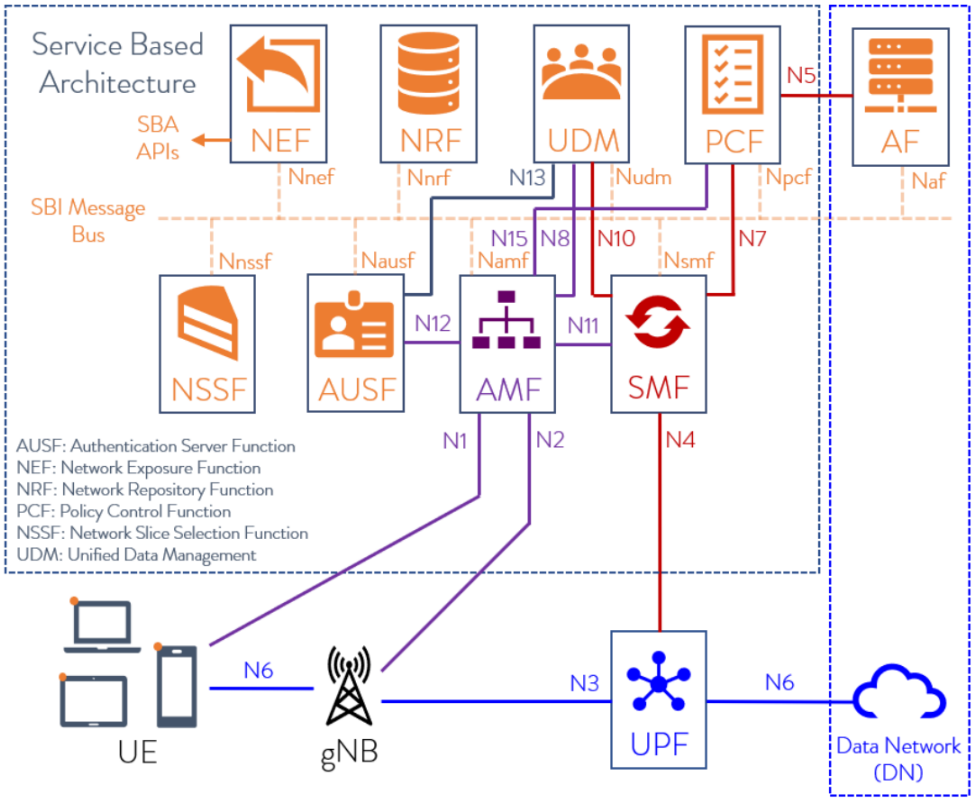

Figure 1.5: 5G Home Network Functions (NFs).

1.3.1 Services and their Network Functions

Every network function is created and categorized for specific tasks. With collaborating different tasks that 5G standard defines, one can benefit of many services with the presence of all 5G entities (User Equipment, Access Network and Home Network). The main services with their responsible NFs that can be provided in 5G are shown below.

1.3.1.1 Local hosting and edge computing

Reducing latency in 5G can be related to many parts where the system relies on; for example, Hosting Environment Service that reduces latency with evading pressure.

This service provides a closer machine-to-machine execution in a Distributed Hosted Services (aka Local Services) instead of a centred system. Different network topologies are shown in figure 1.6.

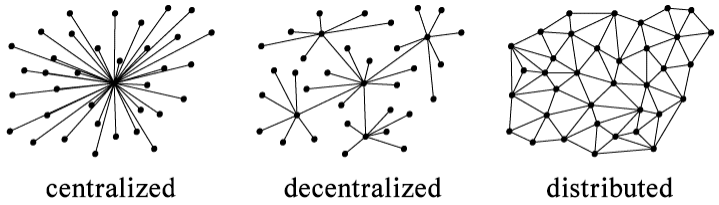

Figure 1.6: Different Network Topology.

The distribution is mainly provided using User Plane Network Function (UPF) under a service of flexibility selection and changing of hosted services. This service provides less pressure and brings balance into the network load during massive usage and users mobility. Both Local hosting and Edge Computing provide the flexibility of choosing the best execution environment depending mainly on user location, calculations and environment performance.

1.3.1.2 Network slicing service

Network slicing is one of the major characteristics in 5G. It provides the ability to customize the network usage depending on users subscriptions. In addition, it defines the network usage with different usages; each usage is associated to different categories. The networks now are named slices, and users can use them depending on what slice is bound to them. For example, a slice characterized for critical IoT users where latency must be very low but bandwidth usage is not dependent; however, another slice can be characterized for massive IoT networks where bandwidth usage is very high but latency is not dependent. Moreover, network providers can choose the number of slices and what services to deploy in it by choice. The 3GPP group already defined a three general types of slices [5] where each slice has defined characteristics.

- Type 1 eMBB slice, for enhanced Mobile Broadband

- Type 2 URLLC slice, for Ultra Reliable Low Latency Communications (URLLC)
- Type 3 mIoT slice, for massive Internet Of Things (mIoT)

1.3.1.3 Unified Access Control service

The congestion in communication systems is a common situation when it comes to a system like 5G due to the massive connections expected in the system. Different aspects are considered to choose which user connection must be provided or not to solve this issue. Aspects such as policies definition from operators, the available services and subscriber profile. In such cases, Unified Access Control categorizes user connections to solve congestion based on Access Identities and Access Categories for category restrictions.

1.3.1.4 Support of 3GPP and non-3GPP access service

The 5G system is deployed mainly following 3GPP technology standard. Operators are allowed to use different Access Network (AN) where non 3GPP technologies (Including non trusted ones) to access 5G Home Network (HN). The feature provides users with better experience considering the availability of resources. The more multiple access technologies in a system, the more available resources are provided. The framework for access technologies based on 3GPP or not is provided by Authentication Server Function (AUSF) function. Users in cases where their mobility is registered in both 3GPP and non-3GPP technologies are accompanied with a Globally Unique Temporary Identifier (GUTI) instead of the 5G globally unique Subscription Permanent Identifier (SUPI).

1.3.1.5 Other services

The services highlighted before (Local Hosting, Edge Computing, Network Slicing, Unified Access Control and Support of non 3GPP access) are not the only services that the system provides. 5G provides more services such as Multimedia Priority Services

(MPS), Public Warning System (PWS) and Mission Critical Services (MCS). All services in 5G have a long range of usage that can be found more listed in release 15 summary of 3GPP specification [5].

1.3.2 Protocols

The Home Network functions provide (the previous services we mentioned and all remaining services) an APIs stack. The stack is shown in Figure 1.7 where TCP is the Transport Layer ,TLS is the security transport and HTTP/2 is the application layer.

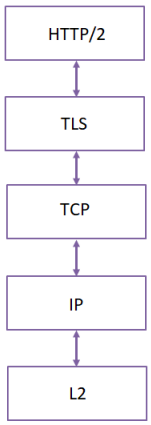

Figure 1.7: 5G-HN Service stack design

Protocols in 5G are categorized in different protocols. The most related protocols of Home Network (HN) related to this work (Authentication) are Non-Access-Stratum (NAS).

1.4 Access Network and related Functions

The 5G NG-RAN is represented with only one entity rather than multiple functions like 5G Home Network; it is shown in Figure 1.8 based on 3GPP standard.

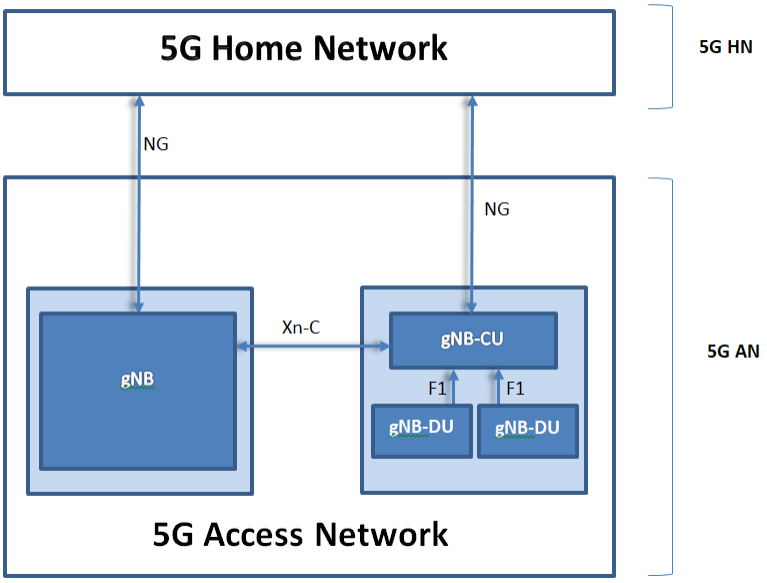

Figure 1.8: Architecture of an Access Network

The 5G-RAN is composed of three types of nodes;i.e, generation Node B (gNB), gNB Centred unit (gNB-CU) and gNB Distributed Unit (gNB-DU) where both gNB-CU and gNB-DU are composed to co-operate as a distributed Access Network which help to be a part of many features such as User Mobility. Every distributed unit gNB-DU can connect only one gNB-CU central unit. A variant applications are possible with this form of distribution like the handling of users mobility (location changing).

The interfaces Xn, F1 and NG link the AN nodes of 5G in a very similar way to the ones in 4G. The interface Xn in 5G is similar to the interface X2 in 4G and the interface NG in 5G is similar to the interface S1 in 4G. The interfaces of 4G obtain an enhancement in order to respond to 5G appeared in some scenarios such as dual connectivity of NG-RAN and LTE. In 5G system, the interfaces responsible for connecting nodes are Xn-C , F1 and NG that enable user/control plane between components [6]. Each interface is defined for specific nodes where each linking interface is shown below.

- The Xn-C interface is a link between gNB & gNB-CU

- The F1 interface is a link between the distributed nodes gNB-CU and gNB-DU.
- The NG interface is a link between 5G Home Network (HN) and

1.4.1 AN Control Plane and User Plane

The traffic control routing in AN is basically similar to 4G using Access and Mobility Function (AMF) in 5G instead of Mobility Management Entity (MME) in 4G. The control plane is shown in (Figure 1.9).

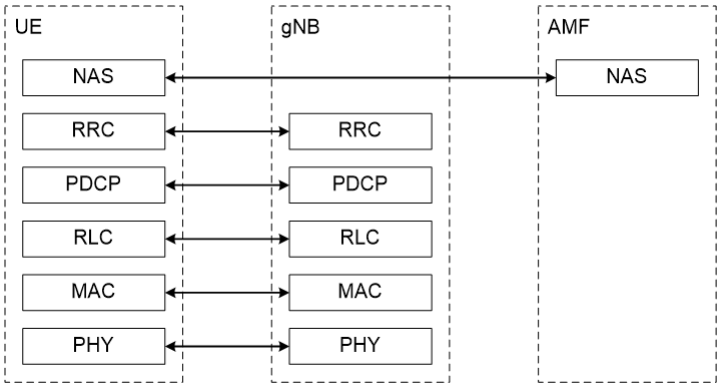

Figure 1.9: 5G Control Plane Stack

- The physical layer (PHY) is a signal modulation and demodulation in AN
- The Media Access Control (MAC) requires different tasks i.e. Mapping between channels, User Equipment (UE) priorities and information scheduling.
- The Radio Link Control (RLC) sublayer requires mainly the transferring of Protocol Data Units (PDUs), Service Data Units (SDUs) discard, segmentation and re-segmentation and re-establishment and detection of protocol errors.
- The Packet Data Convergence Protocol (PDCP) sublayer is responsible for user data transfer where the main functions required are counting sequence, ciphering and deciphering, rejecting PDCP and SDU and user data transfer.

User and Control plane stacks differ in the Service Data Adaptation Protocol (SDAP) and Radio Resource Control (RRC) sublayers respectively. The SDAP requires such services of User Plane stack related to QoS flow mapping and marking between user and 5G AN. The RRC requires mainly Mobility functions and key management security services, selection and deselection of User Equipment (UE) cell with controlling it, managing connections of RRC between UE and NG-RAN.

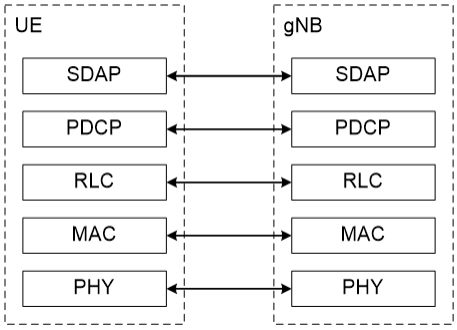

Figure 1.10: 5G User Plane Stack

1.4.2 Massive Multiple-input and multiple-output

Massive Multiple-input and multiple-output (mMIMO) is a radio technology that enhances the old MIMO with a higher scale for a wider capacity and coverage. The mMIMO requires multiple receiving and sending antennas for both 5G Radio Access Network (5G-RAN) and User Equipment (UE).

1.4.3 Beam Forming

Beam forming is a new implemented technique in 5G network (it did not exist in 4G), and it represents an advanced signaling with identical phase and wavelength, all to create a wave form depending on users directed positions. An illustration of beam forming is shown in Figure 1.11.

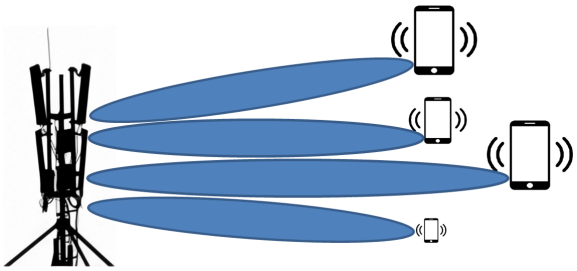

Figure 1.11: Simple Illustration of Beam Forming

1.5 Security Overall

Many upgrades applied in 5G security to correct the reported issues in the previous systems (including 4G network). One of the most important security side in such a wireless system is Authentication security.

