Security for Network Services Delivery of 5G enabled Device-to-Device Communications Mobile Network

A thesis submitted to Middlesex University in partial fulfilment of the requirements for the degree of Doctor of Philosophy (Ph.D.)

By

Ed Kamya Kiyemba Edris

Director of Studies:	**Dr. Mahdi Aiash**
Supervisor:	**Prof. Jonathan Loo**

School of Science and Technology
Department of Computer Science
Middlesex University

February 2021

Declaration of Authorship

I, **Ed Kamya Kiyemba Edris**, declare that this thesis titled, "Security for Network Services Delivery of 5G enabled Device-to-Device Communications Mobile Network" and the work presented in it are my own. I confirm that:

- This work was done wholly for a degree at Middlesex University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at Middlesex University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. Except for such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Abstract

The increase in mobile traffic led to the development of Fifth Generation (5G) mobile network. 5G will provide Ultra Reliable Low Latency Communication (URLLC), Massive Machine Type Communication (mMTC), enhanced Mobile Broadband (eMBB). Device-to-Device (D2D) communications will be used as the underlaying technology to offload traffic from 5G Core Network (5GC) and push content closer to User Equipment (UE). It will be supported by a variety of Network Service (NS) such as Content-Centric Networking (CCN) that will provide access to other services and deliver content-based services. However, this raises new security and delivery challenges. Therefore, research was conducted to address the security issues in delivering NS in 5G enabled D2D communications network.

To support D2D communications in 5G, this thesis introduces a Network Services Delivery (NSD) framework defining an integrated system model. It incorporates Cloud Radio Access Network (C-RAN) architecture, D2D communications, and CCN to support 5G's objectives in Home Network (HN), roaming, and proximity scenarios. The research explores the security of 5G enabled D2D communications by conducting a comprehensive investigation on security threats. It analyses threats using Dolev Yao (DY) threat model and evaluates security requirements using a systematic approach based on X.805 security framework. Which aligns security requirements with network connectivity, service delivery, and sharing between entities.

This analysis highlights the need for security mechanisms to provide security to NSD in an integrated system, to specify these security mechanisms, a security framework to address the security challenges at different levels of the system model is introduced. To align suitable security mechanisms, the research defines underlying security protocols to provide security at the network, service, and D2D levels. This research also explores 5G authentication protocols specified by the Third Generation Partnership Project (3GPP) for securing communication between UE and HN, checks the security guarantees of two 3GPP specified protocols, 5G-Authentication and Key Agreement (AKA) and 5G Extensive Authentication Protocol (EAP)-AKA' that provide primary authentication at Network Access Security (NAC).

The research addresses Service Level Security (SLS) by proposing Federated Identity Management (FIdM) model to integrate federated security in 5G, it also proposes three security protocols to provide secondary authentication and authorization of UE to Service Provider (SP). It also addresses D2D Service Security (DDS) by proposing two security protocols that secure the caching and sharing of services between two UEs in different D2D communications scenarios. All protocols in this research are verified for functional correctness and security guarantees using a formal method approach and semi-automated protocol verifier.

The research conducts security properties and performance evaluation of the protocols for their effectiveness. It also presents how each proposed protocol provides an interface for an integrated, comprehensive security solution to secure communications for NSD in a 5G enabled D2D communications network. The main contributions of this research are the design and formal verification of security protocols. Performance evaluation is supplementary.

You'll Never Walk Alone.

"Are those who have knowledge and those who have no knowledge alike? Only the men of understanding are mindful. " (Quran, 39:9)

Acknowledgements

The completion of this research would not have been possible without the support of many great inspiring people. It is an honour to convey my gratitude to them.

First and foremost, I would like to express my gratitude to Dr. Mahdi Aiash for his conscientious supervision, advice, and guidance not only during this research but from the very beginning of my academic journey. His early supervision, encouragement, and support at the undergraduate level motivated me to keep on pushing for more academic milestones. His brilliance as a scientist and researcher was instrumental in my desire to follow this path. I also want to thank him for giving me teaching opportunities, I will always be indebted to him.

I would also like to thank Prof. Jonathan Loo for his supervision and advice. His originality and expertise helped in the development of the main objective of this research. His desire to guide me and always trying to make me think as an accomplished researcher even when I was not, fuelled my desire of becoming one. I will always be grateful for his efforts.

I dedicate this thesis to the late Kiyemba's, late Nantongo's, Ssebuliba's, Bukenya's, Mubondwa's and late Chabo's families. To my big brother, I have no words to express my gratitude for all you have done for me. You were always there for me, no matter what situation am in or where I am. I owe you more than you will ever know.

To my partner, I have no words to express my gratitude for the support, dedication, love, and constant confidence you put in me, which pushed me to walk this journey with ease even in difficult times. I would like to thank my children for being the light that always shines around me and always reminding me to sleep. I have renewed my purpose in life.

Finally, I would like to convey my gratitude to family and friends for their endless support and love, your contribution during this journey is unmeasurable.

List of Publications

Most of the theories, technical discussions and contributions in this thesis have led to the following publications:

Journal Papers

1. Edris, E. K. K., Aiash, M., Loo, J. K. and Alhakeem, M. S. (2021) 'Formal Verification of Secondary Authentication Protocol for 5G Secondary Authentication', International Journal of Security and Networks, 2021 Vol.16 No.4, pp.223 - 234. DOI: 10.1504/IJSN.2021.119379.

2. Edris EKK, Aiash M, and Loo J. (2020), 'Formal Verification of Authentication and Service Authorization Protocols in 5G enabled Device-to-Device Communications using ProVerif', Electronics 2021, 10, 1608. 10.3390/electronics10131608.

3. Edris EKK, Aiash M, and Loo J. (2021), 'DCSS Protocol for Data Caching and Sharing Security in 5G Network, Network 2021, 1, 75-94. 10.3390/network1020006.

4. Edris EKK, Aiash M, and Loo J. (2021), 'Performance Evaluation using Applied Pi Calculus and Markov Chain based on 5G Security Protocols', International Journal of Communication Networks and Distributed Systems (Submitted February, 2021).

5. Edris EKK, Aiash M, and Loo J. (2021), 'Security Framework for Network Services Delivery in 5G enabled D2D Communications., International Journal of Security and Networks (Submitted February, 2021).

Conference Papers

1. E. K. K. Edris, M. Aiash and J. Loo, "Investigating Network Services Abstraction in 5G Enabled Device-to-Device (D2D) Communications," 2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), Leicester, United Kingdom, 2019, pp. 1660-1665, doi: 10.1109/SmartWorld-UIC-ATC-SCALCOM-IOP-SCI.2019.00296.

2. E. K. Kiyemba Edris, M. Aiash and J. K. Loo, "The Case for Federated Identity Management in 5G Communications," 2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC), Paris, France, 2020, pp. 120-127, doi: 10.1109/FMEC49853.2020.9144855.