1.5.1 Authentication

The system and users are mutually authenticated based on primary authentication, just like 4G with some enhancements. The information of the subscribers was able to be found in the air where possible attacks were easier to apply. Two methods are introduced against this issue; i.e, 5G Authentication and Key Agreement (5G-AKA) and Extensible Authentication Protocol (EAP)-AKA. The 3GPP applied cryptography in both authentication solutions (5G-AKA and EAP-AKA) to conceal information without sending a clear text. Currently in 5G system, critical information of User Equipment (UE) are hidden except for its provider, and the Universal Subscriber Identity Module (USIM) accessible by the Union for International Cancer Control (UICC) circuit in UE.

1.5.1.1 Primary Authentication

The 5G-AKA and EAP-AKA are picked in this step depending on the initial phase of primary authentication. Then, the chosen protocol will be considered to authenticate users. Once the authentication succeeds, the primary authentication is done. [5].

1.5.1.2 Secondary authentication

The secondary authentication in 5G is an authentication with data network that is not related to operator connections. The same service was available in 4G but it is enhanced and implemented with more security methods.

1.5.2 Other enhanced 5G security

The majority of 5G security depends on the new features provided in the system such as:

- Mobility security.
- Inter-operator security.
- Key hierarchy security.
- Service based architecture (SBA) security.
- Central Unit (CU) - Distributed Unit (DU) security.
- Privacy security.
- 5G Authentication security.

In the next section, we will focus on authentication security and the protocols responsible to complete this task.

1.6 Authentication

The 5G authentication is generally composed of two phases; the first phase decides which authentication method to be used in the second phase. Both phases are described as follows:

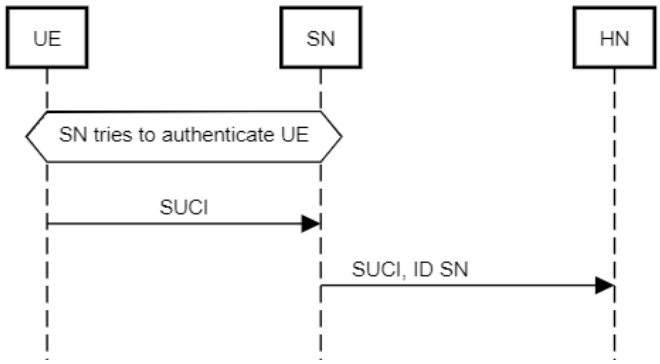

Figure 1.12: 5G Authentication Initialization.

1.6.1 Phase (i)

The initial phase of authentication (Figure 1.12) starts when Serving Network (SN) triggers a mutual authentication for User Equipment (UE). The UE conceals the couple information Subscription Permanent Identifier $SUPI$ and Random Number R (generated by UE) using HN public key $pkHN$ located in the triggered UE. The result is a concealed $SUPI$ called Subscription Concealed Identifier $SUCI$. The new R value is a random generated number in UE. The following equation is related to SUCI creation

$$SUCI = \langle aenc\left(\langle SUPI, R\rangle, pkHN\right), idHN\rangle \tag{1.1}$$

The function $aenc(.)$ in the previous equation denotes asymmetric encryption giving parameters $SUPI$, R and public key of Home Network $pkHN$ found in the triggered User Equipment (UE). The results of concealing is sent accompanied with the $idHN$ to the corresponding Serving Network (SN).

Then SN forwards the concealment SUCI accompanied with its ID to Home Network (HN).

The HN reveals Subscription Concealed Identifier (SUCI) content using its own private key as asymmetric encryption purpose which gives the results $SUPI$ corresponding the triggered UE for authentication. The HN then decides which authentication method to choose either 5G-AKA or EAP-AKA method depending on the results of decrypting $SUCI$ into a $SUPI$. The authentication methods are 5G Authentication and Key

Agreement (5G-AKA) and Extensible Authentication Protocol (EAP)-AKA. In phase (ii), we focus on the the first authentication method 5G-AKA.

1.6.2 Phase (ii)

When HN chooses 5G-AKA, it starts to create challenge-response messages that will be sent to UE. These kind of challenges use one way encryption functions that use a long-term shared secret key K. The key K in Home Network (HN) and in Universal Subscriber Identity Module (USIM) of triggered UE are used for challenge-response encryption. As a result, if the corresponding UE could make it possible to encrypt the random number sent by HN to him, and gets the same results with its own shared key K, the authentication is successful for HN side. Challenge-response elements are shown bellow and in Figure 1.13.

- AUTN AUthentication TokeN.
- R a random number (The challenge).
- $HXRES^*$ Hashed Expected Results.
- K_{SEAF} The key for secure channel used between UE and SN when authentication is completed.

The initial inputs of the challenge-reponse functions are

- SQN_{HN} , Sequence Number of HN using for synchronization of UE and HN.
- K , Long-term shared secret key.
- R , Random number.
- SN_{NAME} It contains the ID of SN.

The HN output challenge-response messages are included in an Authentication Vector (AV) having four elements. Challenge creation functions in HN are detailed in

Figure 1.13 where the orange color (circles) is the functions, amethyst color (rectangles) is the inputs and outputs between functions, red color represents the challenge output (the information AUTN, R, $HXRES^*$ and K_{SEAF}), green color (rectangles) is the initial inputs where functions start and finally the red color surrounded with a green color (rectangle) is an initial input and final output at the same time representing one information which is the Random Number R that is considered sometimes as an input and sometimes as a challenge output.

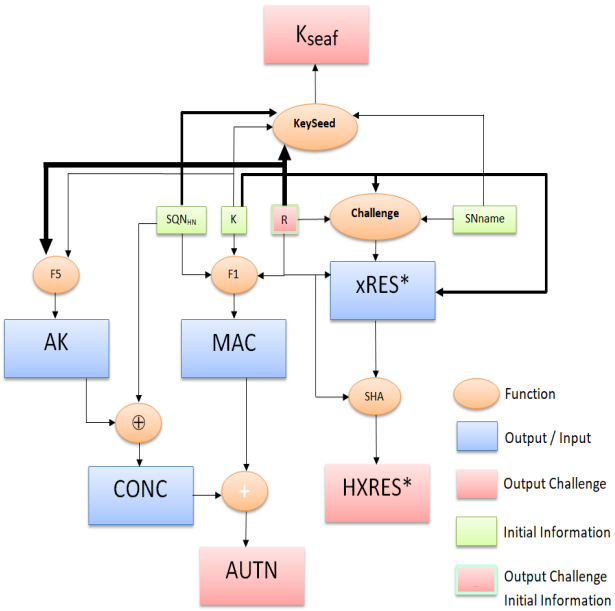

Figure 1.13: 5G-AKA Creation of Authentication Vector Challenge.

The challenge initial inputs are Sequence Number of HN SQN_{HN} , K , R and SN_{NAME}. The functions $F1$ and $F5$ are one-way keyed cryptographic functions. The $F1$ takes three parameters that are SQN_{HN}, K and R to create a code MAC and the $F5$ takes two parameters that are K and R to create an AK. The Xor $\oplus$ function takes two parameters that are the result output AK from $F5$ and Sequence number of Hn SQN_{HN} to create a concealed information $CONC$. The $CONC$ of Xor and the code MAC from $F1$ function are combined to form the AUthentication TokeN $AUTN$ which is an element of Authentication Vector AV of the challenge.

A function called *Challenge* takes three parameters which are R, SN_{NAME} and K to create the Expected Results $xRES^*$. The $xRES^*$ will not be sent in a clear text to the SN in case of a possible malicious SN attack. Using obtained results, the function $SHA256$ takes two parameters; one of them is $xRES^*$ and the other one is the Random R. It creates the Hashed Expected Result $HXRES^*$ which represents an AV element. Finally, the function $KeySeed$ takes four parameters that are Sequence number of Hn SQN_{HN}, key K, random R and SN_{NAME} which are the initial inputs and it creates the K_{SEAF} that will be used as a key channel for exchanging messages. In the HN, after creating the Authentication Vector AV, it sends that last to the Serving Network (SN).

The SN keeps the $HXRES^*$ and K_{SEAF} elements from AV and sends only two elements $AUTN$ and R to UE where the stored information is used in the following steps (See Figure 1.14).

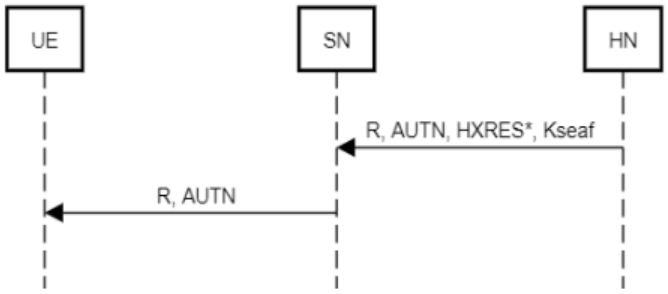

Figure 1.14: 5G-AKA, HN Challenge Results Transfer.

The triggered User Equipment (UE) receives the $AUTN$ and R. Then, it starts creating a response message using the hidden information which is supposed to be revealed only by the corresponding UE. The UE has similar functions of Home Network (HN) such as the one way cryptographic functions $F1$ and $F5$. The function $F5$ takes two parameters K and the random sent number from HN R to conceal it as AK. The function $F1$ can not execute since it needs SQN_{HN} which is hidden. The function Xor $\oplus$ extracts SQN_{HN} from $CONC$ sent by HN in Authentication Token $AUTN$. It makes it unhidden so it would be used in $F1$ function following the next equation (Note that Xor function using three elements A,B and C can extract A from a known B and

C like $C = A \oplus B \rightarrow A = C \oplus B$):

$$CONC = SQN_{HN} \oplus AK \rightarrow SQN_{HN} = CONC \oplus AK \tag{1.2}$$

The function Xor $\oplus$ takes two parameters $CONC$ and AK to reveal the Sequence Number of HN SQN_{HN}. As a result, the function $F1$ can now take the parameters SQN_{HN}, R and its own key K to create a code MAC. The purpose of creating the same information in both sites (Home Network and User Equipment) is to compare the results without sending any clear text in the air. Compared results can be created only using key K and random R in both sites.

The triggered UE compares the results of code MAC received from HN in $AUTN$ (named generally xMAC in User Equipment (UE) to evade similar names) and code MAC created in UE which remains for a three comparison cases shown below and in Figure 1.15.

- The first case must check (i)(ii) where:

  (i) If the result code MAC is equal to the code MAC received from HN and,

  (ii) If the result SQN_{HN} is equal to the local SQN_{UE},

  then the authentication is considered successful in the triggered UE.

- The second case must check (i)(ii) where:

  (i) If the result code MAC is equal to the code MAC received from HN and,

  (ii) If the result SQN_{HN} is not equal to the local SQN_{UE},

  then the UE sends a synchronization failure to SN and updates the SQN_{HN} in HN with the SQN_{UE} sent by the triggered UE. The authentication is considered a fail.

- The second case must check (i)(ii) where:

  (i) If the result code MAC is not equal to the code MAC received from HN and,

  (ii) If the result SQN_{HN} is not equal to the local SQN_{UE},

then UE sends a Mac Failure message to SN. The authentication is considered a fail.

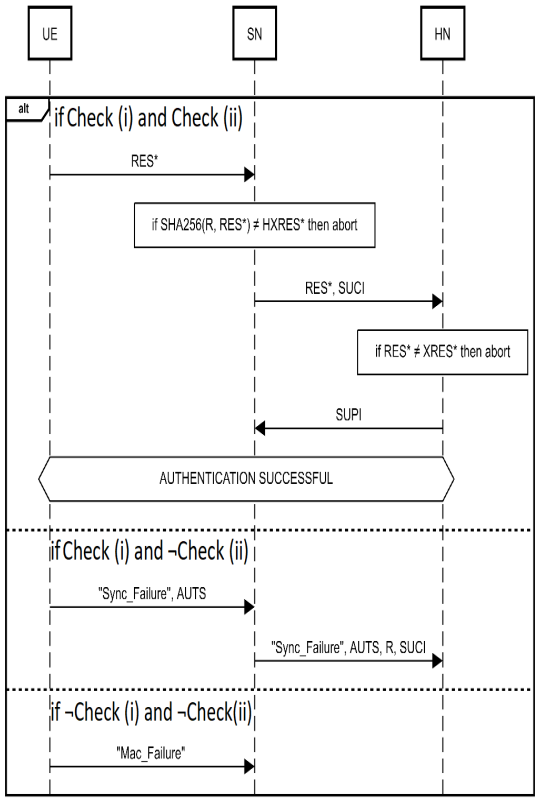

Figure 1.15: 5G-AKA, Authentication Successful.

In case one, the UE starts to create a successful authentication message. It creates the key seed K_{seaf} same as Home Network (HN) created using inputs SQN_{HN}, key K, random R and SN_{NAME}. The K_{seaf} is used as a secured session key encryption between UE, SN and HN. The function $Challenge$ takes three parameters that are key K, random R and SN_{NAME} to create a response message RES^* representing the final information that will be sent to the corresponding Serving Network (SN). The SN has already received a hashed result $HXRES^*$ and a random nonce R in Authentication Vector AV from Home Network (HN) in the initial phase. It remains only for SN to compute the hash value of RES^* received from triggered UE using $SHA256$ function

(same function used in HN to get the $HXRES^*$ hash). The results of $SHA256$ is the hashed result $HXRES^*$ based on received results of triggered UE. Now, the AN compares $HXRES^*$ of UE and $HXRES^*$ of HN and see if it matches or not which gives in two cases. Case (1) if both hashed values of $HXRES^*$ are related to UE and HN are not equal, SN will cancel the authentication considering the reception as a malicious message. Case(2) if both $HXRES^*$ of UE and HN are equal then SN sends the received response RES^* to HN accompanied with SUCI. At the moment, HN does not know either UE and SN are authenticated or not; as a result, it compares the result RES^* received from SN created by UE with its own result RES^* created at the start of the authentication which remains on two cases. Case(1) if HN RES^* equals UE RES^* then HN gets the $SUPI$ from $SUCI$ (received at the start of the authentication) and confirms authentication by sending the corresponding SUPI to SN corresponding the SN_{NAME} of the authentication. Case(2) if both RES^* values are the same, HN will cancel the authentication considering the reception as a malicious message.