3. E. K. Kiyemba Edris, M. Aiash and J. K. Loo, "Network Service Federated Identity (NS-FId) Protocol for Service Authorization in 5G Network," 2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC), Paris, France, 2020, pp. 128-135, doi: 10.1109/FMEC49853.2020.9144706.

4. 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), Paris, France, 2020, pp. 256-261, doi: 10.1109/SDS49854.2020.9143899.

Book Chapter

1. Edris, E. K. K., Aiash, M., and Loo, J. K. (2021) 'Security in Network Services Delivery for 5G enabled D2D Communications: Challenges and Solutions', Challenges in the IoT and Smart Environments - A Practitioners' Guide to Security, Ethics and Criminal Threats. Switzerland: Springer, pp, doi: 10.1007/978-3-030-87166-6.

Contents

List of Figures

List of Tables

Abbreviations and Acronyms

The abbreviations that will be used throughout this thesis are listed here; all other abbreviations are introduced when they are first used.

3GPP Third Generation Partnership Project. ii, 1

5G Fifth Generation. ii, 1

5GC 5G Core Network. ii, 1

AAA Authentication Authorization Accountability. 54

ABAC Attribute-based Access Control. 70

ACL Access Control List. 4

AKA Authentication and Key Agreement. ii, 3

AMF Access and Mobility Access Function. 10

ARPF Authentication Credential Repository. 14

AUSF Authentication Server Function. 14

AV Authentication Vector. 10

BBU Baseband Unit. 9

C-RAN Cloud Radio Access Network. ii, 9

CBAC Capability-based Access Control. 71

CCN Content-Centric Networking. ii, 2

CDN Content Delivery Network. 2

CK Cipher Key. 75

CTMC Continuous Time Markov Chains. 158

D2D Device-to-Device. ii, 1

DCSS Data Caching and Data Sharing Security. 6

DDACap D2D Attributes and Capabilities. 7

DDS D2D Service Security. ii

DDSec D2D Sharing Security. 7

DN Data Network. 4

DoS Denial of Services. 21

E2E End-to-End. 17

EAP Extensive Authentication Protocol. ii, 3

EBAC Encryption-based Access Control. 70

ECC Elliptic Curve Cryptography. 15

ECIES Elliptic Curve Integrated Encryption Scheme. 14

FId Federated Identity. 71

FIdM Federated Identity Management. ii, 6

gNB 5G Node. 14

GPSI Generic Public Subscription Identifier. 45

HN Home Network. ii, 10

IAM Identity and Access Management. 109

IBC Identity Based Cryptography. 15

ICN Information-Centric Networking. 2

ID Identity. 13

IdP Identity Provider. 110

IK Integrity Key. 75

IoT Internet of Things. 1

KDF Key Derivation Function. 75

LTE Long Term Evolution. 1

MAC Message Authentication Code. 10

MNO Mobile Network Operator. 2

NAC Network Access Security. ii

NDO Named Data Object. 22

NF Network Function. 10

NFV Network Function Virtualization. 11

ng-RAN New Generation Radio Access Network. 10

NS Network Service. ii, 2

NS-FId Network Service Federated Identity. 6

NSD Network Services Delivery. ii, 2

NSS Network Service Security. 6

PDU Protocol Data Unit. 10

PKI Public key infrastructure. 15

ProSe Proximity Services. 1

QoE Quality of Experience. 1

QoS Quality of Service. 12

RRH Remote Radio Head. 10

SAP Secondary Authentication Protocol. 6

SBA Service-Based Architecture. 12

SEAF SEcurity Anchor Function. 14

SLS Service Level Security. ii

SMF Session Management Function. 10

SN Serving Network. 10

SNN Serving Network Name. 74

SP Service Provider. ii, 4

SQN Sequence Number. 10

SS Service Server. 67

SSO Single Sign On. 7

SUCI Subscriber Concealing Identifier. 14

SUPI Subscriber Permanent Identifier. 4

UE User Equipment. ii, 1

USIM Universal Subscriber Identity Module. 13

V2V Vehicle to Vehicle. 1

XOR eXclusiveOR. 14

Chapter 1

Introduction

Mobile network technologies have evolved over the years with an immense growth of mobile usage and wireless data traffic surpassing cellular spectrum capacity. Which was no longer adequate for future mobile network traffic. This led to the development of the Fifth Generation (5G) mobile network, to enable new network and service functions to support mission-critical services such as public safety, eHealth, transport, and industrial automation. 5G will also create new use cases and connect vertical industries such as tactile intent, Internet of Things (IoT) and Vehicle to Vehicle (V2V) communication. Additionally, it will improve the Quality of Experience (QoE) for users by providing very high availability, Ultra-Reliable Low Latency Communications (URLLC), massive Machine Type Communications (mMTC), and enhanced Mobile Broadband (eMBB) (Liu & Yu 2018).

The development of 5G was intended to solve future constraints of service access using a high data rate to offload network traffic of Peer to Peer (P2P) links such as Device-to-Device (D2D) communications. By relying on User Equipment (UE) to offload the traffic burden from Base Station (BS) and 5G core network (5GC). With 5G, systems performance and network capacity will be improved, and mobile applications magnified extensively. D2D communication is one of the technologies supporting 5G in enabling new applications and service delivery. Initially, D2D communication was intended for allowing mobile communication multi-hop relays then later enhanced to improve cellular network delay and spectrum efficiency (3GPP 2014). D2D communication was introduced in Long Term Evolution (LTE) as a service by Third Generation Partnership Project (3GPP) to allow UEs to communicate to each other, enabling users to easily access Proximity Services (ProSe) such as streaming, content sharing, and gaming. As the standardization of D2D communication in 5G is still to be finalized, it has been proposed that D2D communication will support priority control of Mission Critical Push to Talk (MCPTT), Augmented Reality (AR), smart city, industry automation, emergency services, V2V, and location-based services. Therefore, there is a need for integrated service delivery and security frameworks that can support 5G enabled D2D communications.

1.1 Background

The increase in mobile traffic is due to the high demand of Over the Top (OTT) applications such as social media, live-streaming, local based advertising, and the popularity of smartphones usage. Mobile traffic is anticipated to keep growing gradually, most of the data traffic will be from mobile devices, Machine to Machine (M2M) communication (Cisco 2017), and core network gateway systems data such as servers (Wang & Yan 2017). 5G development was inspired by the need for ultra-low latency and very high-reliability network to support service delivery and demand of quick access to content data by end users and Mobile Network Operator (MNO). Content distribution and retrieval will dominate mobile activities, however, delivering such services to the end user efficiently and securely is a big challenge. Mobile capacity has been affected by the growing demand for multimedia content, high availability, and better performance by the end users. Whereby network infrastructures are being overloaded and have become highly inefficient for content distribution. End users will be involved in content-based operations through D2D communications, which will also be used as an underlay technology for 5G to offload traffic from the backhaul of 5GC by pushing content to edge closer to users (Gupta & Jha 2015).