In case two, after checking (i) that evaluates true and (ii) that evaluates false, The UE $F1^*$ takes three parameters that are K^*, SQN_{UE} and R to create code MAC^*. The function $F5^*$ takes two parameters K and R to create AK. The function Xor takes parameters AK (created with $F5^*$ function) and SQN_{UE} to create $CONC^*$. Both results $CONC^*$ and MAC^* are combined and sent to SN with adding a 'synchronization failure' message. The SN forwards the received message to HN with corresponding random R and the concealed information $SUCI$. HN extracts $MACS$ from received $AUTS$ and compare it with the existing MAC in HN which remains in two cases. Case(1) if comparing both code MAC evaluates to true then HN updates SQN_{HN} information exists in data base of HN with the new SQN_{UE} received in $CONC$ of $AUTS$ then authentication is considered failed but SQN is synchronized in HN.

In case three, after checking (i) and (ii) that evaluate false, the UE sends a Mac Failure to SN. The authentications has failed.

1.7 Conclusion

The 5G architecture with its three components User Equipment (UE), Access Network (AN) and Home Network (HN) collaborate fully or partially to form two types of systems that can be either a 5G Stand Alone (5G User Equipment, 5G Radio Access Network and 5G Home Network) or 5G Non-Stand Alone (5G User Equipment, 5G-RAN, 4G-LTE and 4G Home Network). The 5g security is based on mutual authentication which itself is based on cryptographic functions with exchanged messages and comparison between results to get successfully authenticated. Every entity in the system is accompanied with a set of initial information (e.g., SUPI for users, id of AN and, sqn of HN) which are used in authentication. The Figure below resumes the main refinement of this chapter.

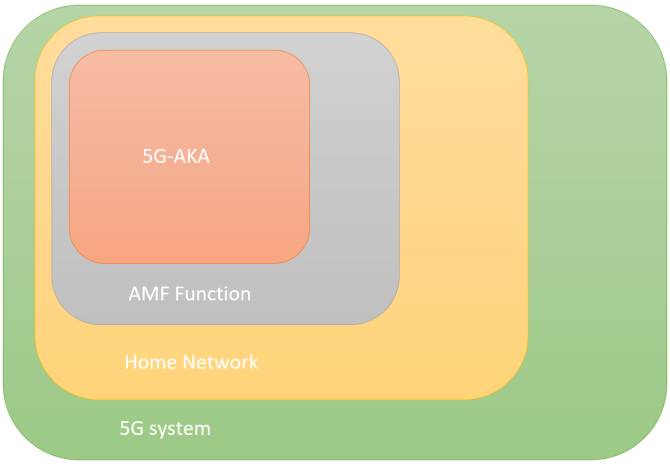

Figure 1.16: 5G Background chapter overall

In the next chapter, we will introduce rigorously Formal Methods focusing on model checking. Then, we will describe the formal language mCRL2 which will be used in the specification then the verification of 5G-AKA protocol.

Chapter 2

Formal Methods

2.1 Introduction

Such a system (e,g., Gas detection, Alarm system) behavior is totally based on software decisions that remain very destructive and sometimes mortal because of the unexpected faulty operations or security threats. This software reliability requires using state-of-the-art approaches to guarantee the correctness of a system behavior, and the absence of unwanted operations in a challenging complex systems like network protocols.

This chapter introduces one of the software engineering approaches i.e, Formal Methods that is based on Model Checking technique. The Model Checking represents mainly a rigorous method for the verification of software dependability in protocol communication network. This technique relies on mathematical aspects in different fields i.e., Algebra, Theory of numbers, Geometry and analysis which are already trusted and proved areas of research.

So many tools, techniques and methods are created based on this mathematical approach (Formal Methods) using model checking technique. We will investigate one of the state-of-the-art tools called mCRL2 language proposed by Jan Friso Groote [10].

2.2 Formal Specification

2.2.1 Introduction

To obtain a formal verifiable model defined mathematically, one should follow a formally defined syntax and semantics provided as a formal language. The results of specification provides a clear understanding of the system behavior. As a results, with having a correctly specified formal model of a system, it is possible to prove and verify that the system conform to its initial specification and promises.

A specification can be verified if it is based on a verifiable logic i.e, Propositional logic, Modal logic and Temporal logic. [33]). As a result, it can be represented as algebraic or graphical design.

Formal specification techniques rely on different specification paradigms where choosing one of them depends on what kind of system to model, their application and which phase is being specified [22].

- Specification based on history, the interest is in the temporal behaviors of the system over time.
- Specification based on states, we don't use time but instead the interest is depicted on a series of chained actions of the system. Some of the languages that implemented this paradigm specification are Z, VDM or B.
- Specification based on Transitions, the importance given to the transition actions from state-to-state. Where units representing system's values are states.
- Functional specification, most likely a well formed set of mathematical functions where two different approaches can be found,

  (i) Algebraic specification approach, it consist basically of representing data types and operations or functions on data types. The functions are defined in two types that of constructor functions to create, initialize and compose data and complex data types. The other function type is the non-constructor aka additional ones where different operations on data can be specified.

(ii) Higher-Order Functions approach, using functions grouped as a set of theories where parameters of such a function can be other functions. t increase expressively the value of the language. Examples of languages that depends on this paradigm i.e theorem prover HOL or Prototype Verification System (PVS).

- Operational specification, the system is specified as a set of processes that have a specific defined structures to which represent an implementation of an abstract processes where functions inside are connected by data flows generally in term of actions. Many languages depend on this paradigms such as Petri Nets or Process Algebra.

Considering these different paradigms, many methods are provided like Automated Theorem Proving for a sequential software system, Property-based testing specification language for software developed in functional structures, Object Constraint Language (OCL) for OOP systems and Process Algebra or Petri-Nets for concurrent and parallel systems in general.

In the next sections we will focus on Functional specification and Operational specification paradigm where both Algebraic specification approach (model mu-calculus) and Operational Specification approach (Algebra of Communicating Processes (ACP)) .

2.2.2 Process Algebra

The system model design represents the global real system behavior abstracted where every action from this behavior represents a step unit (action or event) that have a positioning and timing of execution depending on the current triggered actions in the current state.

Process algebra (PA) or process calculi specification is of type operational specification that rely basically on the process and substantiation definition including interaction mechanism in an algebraic means. It was provided mainly to represent distribution, concurrency and parallelism in related systems (e.g., World wide web, Telecommunication networks in cellular networks and telephone networks) but it didn't stop there

where also specification of dynamic systems (phenomena continuously changing) toke place in algebra in a so-called hybrid process algebras [48]. The behavior related to PA is the discrete-event systems behavior that contains dynamic discrete values over an infinite set of events or actions. These actions are positioned in the system model depending on execution order (computer programs), time or probability. Moreover, 'Algebra' relies on implementing an algebraic approach with different methods (e.g., Deduction proof) and techniques.

The PA usage is basically the call for concurrency and parallelism existence beside the ability to use sequential and alternative behavior.

Generally, three principle process algebra propositions were given which are Milner Calculus of Communicating Systems (CCS), Hoare Communicating Sequential Processes (CSP) and Bergstra Algebra of Communicating Processes (ACP) . These three Process ALgebra approaches were used in different works and projects which make it hard to differentiate between them in terms of fundamentals depicted in declaring actions and data.

In process algebra, generally actions are described with an id such as a, b, send or move where these actions are separated with a separator that generally represented as a dot '.' for example b.send.c, the behavior that represent action b followed action send and then followed by action c. If there are two or more alternative actions we use disjunction '+' to express a+b+c which are a three alternative actions that the system can execute by choice depending on specific conditions that means executing action a or action b or action c. A repetitive action can be expressed with adding a post star like move* ,b* or a*.

One need to learn a specification language based on process algebra in order to design a model of concurrent discrete-event network protocol. In this volume, we will focus on the Algebra of Communicating Processes of Bergstra [8] with extended data and time.

The formal syntax of the implemented ACP process algebra is a five-tuple $PS = (D^{mCRL2}, AD, PE, p, X^{mCRL2})$. It is detailed in the section 2.4.2 with an applied example.

2.2.3 Modal mu calculus

The reason behind using µ in µ-calculus is the addition of defining solutions to greatest and least fixpoint equations i.e $Y = f(X)$ where $X = Y$. This logic is used in properties based on kripke structure [49], typically a system that can be represented as a graph with vertices (states) and edges between vertices (transitions). The kripke definition depicted in transition system is, let say we have a set of propositions PS. The kripke definition is a 4-tuple A = (S, I, R, P) where:

- S is a set of n∈ N states
  $\{S = S_1, S_2, S_3, ..., S_n\}$
- I is a set of initial states.
- R is a transition function on states where:

$$\forall\ S_i \in\ S,\ \ \exists R_{act} \subseteq\ S * S\ \mid R_{act \in A} : S_{i-1} \rightarrow S_i$$

- P is the set of propositions or labeling where: $P :\ S\ \rightarrow\ 2^{PS}$

The model of a system matches the real behavior of a network protocol where possible actions of the protocol can be seen as a virtual paths with a known starting point. Our purpose of creating that model is to mathematically confirm the existence of the initial requirements where these represented are a properties specified using specification languages (Modal mu calculus). Modal mu-calculus with its formal syntax and semantics provide essentially the capability to represent properties using actions and operators including common temporal logic LTL, CTL and CTL* that can be expressed in this language.

The syntax of a modal µ-calculus encloses four elements Multiactions, Action formulas, State formulas and Regular formulas. Therefor, given a fixed set R for relation of symbols, an atoms set At and a set of Var for variables.

$$\phi ::= false \mid true \mid X \mid P \mid \neg\ P \mid \phi \vee \phi \mid \phi \wedge \phi \mid \langle a \rangle \phi \mid [a]\phi \mid \mu X.\phi \mid \nu X.\phi$$

The true and false are the common atomic propositional values. X is a variable. P is an atom (an action in case of ACP). However, the negation, disjunction and conjunction are the common propositional logical operators. The modal logic extends the propositional logic with a box and diamond operators from Milner's logic which are detailed in section 2.4.3 with illustration and example.

2.3 Formal Verification

2.3.1 Introduction

The formal verification is the second step of Model Checking technique as apart of Formal Methods approach. In order to check if a formally specified design of a network protocol either it satisfies a formally specified formulas or not, one needs to apply formal verification of Model checking technique. There are other possible verification methods such as Theorem proving and Equivalence checking. The majority of techniques provided are a based on Boolean equation systems such as satisfiability (SAT) solvers and Binary Decision Diagrams (BDD). We focus in this section (Formal Verification) on Model Checking using formal verification state-of-the-art language called mCRL2.

2.3.2 Model Checking

Model checking is an automatic process of analysing systems basically the ones that are based on state-transition systems. It verifies exhaustively and with symbolic traveling whether the specification holds to a given property through all state-space of the communication network protocol in an automatic way. The results is either positive or negative. If it is positive then the system holds the given property, else if it is negative then the property does not hold to the system specification and considered as a violated property with giving a counter example if possible. Model Checking method is shown in Figure 2.1 [23]

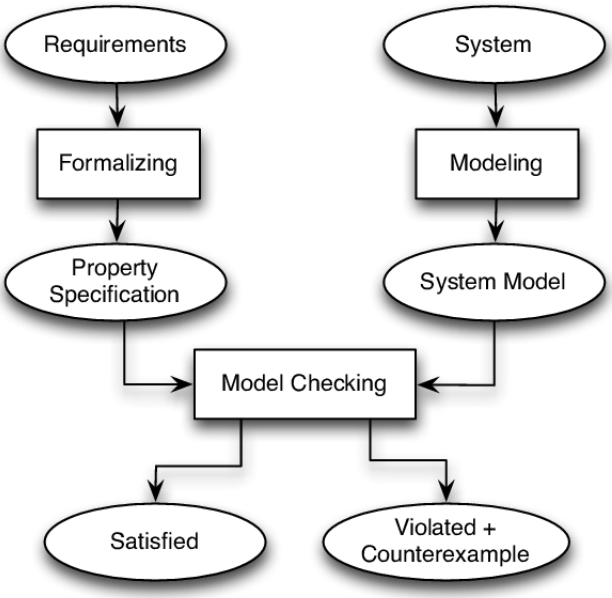

Figure 2.1: General structure of a Model Checking

Model checking process ends generally with either a true or false statement. A fundamental system properties that model checker can verify are: liveness, reachability, fairness and safety properties (Table 2.3.2).

Property type	Definition	Example
Liveness	An event will occur passing by a specific state	In Anti-virus system, if a Trojan is detected it must be blocked
Reachability	An event is possible (or not possible) to happen.	In a coffee machine, serving a coffee should be reached
Fairness	An event in some conditions must (or not) occur eventually	A process in condition of resource availability must get it eventually after requesting it
Safety	An event will never occur passing by a specific state	In chemical control system, the temperature must never be bigger than x degree

Table 2.1: Particular properties of Model Checking.

Different automatic tools created since model checking was provided, a well known tools like Simple Promela Interpreter(SPIN) [24],UPPAAL and the language we are gonna use in this work which is micro Common Representation Language 2 (mCRL2). In the next section we investigates mCRL2 formal specification and verification using model checker technique.

2.4 Formal language mCRL2

2.4.1 Introduction

We used specification and verification language mCRL2 to model 5G-AKA protocol. It uses a formal syntax and semantics that are expressively rich [17]. Thanks give to the existence of state-of-the-art toolset for automatic simulation, visualization, analyzing and

verification of behavioral properties in distributed systems, parallel computer programs and especially protocols (e.g., the network communication protocol 5G-AKA). It also contains three different sub-languages i.e, Rewrite rules for data specification, Algebra of Communicating Processes (ACP) as a process language, and Modal Mu-Calculus to specify formulas of properties. The current state-of-the-art toolset provided by mCRL2 is shown in Figure 2.2 from the official website [10].