D2D communications can utilize cellular resources and offload network traffic, by communicating directly without conveying content through a BS, and UE can act as data consumer as well as playing a role in content distribution and delivery (Chandrasekaran et al. 2018). In 5G, D2D communications will support new use cases and services, while the delivery of these services to the end user will be facilitated by Network Services (NS) and context-aware enabled devices. To be able to distribute and deliver content to the end user, various content delivery models such as Content Delivery Network (CDN) are used. CDN can be deployed at the edge, BS, and Access Point (AP) to support cache servers and with D2D communication, UEs can also be used as cache nodes (Golrezaei et al. 2013). In addition, content caching could also be used to support content delivery and backhaul traffic offloading, as most of the wireless traffic generated is from downloads of popular content replicated in multiple locations (Liu et al. 2016). The introduction of CDN and content caching in a mobile network can be integrated with Information-Centric Networking (ICN) (Liang 2015), transforming the network from connection-centric to information-centric such as Content-Centric Networking (CCN) (Jacobson 2009).

With all its benefits, D2D communications in 5G introduces challenges for secure content delivery, sharing, and traffic offloading. The related work on mobile network has been focussing mainly on network architectures, caching and D2D content delivery solutions (Chandrasekaran et al. 2018). Some studies have investigated security in ICN (Tourani et al. 2018), (AbdAllah et al. 2015) and in D2D communications (Haus et al. 2017), (Gandotra et al. 2017). However, security issues in an integrated system for NS in D2D communications have not been investigated extensively. Furthermore, related work lacks a systematic and abstracted approach to security evaluation. Moreover, investigation on CCN security has not been considered in Heterogeneous Networks (HetNets) like 5G. So far, less attention has been given to the security issues faced by an integrated service model for next generation mobile networks. Therefore, there is a need for a system model that can be used to develop security frameworks for 5G enabled D2D content-centric model. There is also the lack of comprehensive investigation into the security threats and requirements of Network Services Delivery (NSD) in 5G enabled D2D communications. The provision of secure NSD in D2D communications is paramount to achieving the main objectives of 5G.

The research efforts on NS so far have been mainly on how they can be expanded and delivered to end user. Even though the expansion of mobile NS enriches the telecommunications ecosystem and improves QoS and QoE but also brings different requirements and challenges to mobile network security. Therefore, there is a need for a security framework to secure the delivery of NS.

1.2 The Objectives and Scope of the Research

1.2.1 Research Objectives

The research aim is to develop an integrated hybrid security solution that is host and information-centric for NSD in 5G enabled D2D communications network to provide secure communication, protection of entities, and information. The research studied NSD, 5G, and D2D communications security points of view. The objectives and the scope are as follows:

- Explore UE communication and content delivery in D2D communications and CCN in 5G.
- Investigate the security issues affecting NSD in 5G enabled D2D communications extensively.
- Examine the security threats and requirements of NS such as content delivery and sharing between D2D devices in proximity, in coverage, out-of-coverage, and roaming state scenarios.
- Evaluate the security requirements of 5G enabled D2D communications network.
- Propose security mechanisms, design security protocols to address the security concerns for NSD in 5G enabled D2D communications.
- Specify and verify the designed security protocols using analytical and formal analysis.
- Evaluate the security properties and performance of the proposed security protocols.

The main purpose of this research is to introduce a security framework, develop security mechanisms and protocols to address security and privacy issues of NSD in 5G enabled D2D communications network. This research will prove that the proposed security framework can be applied in the provisioning of security and privacy in 5G enabled D2D network.

1.2.2 The Scope of the Thesis

The main concern of the research in this thesis is proposing security mechanisms that specifically consider the security of NSD in 5G enabled D2D communications network. It is not concerned with the application-level security, the focus is on infrastructure and service level security as explained in security evaluation of the system model in chapter 4. In addition, it led to the designing and analysing of security mechanisms using an integrated security framework. The work in this research also focuses on secondary authentication and service authorization methods on the service level as the primary authentication has been studied extensively. However, we adopt the network level's primary authentication methods i.e., 5G-Authentication and Key Agreement (AKA) and 5G Extensible Authentication Protocol (EAP)-AKA' protocols which are introduced and formally analysed to evaluate their security properties as specified by 3GPP. The author believes that primary authentication vulnerabilities could compromise secondary authentication protocols hence the need for formal verification. The security solutions proposed in this thesis address security and privacy in 5G enabled D2D communications, the author also believes that the proposed security mechanisms could be used in the next generation mobile network. Formal methods and automated protocol verifier tool are used to verify 3GPP specified protocols and our proposed security protocols to develop and test these protocols. The verification tool only considers the protocol as a process within a system as per the definition in the protocol description. Attacks found because of implementation environment are not considered as discussed in chapter 3.

The experimental implementation is not the focus of this thesis, though security properties analysis and performance evaluation of the protocols are conducted to show the effectiveness of the protocols' overall performance. Future work includes the implementation of the proposed security protocols in other systems, by converting the protocol code from Objective Caml to a native programming language like C or Java. In addition, the extension of the security framework to cover handover and network slicing in mobile network.

1.3 Problem Definition

NSD in 5G enabled D2D communications network faces new and old security challenges. The security of NS such as CCN in mobile network has not been investigated and their secure provisioning is essential to 5G's main objectives. As 5G promises to push content closer to the end user and while MNOs want to offload backhaul traffic to the edge (Gupta & Jha 2015), D2D communications will be used for this purpose (Chandrasekaran et al. 2018), (Golrezaei et al. 2013).

In addition, UE will have to access NS securely, but since 5G standardization is not finalized yet, there is a lack of an enhanced and integrated security approach to address the security concerns in 5G enabled D2D Communications. The security for such a network is complicated as it comprises multi-layered architecture, multiple wireless access, and services provided by MNO and third-party Service Providers (SP). Due to the characteristics of 5G, the mobile network is no longer monolithic and a one-suits-all type of security solution will be insufficient (5GPPP 2017). Not only the communication channels and network entities but also data and services security must be addressed. That is the same to the user's data which must be protected from any form of attack or exposure. Any privacy solution should protect the UE's Subscriber Permanent Identifier (SUPI) against active attacks. Some basic security mechanisms will apply by design, but a new security framework is required to adapt to the new challenges.

The integration of CCN with mobile network turns the network model into both host and information-centric networking (Chandrasekaran et al. 2018). Whereby D2D communication is used as the underlaying communication and CCN as content delivery mechanism (Ravindran 2019), (Zhang et al. 2019). Therefore, the use of information-centric in a mobile network arises some conflicts such as domain and layer security, host verification, service validity, content identification, trust establishment, UE data caching, and Access Control List (ACL)s. Due to the nature of our model, the security focus is on host to host and the content itself. To address the security concerns on both infrastructure and service levels, a hybrid security approach using both host-centric and information techniques is needed. The research also must consider service-driven limitations on the security architecture which require the optional use of other security measures such as extra security entities.