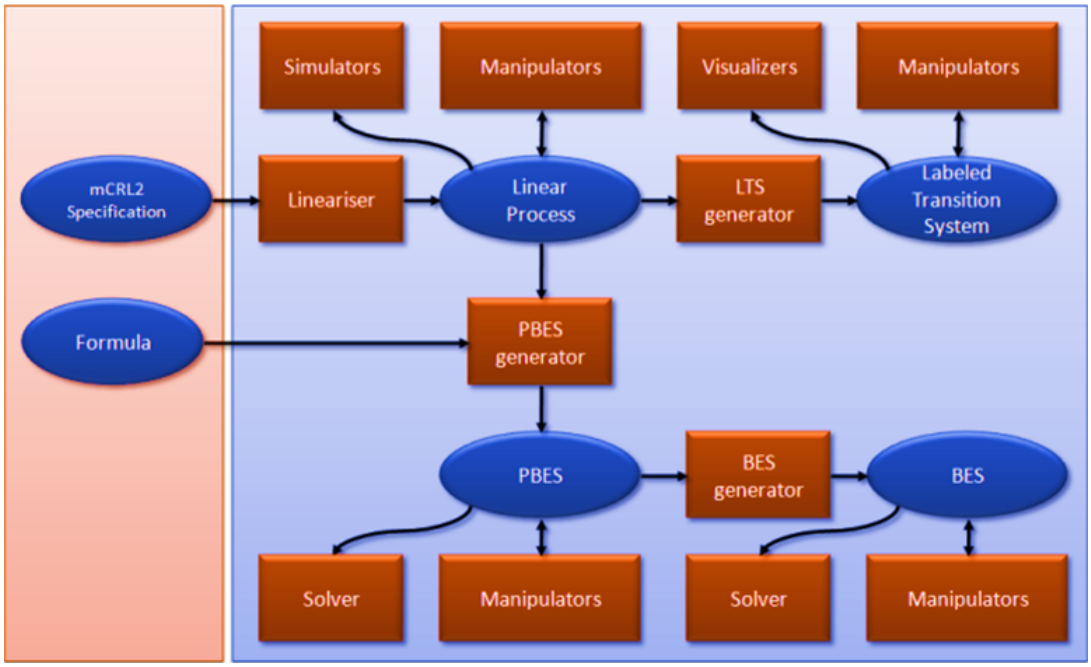

Figure 2.2: mCRL2 full toolset

We will partially on the toolset to use model checking using PBES solver. The mCRL2 related toolset needed depicted in the Figure 2.3.

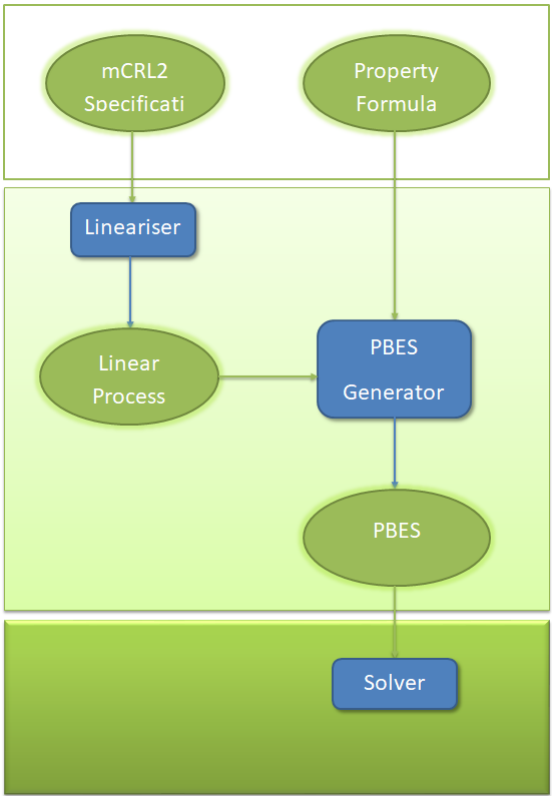

Figure 2.3: mCRL2 particular toolset

The green circles refers to specifications and their different results after applying the functions on amethyst rectangles that are process tools needed for model checking. The mCRL2, first of all, apply linearisation on mCRL2 model specification using a Lineariser tool that results to a Parametrized Boolean Equation System (PBES) using PBES Generator tool and then evaluated and solved using a PBES Solver, if the evaluation ends with a true it means that the systems holds the given property formula, else the system doesn't hold the property which gives a counter example path (if possible) from the system where the property is violated . More details and examples in the next section.

2.4.2 Formal mCRL2 Specification Model

Data creation and transformation is possible in mCRL2 where users can manipulate their own data types by creating constructors (axioms manipulation) and functions that operates on axioms through rewrite rules of mCRL2. Common data types are already defined to use like the boolean, natural number, integer and real where its predefined operations are already defined to use over new created data types of users. Also there complex data types such as sets, lists, and possible function creation for any specific complex data type.

The system behavior is based on process definition where each process contains a set of actions. Users tend to use operators provided by the language to model the system behavior that can be apply it on actions or on processes like multi-action composition '|' , sequential '.', alternative '+' and parallel '||' composition and abstraction operators. The process expression can use data specifications as initial and input parameters which fulfill data modelling of reactive systems requirements that depends on the exchange of information for example a Network communication protocol. The actions of a process can be parameterised by specified data and applying if-then-else constructs on data or on actions where process behaviour depends on information conditions (Encryption validation).

The semantics considering deduction rules of processes are represented in a Structured Operational Semantics (SOS) [50] are formally specified using mCRL2 data specification and validated by its own model checking tools following an approach called Dogfooding approach [51] where this approach definition is when a company uses its own services to create or enhance its products. The corresponding SOS rules of mCRL2 are validated using mCRL2 (rewriting rules) itself. Those SOS semantics are shown in Figure 2.4.

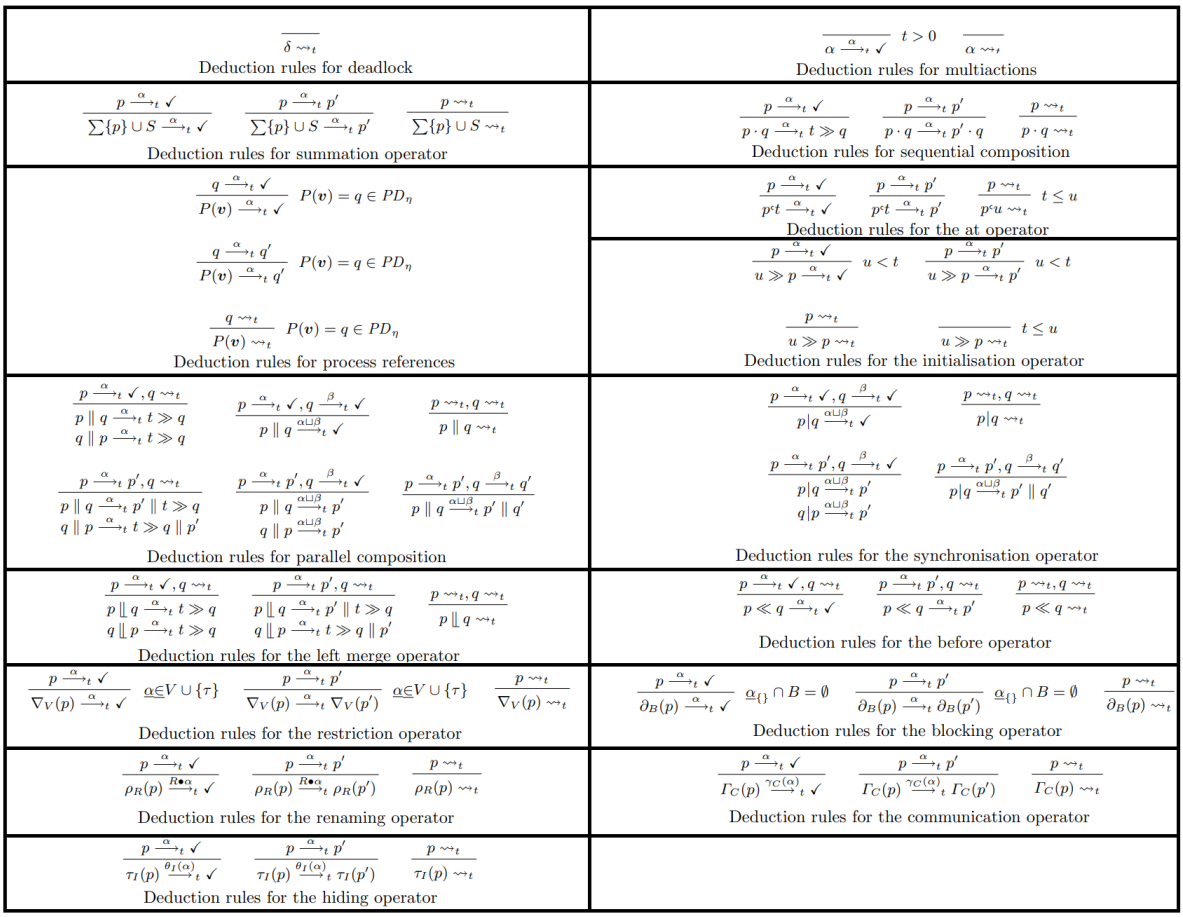

Figure 2.4: mCRL2 Structured Operational Semantics (SOS)

The generation of Labeled Transition System (LTS) is possible using mCRL2 tools (section 2.2.3). An mCRL2 process specification consists of a five-tuple,

$PS = (D^{mCRL2}, AD, PE, p, X^{mCRL2})$ where:

- D^{mCRL2} specification data,
- AD declaration of actions,
- PE equations of process set,
- p expression of process,

- X^{mCRL2} global variables set.

Here $D^{mCRL2} = (\sum^{mCRL2}, E)$ consists of $\sum^{mCRL2}$ signature and data equations set E. A signature $\sum^{mCRL2} = (S^{mCRL2}, C^{mCRL2}, M^{mCRL2})$ remains for a set of sorts names , a constructor functions set and a mapping set resp. Example of using a randomly basic system is initially defined by $PS = (D^{mCRL2}, AD, PE, p, X^{mCRL2})$ where, $D^{mCRL2} = \emptyset$, $AD = \{a, b, c, d, e, f\}$, $PE = \{p1, p2\}$ and $p = \{\emptyset\}$. The symbolic model of the initial system is defined below:

```
act
    a,b,c,d,e,f;
proc
    p1 = b.a.p1;
    p2 = a.b.p2;
init (
    allow({ a , d },
    comm({ a | b -> d  },
    p1 || p2  ) ));
```

The words 'act', 'proc' and 'init' in the example are a key words in mCRL2 used to declare actions, processes and system resp. The process definition of p1 in the example is composed of two actions: action 'b' followed by action 'a' and then process re-execution using same process name p1. The same goes with process p2, an action 'a' followed by action 'b' and re-execution of p2. System's initialisation is done using 'init' having 'allow' function as first parameter where we can allow actions to be visible in the system's LTS. In allow function, the first parameter is a set of actions to be allowed where in this example 'a' and 'd' are allowed. The second argument of init is a communication function 'comm' having two parameters where first parameter is the set of synchronized actions separated by a multi-action constructor '|' . The action 'd' is a synchronization of actions 'a' and 'b'. It is possible to have multiple synchronized actions like 'a | b | c' that synchronize into a 'd'. The second parameter of 'comm' is the parallel processes of the system. The constructor is '||' where in this example 'p1 || p2' are parallel processes that starts execution from the beginning of the system start. The LTS generation of this example using mCRL2 ltsgraph tool is shown in (Figure 2.5) where the green state is the initial state of the system.

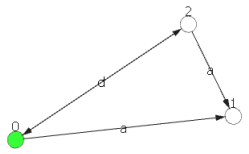

Figure 2.5: mCRL2, LTS Generation of example1.

As it was mentioned before, Expressions of processes and actions can be extended with data where process expressions it represent the initial values to be used during process execution. The data is also extended in actions expressions where the values in these data specification are an expected values from a each two synchronized actions which means if an action is specified with data, the synchronization happens only if data specifications are equal and have the same value of synchronization. The specification below is provided using same example 1 but extended with data:

```
sort myData = Nat;
sort newData = struct H(Nat,Nat);

map
funcOnNewData : myData # myData -> newData;

var
v0 : myData;
v1 : myData;

eqn
funcOnNewData(v0,v1) = H(v0,v1);

act a : Nat;
act b : Nat;
act d : Nat;
act c : newData;
act d,e,f;

proc
    p1(number1 : Nat) =
    sum num : Nat.(num == 1) -> b(num).a(num).
    c(H(number1,num)).p1(num);

    p2(number2 : Nat) = sum num : Nat.(num < 5) ->
    a(num).b(num).p2(num);
```

```
init
    ( allow({a,d},
        comm({ a|b -> d},
        p1(0) || p2(1) ))) ;
```

In the example, the new key words used for new data creation are 'sort' and 'struct'. To specify the name of a new data type we use 'sort' like in this example '$sort\ newData$'. We have two possibilities in data specification, to add a new structure definition using 'struct' followed by the structure name to be used like '$struct\ H(Nat, Nat)$;' where 'H' is meant to be a structure composed of two natural numbers to represent a new declared complex data structure to be manipulated using declared functions. The functions are mapped using 'map' where input and output data types are defined. The equations determines what functions does. In this example the function 'funcOnNewData' takes two variables v0,v1 of type 'myData' that were declared already using key word 'var' and returns an instance of 'newData' data using 'H(v0,v1)' structure exactly as it is mapped. If the input or output doesn't match function mapping, an error occurs while parsing the system specification.

The 'sum' operator is a very expensive solution in asserting data. It generates an infinite possible values for a given data structure where in this example '$sum\ num : Nat$' generates all possible natural numbers which will provide all possible states with all existing values, it can be seen as an unlimited field of data structure definition '$]-\infty, +\infty[$'.

For this reason, the sum should have an upper and lower boundary with using if-then-else conditions. In the previous example 'Example 1' we used the conditions '$(num == 1) \rightarrow$' to fix num into one value which '1' and we used '$(num < 5) \rightarrow$' to fix the upper boundary to '5' and since num is declared natural number, the lower boundary is already '0', which means num values are five values that are '4,3,2,1,0'. In case of disrespecting the boundary limitation, a state space explosion happens and the verification of the system will take extremely high amount of time that could be infinitely in some complex data cases considering even using high end machines to validate models.

The meaning of using sum in this example is that the process 'p1' have a 'sum' on 'num' that is equal to 1 where p2 have a 'sum' on 'num' that have values inferior to 5.

The synchronization of actions requires to have the same value in synchronized actions. Which means there is only one value that can make synchronization possible in this system which is the common value '1' in both actions parameters. The other values are considered will generate new states of deadlocks in this example where the LTS of this example with extended data shows that in (Figure 2.6) As it is appearing, the system

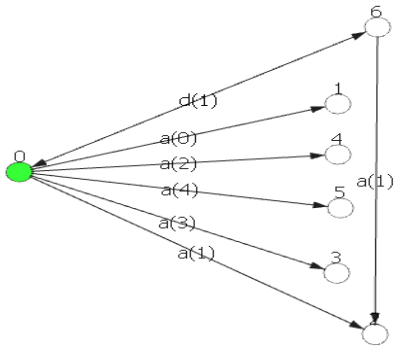

Figure 2.6: mCRL2, LTS Generation of example1.

states in a(0), a(2), a(3) and a(4) are a deadlock free cases clearly from the LTS graph generated by the same mCRL2 ltsgraph tool.

2.4.3 Formal mCRL2 Properties

Different logics can be used to describe properties where the most popular logics are Linear Temporal Logic (LTL), Computation Tree Logic (CTL) or CTL* that can do both LTL and CTL. Currently, the most expressive logic that is commonly used for expressing properties is the modal µ-calculus.

In mCRL2, specification and verification of system properties is done using the first-order modal µ-calculus which is a propositional variant of the modal µ-calculus. The modal µ-calculus is based on Hennessy-Milner logic where all formulas describe properties about states. A Hennessy-Milner logic (HML) formula , using a set of actions 'a ∈ Act' in following syntax:

$\varphi, \Psi ::= true \mid false \mid \neg\varphi \mid \varphi \wedge \Psi \mid \varphi \vee \Psi \mid \varphi \rightarrow \Psi \mid \langle a \rangle \varphi \mid [a]\varphi.$

The proposition $true$ is the formula that denotes a valid state where $false$ is the opposite

of *true*. The Negation, disjunction, conjunction and implication have the same common sense of propositional logic. $\langle a \rangle \varphi$ is a diamond pronounced "a diamond".The $[a]\varphi$ is a box pronounced "a box". The meaning of diamond and box expressed using $\exists$ and $\forall$

- $L, s \models \langle a \rangle \varphi,\ iff\ \exists s' \in S \mid s \xrightarrow{a} s'\ then\ L, s' \models \varphi$
- $L, s \models [a] \varphi,\ iff\ \forall s' \in S \mid s \xrightarrow{a} s'\ and\ L, s' \models \varphi$

A diamond expresses satisfying a formula φ at least once next to a given state. A box expresses all next possible states in a given path where φ satisfies. We resume the meaning of both constructors in the following (Figure 2.7)

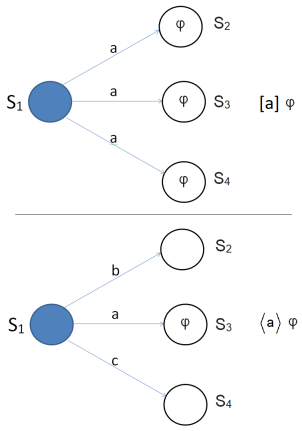

Figure 2.7: Box and diamond of Hennessy–Milner logic (HML).

An important equivalences in HML that must be considered,

- $[Act]\ true \equiv true.$
- $\langle Act \rangle\ false \equiv false.$

An Act box true is always true. Since the nonexistence of transitions in a box makes the formula evaluate to true. An Act diamond false is always false. Due to the nonexistence of transitions in a diamond makes the formula evaluate to false. Some examples of formulas:

- Freedom of deadlock:

$$[true^*] < true > true$$

- Action b may not happen after an action c, unless an action a occurs after this c and before this b:

$$[true^*.e.!f^*.g]false$$

- The action b may not occur unless an action a happens first:

$$[!e^*.f]false$$

2.5 Conclusion

In this chapter, we have exmained a set of formal methods. Indeed, we have focused on the mCRL2 language. This last one provides a rigorous specification and verification following a formal syntax and semantics where specification uses Rewrite Rules for data specification and Algebra of Communicating Processes (ACP) for process declarations. This language uses also the modal mu-calculus for formula specification.

In the next chapter, we apply the mCRL2 formal language in specifying the 5G-AKA protocol and in the verification of a set of properties.

Chapter 3

Specification and Verification of 5G-AKA protocol using mCRL2

3.1 Introduction

In this chapter, we go through the details to build a specification of 5G-AKA using mCRL2 language. The protocol model we provided is composed of data specification, process declaration and properties specification. The mCRL2 model we provide contains a data specification that represents cryptographic messages exchanged in 5G-AKA, a process declaration with synchronized actions to represent the three components of 5G-AKA i.e., User Equipment, Serving Network (or Access Network) and Homes Network (or Core Network) following the specifications provided by 3GPP group [5]. At the verification phase, We extracted different promises of 5G-AKA given by 3GPP specifications as properties to be checked against our model. Indeed, to simplify the build chain of the model, we provide two models where the first model is build upon synchronized processes and the second model is enriched with data specification.

3.2 Data Specification

The specification is given step by step depending on the need of every part during model creation. Also, we provide one example for each part we explain. The full model is provided at the end of this work.

- In our model, every atomic information used in 5G-AKA is declared as a sort of natural number specification such as K, SQN, SUPI, R, PkHN, idHN and idSN For example:

  ```
  sort K = Nat;
  ```

- Next, we specify complex data types obtained during protocol encryption using atomic sort information. The different complex data structures represented are: AENC, SUCI, MAC, AK, CONC, AUTN, RES, HXRES, K_{SEAF} and AUTS. Giving an example of:

  ```
  sort AENC = struct Aenc(SUPI,R,PkHN);
  ```

  Where AENC is the name of a data and Aenc() is a new declaration structure that represents the encryption AENC of a SUPI, R and PkHN. We mapped functions

where inputs and outputs represent atomic or complex specified data. Giving a different input information as parameters will result to have a different output results which represent the specification that allow to distinguish between before and after encryption results, generated from an interface to another. For example:

```
map CRYPT_AENC : SUPI # R # PkHN -> AENC;
```

Where $CRYPT\_AENC$ is the name of the function and # is a separator between given data parameters. Anything after implies $\rightarrow$ refers to the output data of the function. It can be a normal function that extracts a complex data from another complex data where we used this to express decomposition such as:

```
map AUTN_GET_XCONC : AUTN -> CONC;
```

- After that we declared variables using 'var' with atomic and complex specified data. For example:

```
var supi : SUPI, r : R, pkhn : PkHN;
```

- We applied patterns following every function mapping to define how these functions operate.

```
eqn CRYPT_AENC(supi,r,pkhn) = Aenc(supi,r,pkhn);
```

- We define an attitude for naming actions, since we have a communication protocol so we need a send and receive actions, where we named an action send by:

```
Interface_Receiving_Name + SEND + CurrentCounter.
```

and an action receive by:

```
Interface_Sending_Name + RECV + CurrentCounter.
```

The counter starts from 0 and increments after each executed action (See Figure 3.1). Using such a name structure will provide a fluent tracking of data flow detailed in our model. Some examples of action declaration:

```
act SN_SEND0 : SUCI ;
act UE_RECV1 : SUCI ;
act HN_SEND2 : SUCI # idSN;
act SN_RECV3 : SUCI # idSN;
act HN_SEND20;
act SN_RECV21;
```

- Actions with parameters can be specified where their synchronization applied only if the values and types of synchronized actions parameters are equals. The results of synchronization is an action that is declared with the same data type of synchronized actions.

3.3 Process Declaration and LTS Generation

We separated the model behavior implementation into two steps. A first step, where we create the process communication behavior. A second step, where we extend obtained behavior with data specification.

In the first step (communication model), we have three processes that represent the three interfaces UE, SN and HN. Each process is composed mainly of sending actions and receiving actions due to the communicative nature of the protocol. We provide each process with each related actions depending on the 5G-AKA protocol flow depicted in Figure 3.1.

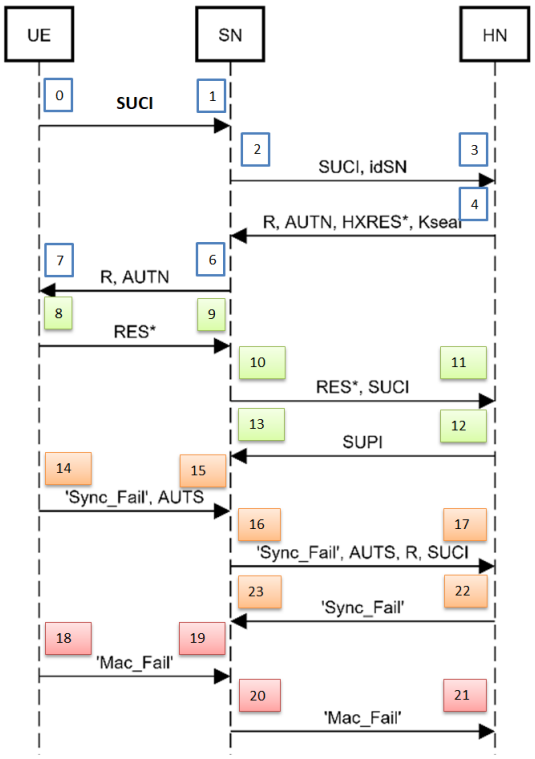

Figure 3.1: The 5G-AKA protocol obtained diagram with identification of Send-receive actions.

By following the identification below, we declare actions in mCRL2 model and we specify the protocol behavior. As shown in Figure 3.2, we specified the colored cases including related actions to create the processes that represent the three components of the system (User Equipment (UE), Serving Network (SN) and Home Network (HN)).

```
UE =
    SN_SEND0.SN_RECV7.(SN_SEND8 + SN_SEND14 + SN_SEND18).UE;
SN =
    UE_RECV1.HN_SEND2.HN_RECV5.UE_SEND6.(UE_RECV9.HN_SEND10.HN_RECV13 + UE_RECV15.HN_SEND16 + UE_RECV19.HN_SEND20+ HN_RECV23).SN;
HN =
    SN_RECV3.SN_SEND4.(SN_RECV11.SN_SEND12 + SN_RECV17 + SN_RECV21 + SN_SEND22).HN;
```

Figure 3.2: Send-receive actions identification used in our model.

In the second step (extended communication model), we extend the model with data

using 'sum' and conditions on 'sum' and using different operators provided in mCRL2 to obtain the 5G-AKA protocol model. The main data concerned to be extended in process User Equipment (UE) as atomic and initial information is the Universal Subscriber Identity Module (USIM):

- supiU , which represent the supi of UE.
- rU , the random number generated by UE
- pkhnU , the public key of the concerned Home Network (HN) in UE
- sqnU , sequence number (SQN) of UE
- idhnU , id of Home Network (HN) in UE

All UE atomic data used in this model are limited by one occurrence in order to get the abstract model, so that a simple behavior graph.

We incremented the number of USIM data after having an abstract model which remains the same results for both abstract system (1 UE) and increased system (2-5 UE). The results can be seen in Table 3.1.

supiU	pkhn	sqnU	idhnU	States
1	1	2	1	14
2	2	2	2	91
3	3	3	3	435
4	4	4	4	1347
5	5	5	5	3253

Table 3.1: Table of occurrences with their number of state space.

Names of data information used in processes declaration are concatenated with their related process names for example the *id* of Home Network (HN) in User Equipment (UE) process is $'idhnU'$, where the *id* of HN in its own process is called $'idhnH'$.

Considering *sqnU* value, it is required to have two cases (two values) which represent the synchronization failure and the authentication success.

Finally, after extending data, the results will be explained step by step as follows.

we parsed the model into mCRL2 specification parser which was a well formed mCRL2 specification.

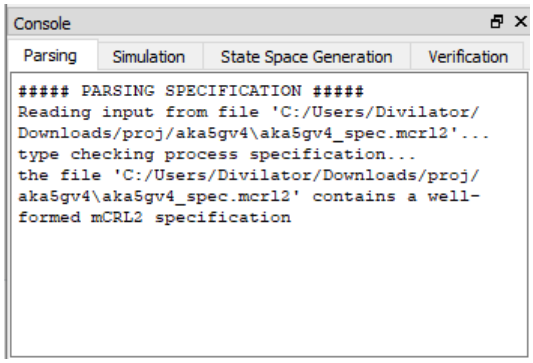

Figure 3.3: 5G-AKA protocol model successfully parsed in mCRL2.

that was validated true through mCRL2 and the LTS is generated for the abstract 5G-AKA protocol behavior (Figure 3.4) considering case of synchronization error.