3GPP addresses security issues on UE, SUPI privacy, integrity protection, radio interface, and network slicing security (3GPP 2020*f*), (5GPPP 2017). However, there is a need to put in place security measures to protect the UE while accessing NS, this could be addressed through a multi-level security framework. Furthermore, D2D will need to be able to share data content securely without the need to involve a central authority such as HN or Data Network (DN). Most existing literature addresses security per layer and a few with multi-level solutions (Lichtman et al. 2018). Therefore, an integrated hybrid security framework will be used to address security in our system model using a modular architecture. These security issues are discussed in detail in section 4.3.

1.3.1 Research Question

The research in this thesis explores 5G security and investigates the security in 5G enabled D2D communications network. This research aims to address the security challenges in NSD and content sharing between UEs, this led to the main question that needed to be answered.
'How to provide secure communications for use cases in 5G enabled D2D communications network?' The solution to the above-mentioned issues required a new security architecture, development of security mechanisms and protocols. In addition, the examination of security design and functions of legacy generations to help in the investigation of threats and vulnerabilities in 5G and NS to support services delivery to the end user. While the evaluation of the security requirements and design of security protocols enables us to achieve the research's main objectives. The main research question was formulated into the following questions:
'How to deliver NS in 5G enabled D2D communications network?'
This required the development of a system model and the introduction of a service framework to illustrate the delivery services to UE.
'What are the threats and vulnerabilities faced by a UE when accessing and delivering services in a 5G enabled D2D communications network?'
A specific security framework was required to answer this question to show the security vulnerabilities of 5G enabled D2D communications network.
'How to provide a secure communication for NSD in 5G enabled D2D communications?'
The answer to the question was the designing of new security protocols to provide a robust security solution that addresses the security and privacy of 5G enabled D2D communications.
How the proposed security protocols can interface to an integrated security mechanism to provide robust security for 5G enabled D2D communications network?'
The proposed approach intends to protect the hosts, data, and network from any attacks, making sure that NSD is secure at all levels, this is achieved by creating an interface between the proposed security protocols.

1.4 Research Methodology

The approach and methods for this study were proposed after identifying the constraints in different studied areas of this research. The literature review has explores related works to date, to enable the evaluation of the existing problems and limitations to present suitable solutions to address the defined problems with an intention of improving the systems' security. Based on the research objectives, the methodology is outlined in different phases. By analysing and concluding each phase, it gives motivation for the next phase, briefly explained as follows:

1. Phase one will present the related work literature view, by providing a solid background on 5G and D2D communications. The focus is to understand the architecture, security and privacy in mobile networks. The approach will start with exploring the security and privacy in both Fourth Generation (4G) and 5G. Analysing threats and vulnerabilities in D2D communications and evaluate existing security mechanisms. Investigating if the security threats and countermeasures in legacy systems can be inherited in 5G.

2. Phase two will study D2D communications and CCN. This part of the research is to study and understand the proposed NS concepts and how they can be applied in 5G. Investigating

NS like content delivery and sharing between D2D devices, in coverage, out-of-coverage, and roaming state.

3. Phase three will provide a threat model and evaluates the security requirements of NSD in 5G enabled D2D communications using the X.805 security framework. The information gathered in phases one and two will be used to investigate threats and vulnerabilities associated with the security and privacy of the system model. Then explore possible countermeasures by investigating the current solutions, attempting to address these security issues and find a gap in the study.
4. Phase four will propose security solutions, the information gathered from the literature review and security analysis will be used to develop protocols to provide security and privacy protection in 5G enabled D2D communications network.
5. Phase five will analyse and verify the developed security mechanisms and protocols, using formal methods and an automated tool. It will evaluate the security properties and performance of the protocols using simulation and analytical modelling techniques.
6. Phase six will make recommendations, reach conclusions, write research papers for publication, and then write the thesis.

Detailed methods and approaches used in this research are presented in chapter 3.

1.5 Main Contributions

In this thesis, security for NSD in 5G enabled D2D communications has been investigated by critically reviewing 5G system, security, and the related work. The research introduced a NS abstraction of 5G enabled D2D communications to align the NSs with the 5G protocol stack and internet protocol. This research lead to the introduction of a NSD framework, which can be used to retrieve and share services in 5G enabled D2D communications network. It provided a comprehensive, systematic, and modular approach of investigating security and privacy of NS using X.805 security framework (Zeltsan 2005) that identified security threats and vulnerabilities in 5G enabled D2D communications.

In order to address those security challenges, a novel approach was proposed that introduced a Network Service Security (NSS) framework with an integrated system model for a next generation mobile network. The security framework consisted of a network, service D2D levels, and underlying security mechanisms that provided authentication, authorization, and federated security at those levels. In order to provide a comprehensive security solution, 3GPP specified 5G-AKA and 5G-EAP'-AKA protocols were formally verified and analysed for security guarantees and effectiveness at the network-level.

The research introduced security mechanisms to address security at the service level, proposing a Federated Identity Management (FIdM) model for supporting the integration of federated security in 5G. Proposed Secondary Authentication Protocol (SAP)-AKA protocol to provide secondary authentication, Network Service Federated Identity (NS-FId) protocol to provide federated authentication and authorization of UE to SP, and Data Caching and Data Sharing Security (DCSS) protocol to authorize the UE to cache and share data. Furthermore, to addresses security at the

D2D level, D2D Sharing Security (DDSec) and D2D Attributes and Capabilities (DDACap) protocols are proposed to secure the caching and sharing of services between two UEs in network assisted and non-network assisted D2D communications scenarios, respectively.

All protocols in this research are verified for functional correctness and security guarantees using a formal methods approach and semi-automated protocol verifier tool and applied *pi* calculus (Blanchet et al. 2020). The research conducted security properties analysis based on two taxonomies (Lowe 1997), (Menezes et al. 2018). It also conducted performance evaluation of the protocols using analytical and simulation modelling for their effectiveness based on the Markov chain model (Stewart 1994) and network simulator (Nsnam 2021), respectively. In order to provide an integrated security solution for secure delivery of NSs in 5G enabled D2D communications, all protocols in this thesis interface with each other as they address the security requirements from one level to another.

1.6 Originality of the Intended Work

Different research studies have been carried out to solve the security problems in 5G enabled D2D communications by proposing various security solutions. Others tried to propose solutions for services delivery and integration of 5G and CCN to solve the problem of the high demand for multimedia content by end users through their UEs.

The problem is that their efforts lacked a systematic solution approach to the problems. We believe that addressing security in D2D communications requires an abstracted and layered solution to provide multi-level security. Not providing security on one level, might compromise security on another secure level, hence the need to address security on the network and service levels. The characteristics of 5G enabled D2D communications network make these efforts inadequate to solve the mentioned problem, we believe that HetNets like 5G requires an integrated solution that solves the host-centric and information-centric security problems with the same security mechanisms. Since 5G security standardization is still in progress, a robust security framework that can easily be implemented to future next generation architecture using seamless unified authentication and authorization procedures, supporting multiple SPs in different security domains is required. The author believes that 5G enabled D2D communications security framework requires another security level that deals with authentication and authorization procedures for service access and sharing between D2D users. This should be complemented by implementing federated authorization mechanisms.

Therefore, the research in this thesis addresses the inadequateness of the other works by proposing a novel solution approach that is integrated and multi-layered. It provides security and privacy to both data and the participating entities such as the UE, HN, and SPs at service and infrastructure levels. The framework provides security for the host and data being transmitted from the point where the connection is initiated to the point of accessing and sharing the data with other UEs. It addresses security at the network, service, and D2D levels. Furthermore, it provides a federated service authorization and Single Sign On (SSO). This was achieved by proposing security protocols that were verified and formalized using formal methods and an automated protocol verifier.

With security mechanisms deployed at different levels and integrated with a multi-layered security framework, this provides a unified, secured, and integrated solution for service delivery in a 5G enabled D2D communications network.

1.7 The Outline of the Thesis Structure

The thesis is structured as follows:

- **Chapter 2**: Studies the literature relevant to our research questions, by describing the related work on 5G system, security architecture, D2D communications, CCN, and general security concepts. It also explains the integration of CCN and mobile network.
- **Chapter 3**: Discusses the proposed approaches and methods used to address the research questions. This includes the threat, formal and analytical modelling approaches.
- **Chapter 4**: Introduces a service delivery framework for NS in 5G enabled D2D communications network. It also provides a security analysis of NSD using the X.805 security framework.
- **Chapter 5**: Introduces the NSS framework and the proposed security model used to develop the underlying security protocols.
- **Chapter 6**: Analyses the primary authentication methods used for authenticating the UE and the HN for network access. It evaluates 5G-AKA and 5G EAP-AKA' protocols, formally analyses and verifies these protocols.
- **Chapter 7**: Develops the SAP-AKA for authenticating UE and SP. It introduces the FIdM model, then develops a NS-FId protocol for service access authentication and authorization of the UE. It also develops a DCSS protocol for authorizing data caching and sharing to the UE.
- **Chapter 8**: Develops DDSec and DDACap protocols responsible for D2D secure caching and sharing of data in network assisted and non-network assisted D2D communications, respectively.
- **Chapter 9**: Evaluates the performance of the security protocols using Markov chain stochastic process and network simulator. Investigates the possible integration of the proposed security mechanisms.
- **Chapter 10**: Concludes the thesis by summarising this research's main contributions and future work direction.

Chapter 2

Overview of 5G and D2D Communications Networks

2.1 Introduction

5G is intended to connect end users faster, ubiquitously, and be more reliable than legacy systems. MNOs are adopting new technologies to provide service anywhere, reduce operational costs, improve spectral efficiency, maintain optimal performance by applying more advanced self-organizing functions. Additionally, 5G will be supported by cloud services to facilitate computational operations and virtualization implementation. To address the end user requirements, 5G will use technologies such as D2D communications, ultra-dense network, and CCN to improve edge services. Moreover, technologies such as the Multi-in Multi-out (MIMO) and beamforming techniques have been adopted to enable Millimetre Wave (mmWave) spectrum access to support ultra-dense high-capacity provisioning (Bogale & Le 2016). It will manage multiple cells and users together, deploy networks with multi-Radio Access Network (RAN) technologies to coexist with HetNets. It will also support new services such as Virtual Reality (VR), massive sensor communication as well as reducing signalling overhead and energy consumption to improve end user experience (Sharma et al. 2019). 5G promises to support faster transmission with ultra-low delay, connect more devices, manage huge volume of traffic, and support fast mobility of UEs (Parvez et al. 2018). Its objective is to provide higher data rates, higher availability, lower latency, better energy efficiency, and security than the legacy networks.

This is supported by various technologies and concepts, so this chapter gives an overview of the 5G system, security architectures, concepts, NS, D2D communications, and CCN. Some of the work in this chapter is also presented in (Edris et al. 2019).

2.2 Overview of 5G Network Architecture

2.2.1 Network Architecture

5G adopts the Cloud Radio Access Network (C-RAN) architecture (Checko et al. 2015), that centralizes the baseband resources to a single virtualized Baseband Unit (BBU) pool, connecting to

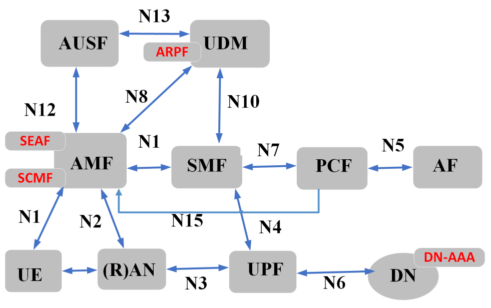


Figure 2.1: 5G System Architecture

radio transceiver units called Remote Radio Head (RRH) which are connected by optical fronthaul links. In addition, virtualization is enabled to facilitate network infrastructure sharing and the circuit interface between BBU and RRH has been changed to packet fronthaul. The BBU is divided into a distributed unit responsible for the physical layer and medium access control layer process and a centralized unit responsible for upper layer computations process (Zhang et al. 2019).

The mobile network still comprises three main entities:

- UE: A Mobile Equipment (ME) containing the USIM with cryptographic capabilities such as algorithms, symmetric and asymmetric encryptions, Message Authentication Code (MAC). It also stores SUPI, HN public key, long-term key K, and Sequence Number (SQN) counter.
- Home Network (HN): It houses the database, other security functions, generates Authentication Vector (AV), and stores user's subscription data. It also shares security context such as UE ID and long-term key K with the UE.
- Serving Network (SN): The access network to which the UE connects to. The SN and the HN are managed by different MNOs, usually connected over a secure channel using Internet Protocol Security (IPsec) or Transport Layer Security (TLS).

The UE will connect to New Generation Radio Access Network (ng-RAN) or Access and Mobility Access Function (AMF) to access the network. The 5GC consists of Network Function (NF) such as Session Management Function (SMF), User Plane Function (UPF), Policy Control Function (PCF) and Application Function (AF) as shown in Fig. 2.1 and $N1.2.3.4...$ are reference points that show the entities interaction with each other (3GPP 2016). The system architecture entities are responsible for connectivity and access procedures in 5G.

- ng-RAN: A RAN that supports new and old radio extensions that it connects to 5GC.
- SMF: Deals with session establishment, modification, and release.
- AMF: Deals signaling, access authentication, and authorization.
- UPF: Handles the user data and is the anchor point for intra- and inter-Radio Access Technology (RAT) mobility. It is the external Protocol Data Unit (PDU) session point to interconnect DN and it also handles packet routing and forwarding.
- PCF: In 5G it deals with policy rules.