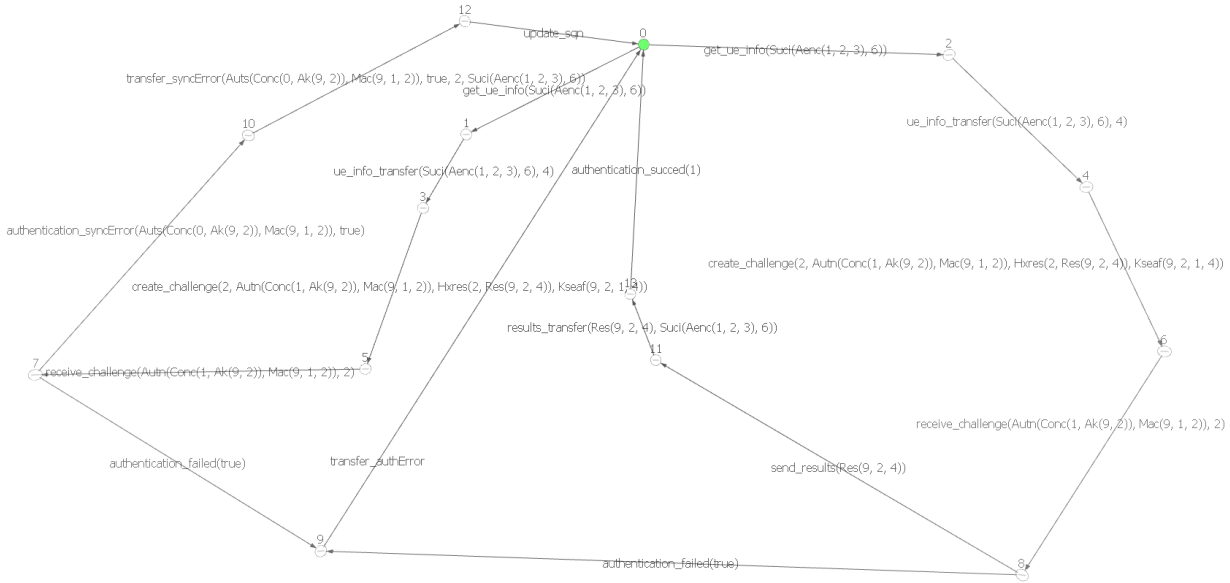

Figure 3.4: LTS of 5G-AKA protocol generated using mCRL2.

The model can be extended with more data by increasing the limits in sums con-

ditions related to supiU , pkhnU , sqnU and idhnU in mCRL2 model. The extended communication model is uploaded in [20].

3.4 Properties Specification and Verification

Since we had our model of 5G-AKA in mCRL2, we are able now to verify some properties. First the general properties (e.g., Deadlock) and then the properties of 5G-AKA extracted as formulas from 3GPP specifications. We specify then apply a verification of mCRL2 that works with PBES tool [29] after applying a linearisation on given mCRL2 specification where all parallelism is removed. The PBES tool takes both linearisation and given formula specification to evaluate a logical equation. Below, giving atomic information of UE 1, we list all properties and their results of model checking verification in a 20 occurrences model of 5G-AKA (20 cases of a UE, a 808003 states and 1128002 transitions):

- Absence of deadlock:

  `[true*] <true> true`

  Evaluated: True .

  In the first property (absence of deadlock): which means all paths that starts from initial and executes any action (or none) for example: true.true.true path. Every action found after this path will make the formula true for that path. And so on for remaining paths.

- The system can reach a successful authentication:

  `[true*] <true*.authentication_succeed(1)> true`

  Evaluated: True .

  The second property is a reachability property: It is a 'true star box' same as the first property followed by a 'true star authentication succeed 1 diamond' then 'true'. Some actions (or none) can be executed then authentication succeed.

- An HN must create a challenge after receiving a data:

```
[true*.ue_info_transfer(Suci(Aenc(1,1,1),1),1)] <create_challenge(1,Autn(Conc(1,
    Ak(1,1)),Mac(1,1,1)),Hxres(1,Res(1,1,1)),Kseaf(1,1,1,1)) > true
```

Evaluated: True .

The third property is a liveness property: It means for all paths starting from initial state where transfering user information exists. a challenge is created using a cryptographic parameters will evaluate true.

- A UE must respond to a challenge before being authenticated:

```
[true*.receive_challenge(Autn(Conc(1,Ak(1,1)), Mac(1,1,1)),1)] <true*.
    authentication_succeed(1)>true
```

Evaluated: True .

The forth property is a safety property: For every path where a challenge is received, some actions can be executed and ends with an authentication action.

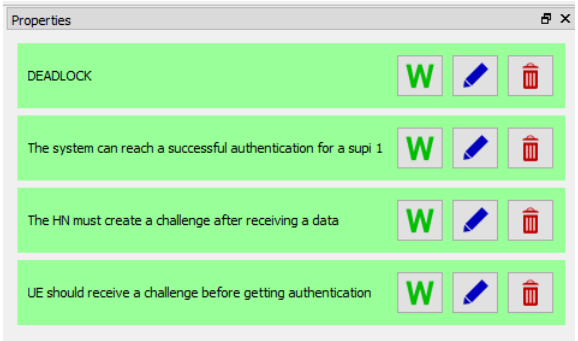

Figure 3.5: Properties evaluation through mCRL2 GUI.

3.5 Conclusion and discussion

The 5G-AKA model specification and verification using mCRL2 depending on Model checking remains strongly considered as a dependability guaranteeing tool due to the formalism used in its different tools. The results of this chapter have proved that the specification of the considered protocol using the chosen formal language is feasible. The verification phase has investigated some specific properties formulated based on the 3GPP specifications. Those properties hold on the current specification protocols.

We have done a comparative study between results of our work and those who applied formal methods on 5G authentication protocol. We have obtained the following results depicted in table 3.5.

Work	Language	Tool	Properties	Evaluation
Our work	mCRL2	mCRL2 and its toolset	Security and Behavior Correctness	Positif
[27] ELRN models	Petri nets	CPN tools	Behavior Correctness	Positif
[45] 5G-EAP	Scyther	Scyther tool	Security Faultiness	Positif (Security violation)
[52] 5G-AKA	ProVerif	ProVerif tool	Security Faultiness	Positif

Table 3.2: Comparative results of different works based on formal methods.

General Conclusion

Throughout this dissertation, we have discovered the 5G system from the general infrastructure until reaching deeply inside the protocols standard like Security Protocols. We have focused on a security protocol i.e., 5G Authentication and Key Agreement (5G-AKA) protocol which represents an enhanced version of the old authentication used in 4G with new security features that 5G promised. The 5G uses a full cryptographic messages exchanged between interfaces where the first ever message is an encrypted and concealed message Subscription Concealed Identifier (SUCI).

We have used micro Common Representation Language (mCRL2) formal language with its toolset to formally specify 5G-AKA. This language makes it possible to extend data and also to verify cryptographic protocols.

In 5G specifications, we have noticed some missing messages which can be a consequence of the early protocol release where confirmation messages are not existing. Following this incomplete specification, the protocol shows deadlock. We have added these messages assuming their existence exactly in cases where there are unsuccessful authentications decisions which are Mac Error message from SN to HN and Synchronization Error message from HN to SN.

We plan in our future works to improve our 5G-AKA model with more details and properties in parallel with upcoming 3GPP specification updates considering authentication 5G-AKA. We will go one step further into investigating the next triggered protocols after a considered successful authentication protocol in this model. Moreover, we will be capable to integrate and verify the behavior of these new investigated protocols with the protocol 5G-AKA we already specified and verified. Such a verification is possible due to the performance of evaluation of mCRL2 language.

This page left empty intentionally.

Appendices

Appendix A

Appendix for chapter 1

A.1 How and why did we choose 5G-AKA ?

We discussed first of all, all protocols in the system after going from the initial architecture (User Equipment , Access Network and Home Network) into deeply inside the protocols sets in Network Functions (NFs) which belong to HN. It is obvious that any system should knows who are its connected devices. The reason we found that this task is almost the most important focus on as a first work on 5G protocols.

For that reason, we choose to investigate our research in the authentication side where 5G system recognizes its users. After that, we have found that the system relies on the protocols of the family AKA which is a protocol standard based on asymmetric cryptography with exchanged keys. Next, we found that 5G-AKA and EAP-AKA protocols are both used in authentication where differences between them in term of security is not relevant. So we decided to focus on one of the protocols and we picked 5G-AKA.

Appendix B

Appendix for chapter 3

B.1 Details on how did we obtain our model

Generally, after investigating 5G system and 5G-AKA protocol, we started to model the protocol semi-formally using UML according to 3rd Generation Partnership Project (3GPP) described. The model is created in nine steps where we explain these steps one by one:

1. Data creation.
2. Functions expression mappings.
3. Declaration of variables.
4. Definition of functions depending on expression mappings.
5. Actions declarations.
6. Process declarations, UE, SN and HN.
7. System initialisation, Allowed actions that can be considered in the global system behavior
8. System initialisation, Communication or synchronization between actions

9. System initialisation, Parallel processes that are specified in this system

B.1.1 Data creation

We collected needed data and we found that we have two types of data (i) Atomic data and (ii) Complex data.

B.1.1.1 Atomic data

The information that doesn't need other data to be created which is generally information that comes initially in entities (User Equipment (UE), Access Network (AN) and Home Network (HN)). Our Atomic data:

```
% Atomic data specification
sort K = Nat;
sort SQN = Nat;
sort SUPI = Nat;
sort R = Nat;
sort PkHN = Nat;
sort idHN = Nat;
sort idSN = Nat;
```

B.1.1.2 Complex data

It is the information that needs at least two atomic information to be created, generally this information represents the results of cryptography or combinations. Our complex data:

```
% Complex data specification
sort AENC = struct Aenc(SUPI,R,PkHN);
sort SUCI = struct Suci(AENC,idHN);
sort MAC = struct Mac(K,SQN,R);
sort AK = struct Ak(K,R) ;
sort CONC = struct Conc(SQN,AK);
sort AUTN = struct Autn(CONC,MAC);
```

```
sort RES = struct Res(K,R,idSN);
sort HXRES = struct Hxres(R,RES);
sort KSEAF = struct Kseaf(K,R,SQN,idSN);
sort AUTS = struct Auts(CONC,MAC);
```

B.1.2 Functions expression mappings.

The mapping of a function is the expression of it without describing what it does. It is similar to prototypes in C language. Our functions:

```
% Functions expression mappings
map CRYPT_AENC : SUPI # R # PkHN -> AENC;
map CREATE_SUCI : AENC # idHN -> SUCI;
map SUCI_GET_SUPI : SUCI -> SUPI;
map GET_R : AENC -> R;
map GET_PKHN : AENC -> PkHN;
map GET_IDHN : SUCI -> idHN;
map F1 : K # SQN # R -> MAC;
map F5 : K # R -> AK;
map F1' : K # SQN # R -> MAC;
map F5' : K # R -> AK;
map CREATE_CONC : SQN # AK -> CONC;
map CREATE_AUTN : CONC # MAC -> AUTN;
map CHALLENGE : K # R # idSN -> RES;
map SHA256 : R # RES -> HXRES;
map KEYSEED : K # R # SQN # idSN -> KSEAF;
map UPDATE : SQN -> SQN;
map CREATE_AUTS : CONC # MAC -> AUTS;
map AUTN_GET_XCONC : AUTN -> CONC;
map AUTN_GET_XMAC : AUTN -> MAC;
map XCONC_GET_XSQN : AK # CONC -> SQN;
map AUTS_GET_MAC : AUTS -> MAC;
```

B.1.3 Declaration of variables

Variables in our model:

```
% Declaration of variables
var supi : SUPI,
```

```
    r : R,
    k : K,
    auts : AUTS,
    ak : AK,
    pkhn : PkHN,
    idhn : idHN,
    idsn : idSN,
    suci : SUCI,
    sqn : SQN,
    mac : MAC,
    conc : CONC,
    autn : AUTN,
    res : RES,
    hxres : HXRES,
    kseaf : KSEAF,
    aenc : AENC;
```

B.1.4 Definition of functions depending on mappings.

The equations apply patterns definition on previous created mappings and variables to define function's output. Our definition of functions:

```
% Definition of functions depending on expression mappings
eqn CRYPT_AENC(supi,r,pkhn) = Aenc(supi,r,pkhn);
    CREATE_SUCI(aenc, idhn) = Suci(aenc,idhn);
    SUCI_GET_SUPI(Suci(Aenc(supi,r,pkhn),idhn)) = supi;
    GET_R(Aenc(supi,r,pkhn)) = supi;
    GET_PKHN(Aenc(supi,r,pkhn)) = pkhn;
    GET_IDHN(Suci(aenc,idhn)) = idhn;
    F1(k,sqn,r) = Mac(k,sqn,r);
    F5(k,r) = Ak(k,r);
        F1'(k,sqn,r) = Mac(k,sqn,r);
    F5'(k,r) = Ak(k,r);
    CREATE_CONC(sqn,ak) = Conc(sqn,ak);
    CREATE_AUTN(conc, mac) = Autn(conc,mac);
    CHALLENGE(k,r,idsn) = Res(k,r,idsn);
    SHA256(r,res) = Hxres(r,res);
    KEYSEED(k,r,sqn,idsn) = Kseaf(k,r,sqn,idsn);
    UPDATE(sqn) = succ(sqn);
    CREATE_AUTS(conc,mac) = Auts(conc,mac);
    AUTN_GET_XCONC(Autn(conc,mac)) = conc;
```

```
AUTN_GET_XMAC(Autn(conc,mac)) = mac;
XCONC_GET_XSQN(ak ,Conc(sqn,ak)) = sqn;
AUTS_GET_MAC(Auts(conc,mac)) = mac;
```

B.1.5 Actions declarations.

Actions can be created after creating all data needed. Our actions:

```
% Actions declarations
act AUTH_START;
act AUTH_ACCEPT;
act SN_SEND0 : SUCI ;
act UE_RECV1 : SUCI ;
act HN_SEND2 : SUCI # idSN;
act SN_RECV3 : SUCI # idSN;
act SN_SEND4 : R # AUTN # HXRES # KSEAF;
act HN_RECV5 : R # AUTN # HXRES # KSEAF;
act UE_SEND6 : AUTN # R;
act SN_RECV7 : AUTN # R;
act SN_SEND8 : RES;
act UE_RECV9 : RES;
act HN_SEND10 : RES # SUCI;
act SN_RECV11 : RES # SUCI;
act SN_SEND12 : SUPI;
act HN_RECV13 : SUPI;
act SN_SEND14 : AUTS # Bool;
act UE_RECV15 : AUTS # Bool;
act HN_SEND16 : AUTS # Bool # R # SUCI;
act SN_RECV17 : AUTS # Bool # R # SUCI;
act SN_SEND18 : Bool;
act UE_RECV19 : Bool;
act HN_SEND20;
act SN_RECV21;
act SN_SEND22;
act HN_RECV23;
act update_sqn;

% Synchronization actions declaration
act prepare_system;
act get_ue_info : SUCI;
act ue_info_transfer : SUCI # idSN;
```

```
act create_challenge : R # AUTN # HXRES # KSEAF;
act receive_challenge : AUTN # R;
act send_results : RES;
act results_transfer : RES # SUCI;
act authentication_succeed : SUPI;
act authentication_syncError : AUTS # Bool;
act transfer_syncError: AUTS # Bool # R # SUCI;
act authentication_failed: Bool;
act transfer_authError;
```

B.1.6 Process declarations, UE, SN and HN.