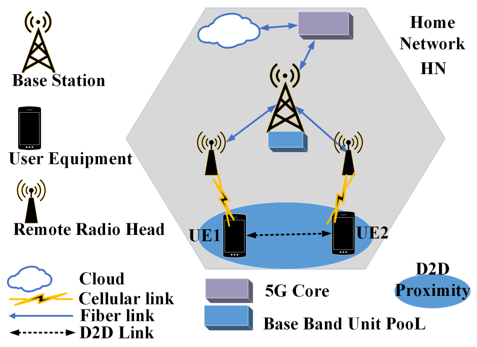


Figure 2.2: 5G enabled D2D Architecture

- AF: Provides services such as application influence on traffic routing, access to the Network Exposure Function (NEF), and interaction with the policy framework.
- DN: It identifies SP services, Internet access, or 3rd party services.

Since D2D communications and ProSe functionality for 5G have not been defined yet, this research uses D2D communications architecture discussed in chapter 4 as shown in Fig. 2.2 and leverages CCN and 5G NFs. The integration of CCN with mobile network enables content-aware operations such as content resolution at the edge which is within 3GPP standardization. The Network Function Virtualization (NFV) (Liang & Yu 2015) allows the sharing of infrastructure resources and content between SPs and the delivery of content to users through network slicing. CCN in 5G can be implemented using NFV and Software Defined Network (SDN) where CCN are inherently integrated into the mobile network infrastructure (Ravindran 2019).

2.2.2 Network Services and Functions of 5G Core Network

NFs are used to facilitate the enablement of services and elements of 5GC, most of them turned into software-based functions. The modularity of NFs also enables network slicing, allowing multiple logical segments to run on a shared infrastructure. A NF can provide multiple NF services, consisting of operations based on a request-response or a subscribe-notify model between a SP and a consumer (3GPP 2020*g*). As a converged network, 5G will offer support for new network deployments and use cases seamlessly. The International Telecommunication Union (ITU) classified 5G NS as eMBB, mMTC, and URLLC. The eMBB intends to support users' digital lifestyle and provide high bandwidth services, such as High Definition (HD) video, VR, and AR. The mMTC intends to support tactile internet and governments' digitalized society by providing high-density connections, such as intelligent transportation, smart grid, and intelligent manufacturing. The URLLC intends to support the market and enterprises' requirements of digitalized industries, and by focusing on latency sensitive services, such as automated and assisted driving, and remote control.

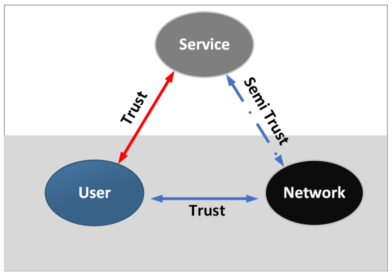
The heterogeneous service requirements have been recognised, whereby the benefits of CCN in 5G such as name-based networking, in-network storage, edge computing, security, and user mobility have been explored (Ravindran et al. 2017). Normally CDN framework enables subscribers to access content from remote BS or 5GC provided by MNO through a backhaul but with NFV, a subscriber can get the content from a BS or UE through D2D communications (Wang et al. 2017). Moreover, adding network service delivering functionality to the infrastructure can make the distribution of other services to the end users seamless (ETSI 2014). NS can be classified and mapped into abstraction levels, discussed in detail in section 2.5.

2.3 Overview of 5G Security Architecture

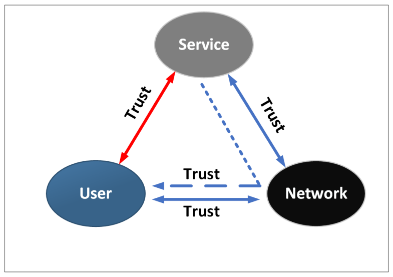
Traditionally, network security focuses on providing protection to network edges, preventing external threats from gaining access to the network resources, while attackers try to find vulnerabilities in the network and applications to exploit. This includes old threats inherited from 4G and new threats in 5G that can attack the radio interfaces, signalling plane, user plane, data plane privacy, and service-based interfaces, which require integrated security mechanisms. Since 5GC is based on a Service-Based Architecture (SBA), which did not exist in earlier generations, it should also be protected. The main objectives of designing a security architecture are confidentiality, integrity, availability with interoperability, usability, Quality of Service (QoS) and cost-effectiveness.

The legacy mobile networks' security architectures are now inadequate for addressing security in 5G. This is due to the introduction of new business models, threat environment, management, trust model, and network softwarization. Different services with specific requirements will be from both physical and Virtualized Network Functions (VFN) operating on the underlying infrastructure in 5G with many actors providing services, NF, and resources. The information technology environment and consequently 5G exhibit more advanced threats to NF and NS, which mission-critical services will rely on.

There is no complete trust model for earlier generations, however, this does not imply that they were insecure, but it does create an issue in 5G. Thus, a need to have a security architecture in which trust is considered in the security design of the network. Additionally, the security management domain was not part of legacy systems, it was operator dependent. 5G technologies will be integrated, hence the need for a unified management view for continuous interworking between NS and NF, enabling delegation of management responsibilities between HN and SP while providing specific services (5GPPP 2017). The existing trust model is not in line with 5G business and technology capabilities; however, it does not mean that security needs redesigning completely. The limitations in the architecture are due to trust in the network as edge and terminals are less trusted. The new trust model must be more flexible to include endpoints, cloud, and fog managed by HN



(a) 4G Trust



(b) 5G Trust

Figure 2.3: Service Trust Model

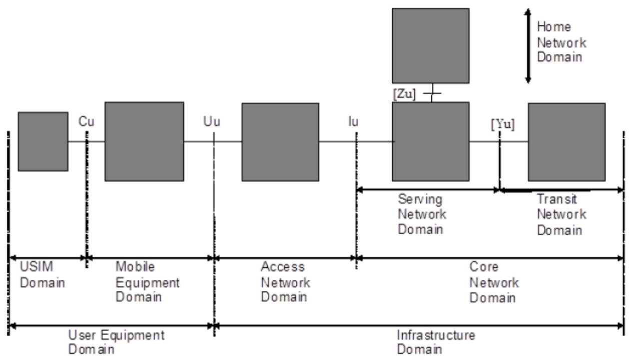


Figure 2.4: Strata and Domains (3GPP, 2018)

and SP as shown in Fig. 2.3. Moreover, there was only minimum protection of subscription/UE Identity (ID) and subscription authentication between the UE and third-party SP. The current IDs and attributes repositories also will not scale well due to the expected increase in the number of IDs in 5G, hence a need for FIdM.

In 5G, domain and stratum concepts in the core of the architecture are revised and expanded. The domains and reference points provide different viewpoints of the network as well as the model entities. Some domains consist of subdomains for entities' fine-grained grouping as shown in Fig. 2.4. For instance, the UE domain consists of the ME domain and Universal Subscriber Identity Module (USIM) domain (5GPPP 2017).