The most important part is the Processes declarations. It defines the three entities of the system where every entity have its own actions. The following enumeration matches the numbering in our model:

1. Apply sum to create atomic data USIM (e.g., SUPI).
2. Limit declared USIM data with one occurrence.
3. The UE conceal atomic information.
4. SN creates a suciU without limiting the sum because it is already synchronized with SN_SEND0 action.
5. The only atomic information that SN have is its ID which is created as idsnS. Both data information suciU and idsnS are sent to HN through HN_SEND2.
6. Same as number 4 with additional information that is idsnS.
7. Creation of HN's atomic information.
8. The challenge creation by HN represented in the Authentication Vector AV with received data values from UE.
9. AV data rH,autnH,hxresH and kseafH are created and received through synchronized actions.

10. Only autnH and rH are sent to UE by SN.

11. UE receives the autnH and rH data values from SN. Now it starts to react for the three cases of authentication through if-than-else conditions.

12. The first case is a positive MAC and SQN comparison. The response Res is created and sent back to SN through SN_SEND8 action.

13. The SN receives Res of User using synchronization on resU summation.

14. The received resU will be forwarded accompanied with its corresponding suciU in action HN_SEND10

15. The HN receives resU using SN_RECV11 from UE.

16. A confirmation is done. The supiUA is sent to the SN.

17. The SN receives the supi sent by HN using HN_RECV13 action. The AUTHENTICATION IS SUCCESSFULLY DONE.

18. The second case evaluates positive for MACs and negative for SQNs. The reaction of UE is to create a syncError , to create auts and send it to SN accompanied with rH through action SN_SEND14

19. SN receives autsU and syncError from UE using action UE_RECV15

20. SN forwards the syncError and autsU accompanied with rH and suciU using action HN_SEND16

21. HN receives the syncError, autsH, rH and suciU through action SN_RECV17.

22. End of syncError case , where sqn is updated using action HN_RECV23. The AUTHENTICATION IS FAILED !

23. The third case evaluates negative for both MACs and SQNs. UE sends an authentication error using SN_SEND18 action.

24. The SN receives the authentication error using action UE_RECV19

25. A confirmation of fail is sent to HN using HN_SEND20

26. The HN receives the authentication error confirmation using action SN_RECV21. The AUTHENTICATION IS FAILED !

Below, the three processes declarations are provided:

```
UE =
    % 1
    sum supiU : SUPI, rU : R, pkhnU : PkHN, sqnU : SQN, idhnU : idHN.
    % 2
    (supiU == 1 && rU == 1 && pkhnU == 1 && sqnU < 2 && idhnU == 1) ->
    % 3
    SN_SEND0(Suci(Aenc(supiU,rU,pkhnU),idhnU)).
    % 11
    sum autnH : AUTN, rH : R. SN_RECV7(autnH,rH).(

    % 12
    (sum kU : K,idsnU : idSN. (kU == 1 && idsnU == 1 && AUTN_GET_XMAC(autnH) ==
    Mac(kU,XCONC_GET_XSQN(Ak(kU,rH),AUTN_GET_XCONC(autnH)),rH) && (sqnU ==
    XCONC_GET_XSQN(Ak(kU,rH),AUTN_GET_XCONC(autnH)))) -> SN_SEND8(Res(kU,rU,idsnU))) +
    % 18
    (sum kU : K,idsnU : idSN. (kU == 1 && idsnU == 1 && AUTN_GET_XMAC(autnH) ==
    Mac(kU,XCONC_GET_XSQN(Ak(kU,rH),AUTN_GET_XCONC(autnH)),rH) && (sqnU !=
    XCONC_GET_XSQN(Ak(kU,rH),AUTN_GET_XCONC(autnH)))) ->
     sum syncError : Bool.(syncError == true)->
    SN_SEND14(Auts(Conc(sqnU,Ak(kU,rU)),Mac(kU,XCONC_GET_XSQN(Ak(kU,rH),AUTN_GET_XCONC(autnH)
    ),rH)),syncError)) +
    % 23
    (sum authError : Bool. (authError == true) -> SN_SEND18(authError))).UE;


SN =
    % 4
    sum suciU:SUCI.UE_RECV1(suciU).
    % 5
    sum idsnS : idSN.(idsnS == 1) -> HN_SEND2(suciU,idsnS).
    % 9
    sum rH : R, autnH:AUTN,hxresH : HXRES, kseafH : KSEAF.
    HN_RECV5(rH,autnH,hxresH,kseafH).
    % 10
    UE_SEND6(autnH,rH).(
    % 13
    (sum resU : RES. UE_RECV9(resU).
    % 14
     HN_SEND10(resU,suciU).
    % 17
     sum supiUA : SUPI. HN_RECV13(supiUA)) +
    % 19
    (sum autsU : AUTS, syncError : Bool.UE_RECV15(autsU,syncError).
    % 20
    HN_SEND16(autsU,syncError,rH,suciU).
    % 22
    HN_RECV23) +
    % 24
    (sum authError: Bool.UE_RECV19(authError)
    % 25
    .HN_SEND20)).SN;
```

```
HN =
    % 6
    sum suciU:SUCI, idsnS:idSN. SN_RECV3(suciU,idsnS).
    % 7
    sum sqnH:SQN, kH : K, rH : R. (sqnH==1 && kH==1 && rH==1)->
    % 8
    SN_SEND4(rH,Autn(Conc(sqnH,Ak(kH,rH)),Mac(kH,sqnH,rH)),Hxres(rH,Res(kH,rH,idsnS)),Kseaf(kH,rH,s
    qnH,idsnS))
    % 15
    .((sum resU:RES, suciUA:SUCI. SN_RECV11(resU,suciUA).
    % 16
    (resU == Res(kH,rH,idsnS)) -> sum supiUA : SUPI.(supiUA == SUCI_GET_SUPI(suciUA))->
     SN_SEND12(supiUA)) +
    % 21
    (sum autsU : AUTS, syncError : Bool,rU : R,suciUF : SUCI.SN_RECV17(autsU,syncError,rU,suciUF).
    ((AUTS_GET_MAC(autsU) == Mac(kH,sqnH,rH))-> SN_SEND22 ))+
    % 26
    SN_RECV21).HN;
```

B.1.7 System initialisation

The initialisation of the system is done with *allow* function. This function has two parameters (i) Set of allowed actions, and (ii) Function *comm*. The function *comm* itself has two parameters (i) Synchronized actions (ii) Parallel processes. The initialisation allow is depicted in FigureB.1

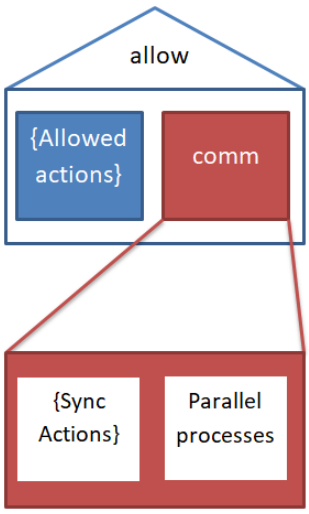

Figure B.1: Simple illustration of Allow function

B.1.7.1 Allowed actions

Actions allowed are the actions that will be considered in validation. Generally, allowed actions are the actions that represent names of synchronised actions. The actions allowed are:

```
{get_ue_info,ue_info_transfer,create_challenge,receive_challenge,send_results,
    results_transfer,authentication_succeed,authentication_syncError,transfer_syncError
    ,authentication_failed,transfer_authError,update_sqn}
```

B.1.7.2 Synchronized actions

It can be provided using multiaction operator '|'. The synchronized actions are:

```
{SN_SEND0 | UE_RECV1 -> get_ue_info,
        HN_SEND2 | SN_RECV3 -> ue_info_transfer,
        SN_SEND4 | HN_RECV5 -> create_challenge,
        UE_SEND6 | SN_RECV7 -> receive_challenge,
        SN_SEND8 | UE_RECV9 -> send_results,
        HN_SEND10 | SN_RECV11 -> results_transfer,
        SN_SEND12 | HN_RECV13 -> authentication_succeed,
        SN_SEND14 | UE_RECV15 -> authentication_syncError,
        HN_SEND16 | SN_RECV17 -> transfer_syncError,
        SN_SEND18 | UE_RECV19 -> authentication_failed,
        HN_SEND20 | SN_RECV21 -> transfer_authError,
        SN_SEND22 | HN_RECV23 -> update_sqn}
```

B.1.7.3 Initial parallel processes

The last step in this creation is system processes initiation, sometimes processes needs parameters in their expression but in our case no parameters needed. The parallel processes initialisation is:

```
        UE || SN|| HN
```

The model is provided via this link [20]

B.2 Execution of model in mCRL2 GUI

The execution of our model (full model and properties can be found in [20]) is depicted in three steps:

- Loading the project into mCRL2 graphical interface.

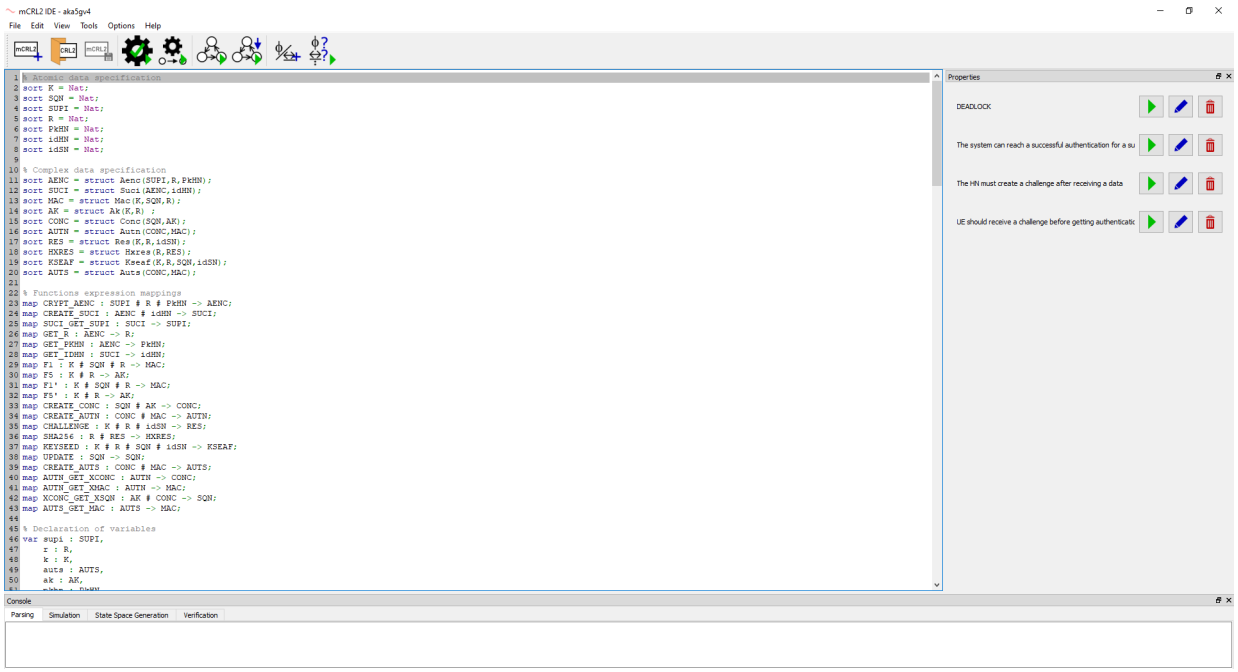

Figure B.2: mCRL2 graphical interface.

- Launch the parsing through parsing button and check results in the console below.

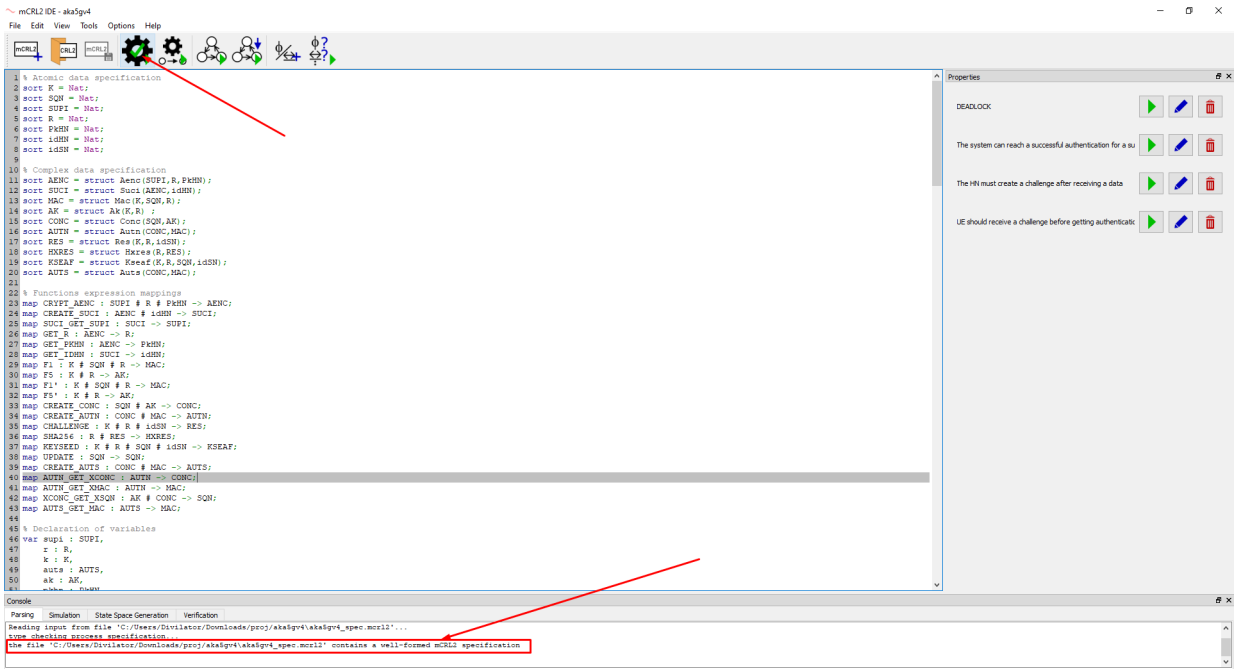

Figure B.3: mCRL2 project parsing

- Check the properties provided where green means satisfied.

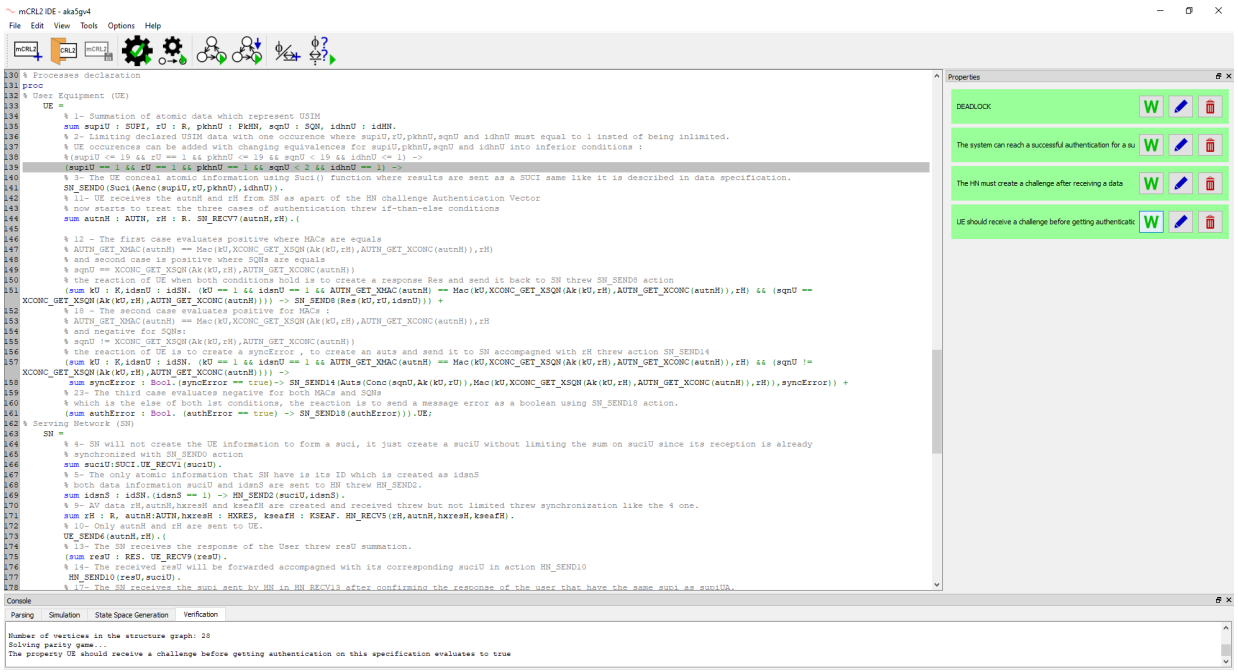

Figure B.4: mCRL2 properties validation

Bibliography

[1] R. Chandra Bhushan and D. K. Yadav, "Modeling and Formally Verifying a Safety-Critical System Through MCRL2," 2018 8th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 2018, pp. 775-779, doi: 10.1109/CONFLUENCE.2018.8442999.

[2] I. Ahmad, T. Kumar, M. Liyanage, J. Okwuibe, M. Ylianttila and A. Gurtov, "5G security: Analysis of threats and solutions," 2017 IEEE Conference on Standards for Communications and Networking (CSCN), 2017, pp. 193-199, doi: 10.1109/CSCN.2017.8088621.

[3] I. Ahmad, T. Kumar, M. Liyanage, J. Okwuibe, M. Ylianttila and A. Gurtov, "Overview of 5G Security Challenges and Solutions," in IEEE Communications Standards Magazine, vol. 2, no. 1, pp. 36-43, MARCH 2018, doi: 10.1109/MCOMSTD.2018.1700063.

[4] G. Liu, Y. Huang, Z. Chen, L. Liu, Q. Wang and N. Li, "5G Deployment: Standalone vs. Non-Standalone from the Operator Perspective," in IEEE Communications Magazine, vol. 58, no. 11, pp. 83-89, November 2020, doi: 10.1109/MCOM.001.2000230.

[5] 3GPP. 2018. Security architecture and procedures for 5G system. TS 33.501, v15.1.0.

[6] 5G NG-RAN, Xn general aspects and principles TS 38.420, v15.0.0.

[7] A. Koutsos, "The 5G-AKA Authentication Protocol Privacy," in 2019 IEEE European Symposium on Security and Privacy (EuroS&P), Stockholm, Sweden, 2019 pp. 464-479.

[8] J.A. Bergstra, J.W. Klop,Algebra of communicating processes with abstraction,Theoretical Computer Science,Volume 37,1985,Pages 77-121,ISSN 0304-3975, https://doi.org/10.1016/0304-3975(85)90088-X.

[9] Official Website of CPN tools `https://cpntools.org` Last Visited: 03-07-2021.

[10] Jan Friso Groote, Aad Mathijssen, Michel Reniers, Yaroslav Usenko, & Muck van Weerdenburg (2007). The Formal Specification Language mCRL2. In Methods for Modelling Software Systems (MMOSS). Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

[11] E. K. K. Edris, M. Aiash and J. K. Loo, "Formal Verification and Analysis of Primary Authentication based on 5G-AKA Protocol," 2020 Seventh International Conference on Software Defined Systems (SDS), 2020, pp. 256-261, doi: 10.1109/SDS49854.2020.9143899.

[12] mCRL2 Official Website, https://www.mcrl2.org/web/user_manual. Last visited 01/06/2021.

[13] Basin, D., Dreier, J., Hirschi, L., Radomirović, S., Sasse, R., & Stettler, V. (2018). A Formal Analysis of 5G Authentication. In 25th ACM Conference on Computer and Communications Security (Vol. 2018, pp. 1383-1396). [745] Association for Computing Machinery (ACM). https://doi.org/10.1145/3243734.3243846

[14] I. Alawe, Y. Hadjadj-Aoul, A. Ksentini, P. Bertin and D. Darche, "On the scalability of 5G core network: The AMF case," 2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC), 2018, pp. 1-6, doi: 10.1109/CCNC.2018.8319194.

[15] Bunte, Tim A. C.. "The mCRL2 Toolset for Analysing Concurrent Systems." Tools and Algorithms for the Construction and Analysis of Systems. Springer International Publishing,

[16] Z. Chen, S. Guo, R. Duan and S. Wang, "Security Analysis on Mutual Authentication against Man-in-the-Middle Attack," 2009 First International Conference on Information Science and Engineering, 2009, pp. 1855-1858, doi: 10.1109/ICISE.2009.1051.

[17] F. P. M. Stappers, M. A. Reniers, S. Weber and J. F. Groote, "Dogfooding the Formal Semantics of mCRL2," 2012 35th Annual IEEE Software Engineering Workshop, 2012, pp. 90-99, doi: 10.1109/SEW.2012.16.

[18] Service requirements for the 5G system.

[19] A. P. L. Ferreira, "Model Checking," 2011 Workshop-School on Theoretical Computer Science, 2011, pp. 9-14, doi: 10.1109/WEIT.2011.34.

[20] Our 5G-AKA model using mCRL2 language. `https://drive.google.com/file/d/1BLlgw2NVGxivbr-9GJTKI8Xaw4z_nrrs/view?usp=sharing`

[21] Choi, Jiyoung and Tsourdos, Antonios. (2011). Verification of Decision Making Behaviour for Heterogeneous Multi-Agent System. AIAA Guidance, Navigation, and Control Conference 2011. 10.2514/6.2011-6239.

[22] Lamsweerde, A. 2000. Formal Specification: A Roadmap. In Proceedings of the Conference on The Future of Software Engineering (pp. 147–159). Association for Computing Machinery.

[23] Arias Almeida, Jaime. (2013). Model Checking for TCC Calculus.

[24] Spin official website `http://spinroot.com/spin/whatispin.html` Last visited - 06-25-2021

[25] Official website of UPPAAL. `https://uppaal.org` Last visited : 25-06-2021

[26] Jim Woodcock, Peter Gorm Larsen, Juan Bicarregui, and John Fitzgerald. 2009. Formal methods: Practice and experience. ACM Comput. Surv. 41, 4, Article 19 (October 2009), 36 pages. DOI:https://doi.org/10.1145/1592434.1592436

[27] Laid Kahloul, Allaoua Chaoui, and Karim Djouani 2015. Modelling and Analysis of Mobile Computing Systems: An Extended Petri Nets Formalism. INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL, 10(2), p.211–221.

[28] Official mCRL2 website page mu-calculus `https://www.mcrl2.org/web/user_manual/language_reference/mucalc.html` Last Visited: 03-07-2021.

[29] Groote, J.F., Willemse, T.A.C.: Parameterised boolean equation systems. Theoret. Comput. Sci. 343(3), 332–369 (2005)

[30] Clarke, E. M.; Emerson, E. A.; Sistla, A. P. (1986), "Automatic verification of finite-state concurrent systems using temporal logic specifications", ACM Transactions on Programming Languages and Systems, 8 (2): 244, doi:10.1145/5397.5399

[31] Allen Emerson, E.; Clarke, Edmund M. (1980), "Characterizing correctness properties of parallel programs using fixpoints", Automata, Languages and Programming, Lecture Notes in Computer Science, 85: 169–181, doi:10.1007/3-540-10003-2_69, ISBN 978-3-540-10003-4

[32] Edmund M. Clarke, E. Allen Emerson: "Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic". Logic of Programs 1981: 52-71.

[33] J. Choi 2013. Model checking for decision making behaviour of heterogeneous multi-agent autonomous system.

[34] Astrauskas, V., Müller, P., Poli, F., and Summers, A. 2019. Leveraging Rust Types for Modular Specification and Verification. Proc. ACM Program. Lang., 3(OOPSLA).

[35] Official website of ProVerif tool `https://prosecco.gforge.inria.fr/personal/bblanche/proverif/` Last Visited: 06/28/2021

[36] Kahloul, Y. 2010. Modeling and Verification of RBAC Security Policies Using Colored Petri Nets and CPN-Tool. In Networked Digital Technologies (pp. 604–618). Springer Berlin Heidelberg.

[37] Zohra H, Kahloul L, Benharzallah S (2020) Using priced timed automata for the specification and verification of CSMA/CA in WSNs. Int J Inf Commun Technol 17(2):129–145

[38] R. Bettira, L. Kahloul, M. Khalgui and Z. Li, "Reconfigurable Hierarchical Timed Automata: Modeling and Stochastic Verification," 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), 2019, pp. 2364-2371, doi: 10.1109/SMC.2019.8913890.

[39] Laid Kahloul, and Messaouda Grira 2014. Formal Specification and Verification of Mobile Agent Systems. INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL, 9(3), p.292–304.

[40] Laid Kahloul 2015. Wireless Sensor Networks for the Surveillance of Wide Date Palm Forests: (Specification and Verification Levels). Procedia Computer Science, 56, p.598-603.

[41] M. Houimli, L. Kahloul and S. Benaoun, "Formal specification, verification and evaluation of the MQTT protocol in the Internet of Things," 2017 International Conference on Mathematics and Information Technology (ICMIT), 2017, pp. 214-221, doi: 10.1109/MATHIT.2017.8259720.

[42] L. Kahloul, A. Chaoui and K. Djouani, "Modeling and Analysis of Reconfigurable Systems Using Flexible Petri Nets," 2010 4th IEEE International Symposium on Theoretical Aspects of Software Engineering, 2010, pp. 107-116, doi: 10.1109/TASE.2010.28.

[43] M. Ramdani, L. Kahloul, M. Khalgui, Z. Li and M. Zhou, "RCTL: New Temporal Logic for Improved Formal Verification

[44] Official website of scyther verification tool `https://people.cispa.io/cas.cremers/scyther/` Last Visited: 06/28/2021

[45] J. Zhang, Q. Wang, L. Yang and T. Feng, "Formal Verification of 5G-EAP-TLS Authentication Protocol," 2019 IEEE Fourth International Conference on Data Science in Cyberspace (DSC), 2019, pp. 503-509, doi: 10.1109/DSC.2019.00082.

[46] A. Koutsos, "The 5G-AKA Authentication Protocol Privacy," 2019 IEEE European Symposium on Security and Privacy (EuroS&P), 2019, pp. 464-479, doi: 10.1109/EuroSP.2019.00041.

[47] Website source including Figure 1.5 `https://www.metaswitch.com/knowledge-center/reference/what-is-the-5g-session-management-function-smf` Last Visisted - 29-06-2021.

[48] Cuijpers, PJL 2004, 'Hybrid process algebra', Doctor of Philosophy, Mathematics and Computer Science, Eindhoven. https://doi.org/10.6100/IR582613

[49] Clarke, Edmund M., Jr; Grumberg, Orna; Peled, Doron (December 1999). Model Checking. Cyber Physical Systems Series. MIT Press. p. 14. ISBN 9780262032704.

[50] Groote, Jan & Mathijssen, Aad & Reniers, Michel & Usenko, Yaroslav & Weerdenburg, Muck. (2006). The Formal Specification Language mCRL2.

[51] Stappers, F., Reniers, M., Weber, S., & Groote, J. (2012). Dogfooding the Formal Semantics of mCRL2. In 2012 35th Annual IEEE Software Engineering Workshop (pp. 90-99).

[52] Cas Cremers, & Martin Dehnel-Wild (2019). Component-Based Formal Analysis of 5G-AKA: Channel Assumptions and Session Confusion. In NDSS. The Internet Society.