In addition, security mechanisms are categorised by defining security feature groups under TS 33.102 (3GPP 2020*a*) and TS 33.401 (3GPP 2020*c*) as (I) network access, (II) network domain, (III) user domain, (IV) application domain, and (V) visibility and configurability of security as shown in Fig. 2.5. Also, the architecture defines the security realms which are used to compliment the domain and the strata. To address the security requirements of a stratum or domain in 5G, security realms were introduced to capture their security needs in the form of access network, application, management, UE and infrastructure, and virtualization. Security enablers also try to address key security concerns such as authentication, authorization, availability, trust, security

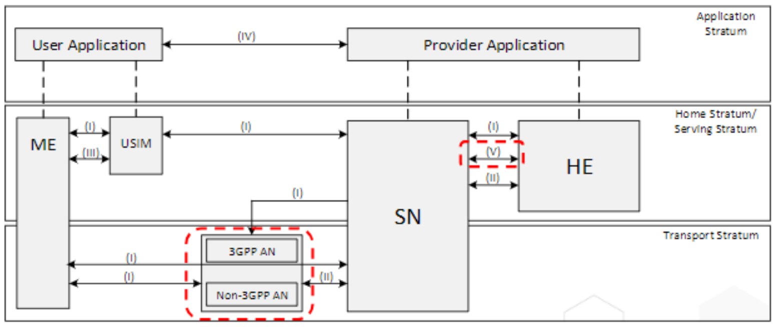


Figure 2.5: 5G Feature Groups (3GPP, 2018)

monitoring, privacy, and virtualization isolation, mapped with physical and functional viewpoints. Whereby Security Control Classes (SCC) are introduced to describe the security aspect for 5G system, which in most cases is neglected by existing security models. SCC is a collection of security functions mechanism mapped with security aspects, inspired by ITU-Telecommunication Standardization Sector (ITU-T)'s X.805 eight security dimensions (Zeltsan 2005), SCC are authentication, ID and access management, non-repudiation, confidentiality, integrity, availability, privacy, audit, compliance, trust, and assurance (5GPPP 2017).

2.3.1 Security Architecture

As explained earlier, the legacy security standards were not adequate for 5G, a new security architecture (3GPP 2020*f*) was introduced to enforce trust between new actors and other entities in the network. The new security techniques have been introduced like encrypting SUPI into Subscriber Concealing Identifier (SUCI). In addition, new security entities have been introduced in 5G security architecture (3GPP 2020*f*) such as SEcurity Anchor Function (SEAF), Authentication Server Function (AUSF), Authentication Credential Repository (ARPF), Subscriber Identity Deconcealing Function (SIDF) for decrypting SUCI back to SUPI, and Security Edge Protection Proxy (SEPP) for interconnecting one Public Land Mobile Network (PLMN) to another. The security architecture will have to consider NFV and SDN to achieve the overarching objectives of 5G. These security entities are responsible for UE and HN mutual authentication and service authorization, discussed in detail in chapter 5.

2.3.2 Trust Model

The trust model in 5G also has been modified but still, the UE has two trust domains, the Universal Integrated Circuit Card (UICC) which is tamper-proof on which the USIM resides as trust anchor, and network side trust model is for roaming and non-roaming cases with multiple layers trust models. The ng-RAN is separated into distributed and centralized units which make up the new generation 5G Node (gNB) node of BS. The distributed unit has no access to subscriber's data in transit due possibility of being deployed in insecure locations. The Access Stratum (AS) security will be terminated at a centralized unit and the non-3GPP interworking function will have restricted access whereas the AMF serves as the termination point for No-Access Stratum (NAS) security. The AMF is housed together with SEAF will hold the anchor key for the Visited Network (VN). 5G security design enables the separation between the security and mobility functions. Furthermore, the roaming architecture, HN, and VN are connected through SEPP for the control plane of the internetwork inter-connectivity.

2.3.3 Cryptography

To secure 5G communication, Elliptic Curve Integrated Encryption Scheme (ECIES) cryptography (Shoup 2001) is applied as one of the security measures to provide authentication, integrity, and confidentiality. The techniques applied include symmetric and asymmetric encryption, hash functions, MAC, SQN, eXclusiveOR (XOR), digital signature, and other techniques are explained next.

Public Key Infrastructure

Public key infrastructure (PKI) is a system of policies, procedures, and services that support the use of asymmetric encryption to obtain secure communication, developed by ITU-T under X.509 standard (ITU-T 2019). The security measures are implemented utilising key pairs among users, the message is encrypted by the sender with the receiver's public key and decrypted with the receiver's corresponding private key. An X.509 standard also defines the digital certificate format that binds the ID of the key holder to the holder's public key. For more trust authority, certification authority was introduced as the trusted party responsible for verifying and signing the certificates. Whereby node A signs the message with its private key and encrypts it with node B's public key, node B decrypts the message with its private key and the signature is verified with node A's public key.

Identity Based Cryptography (IBC) scheme (Shamir 1985) also uses asymmetric encryption and Trusted Key Generation (TKG) centres. In IBC, each node chooses its identifier such as Internet Protocol (IP) address or name as a public key, TKG generates the corresponding private key and distributes it to the node securely. With this scheme, the users can communicate and verify each other's signatures securely without exchanging keys. In 5G, this can be combined with other information such as timestamps, random nonces to UE identifiers when generating their public keys, ensuring a periodic update. However, it requires all users to have the most recent public key for that specific node. Additionally, ID-Based Encryption (IBE) can be used with Elliptic Curve Cryptography (ECC) primitives in constrained environments (Fanian et al. 2010), combined with Elliptic Curve Diffie-Hellman (ECDH) for key exchange to establish a session key (Qin et al. 2017). Hence, that is why 5G adopted ECIES which complements all its security objectives.

Elliptic Curve Integrated Encryption Scheme

ECC is a public cryptographic scheme that provides high-security performance with lower computational costs, making it suitable for low computational capabilities and constrained resources systems, such as mobile devices and RFID as it uses the point multiplication operations. Procedures such encryption and decryption of data, digital signatures, and key exchange can use ECC with the following applications:

- ECDH uses Diffie-Hellman technique with elliptic curves for generic key exchange.
- Elliptic Curve Digital Signature Algorithm (ECDSA), is the elliptic curve variant of the Digital Signature Algorithm (DSA), uses Rivest–Shamir–Adleman (RSA) and DSA algorithm.
- ECIES, an extended encryption and decryption scheme.

ECIES's encryption function is symmetric but the encryption key is generated with the ECC public/private keys of the entities communicating, defined under standards for efficient cryptography (SECG 2009). ECIES cryptosystem is based on elliptic curves that uses Elliptic Curve Discrete Logarithm Problem (ECDLP) (Smart 1999) for security guarantees. ECIES is an integrated encryption scheme with the following functions:

- Key generation and key agreement: Functions for generating a shared secret of two entities.
- Key derivation: Generates a set of keys from keying material and other optional parameters.
- Encryption: Symmetric encryption algorithm.

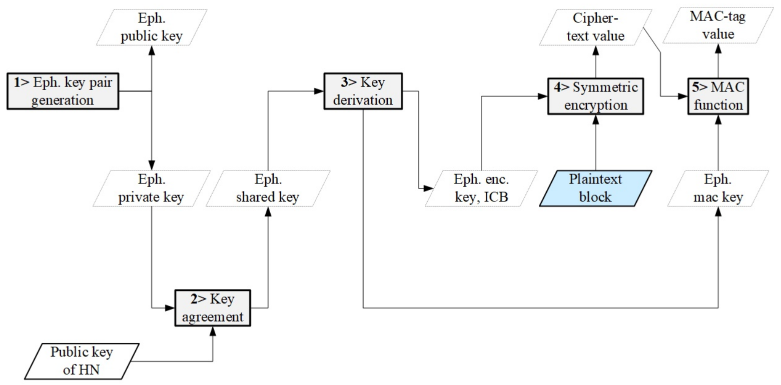


Figure 2.6: UE ECIES Encryption (3GPP, 2020)

- MAC: Message authenticating data.
- Hash: A digest function, applied with key derivation and MAC functions.

To encrypt the messages using ECIES, two communicating parties Alice(A)-Bob(B), A and B have ephemeral public keys `U` and `V` with responding private keys `u` and `v`, respectively. With ECC, private keys are elements of the finite field, are either `GF(p)` or `GF(2m)`, while public keys are points that belong to the elliptic curve and are calculated from the private key and generator `G` of the elliptic curve (SECG 2009).

ECIES-based Protection Scheme in 5G

The use of ECIES in 5G is to encrypt and decrypt SUPI which is defined in (3GPP 2020*f*) and shall follow the specifications in (SECG 2009) and (SECG 2010). It includes SUPI of type International Mobile Subscriber Identity (IMSI) with Mobile Subscription Identification Number (MSIN) and SUPI of type network specific identifier. Encryption on the UE side, computing a fresh SUCI, the UE uses the HN public key with new generated ECC ephemeral public/private keys from ECIES parameters provided by HN and based on the encryption operation defined in (SECG 2009) with few changes as in defined in (3GPP 2020*f*) as shown in Fig. 2.6. This output should be the ECC ephemeral public key, the ciphertext value, the MAC tag value, and ”any other parameters”. With ”any other parameter” it allows certain cases, e.g., point compression, it enables the sender to send additional sign indications. Decryption on the HN side, the decrypting of a SUCI by HN, the received ECC ephemeral public key of the UE, and the private key of the HN are used, based on the decryption operation defined in (SECG 2009), (3GPP 2020*f*) as shown in Fig. 2.7. There is no need for HN to generate new ephemeral key pair for each decryption.

2.3.4 Analysis

This section presented the security architecture overview of 5G, the related work to security mechanisms in this research such as authentication and authorization methods. It also discussed cryptography schemes and techniques specified in 5G standards such as the UE's cryptographic communic-

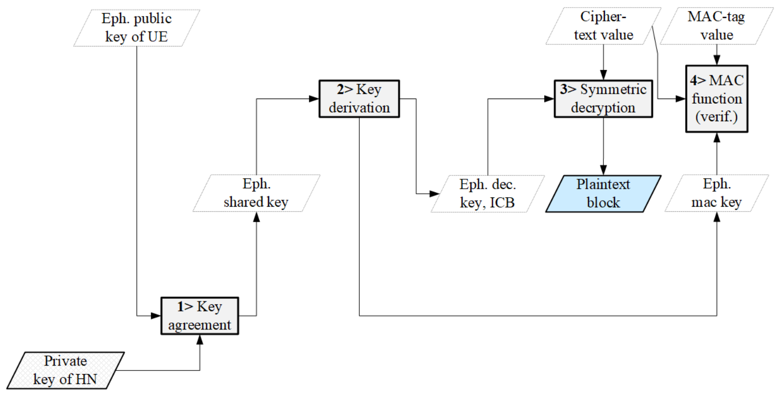


Figure 2.7: HN ECIES Decryption (3GPP, 2020)

ation capabilities and ECC scheme. The next section presents an overview of D2D communications and the related work. It aligns 5G enabled D2D communications with NS.

2.4 Overview of D2D Communications Network

D2D communication was defined in (3GPP 2014) for two devices in proximity to communicate directly without the BS routing assistance. It was introduced as underlay technology for cellular network to develop timing synchronization, peer discovery, and link management (Wu et al. 2013). D2D communication was intended to improve spectral efficiency, allow multi-hop relays in a cellular network, increase throughput and End-to-End (E2E) performance delay. The architectures of D2D communications, Mobile Ad Hoc Network (MANET) and cognitive radio networks are similar, but D2D communication is controlled by the BS for connection and synchronization. Some of the D2D use cases and applications are multicasting, video dissemination, cellular offloading, P2P, and M2M communications (Pratas & Popovski 2013), with 5G D2D applications shown in Fig. 2.8. The ProSe in LTE based on D2D communications and its architecture were explored by 3GPP, to

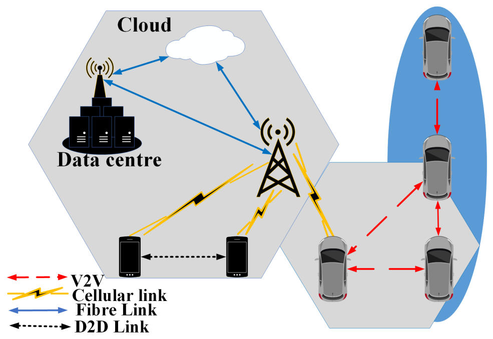


Figure 2.8: 5G enabled D2D and V2V Communications

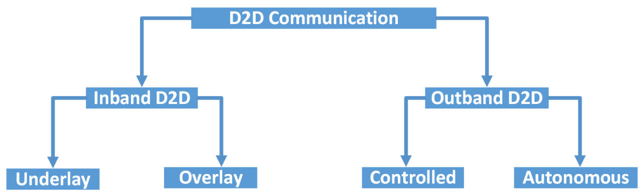


Figure 2.9: D2D Communications Modes

provide D2D users with exceptional connectivity. This was also a business opportunity for SP to provide new services to users such as location-based and context-aware services for local connectivity and content sharing. Hence, offloading traffic from BS and core network, leading to overcoming delay, and timing issues in D2D applications (Fodor et al. 2012). They also wanted to use D2D Communications for emergency communication in case of a disaster.

In LTE Advanced (LTE-A), LTE direct and D2D communications included ProSe and V2V communications to improve Vehicle Ad Hoc Network (VANET), while with IEEE 802.11 protocol Wi-Fi direct technology, devices could automatically act as both AP and client in D2D communications. Proximity-based data sharing services will also be supported in 5G enabled D2D communications. D2D communications transpire on a cellular licensed spectrum or unlicensed spectrum using the inband and outband modes. With inband mode, D2D and cellular users use the same licenced spectrum, sharing resources or D2D users get dedicated cellular resources, underlay, or overlay inband, respectively. The outband mode, allows the use of unlicensed spectrum by D2D users in controlled or autonomous mode, the radio interface direction is controlled by cellular network, and D2D communication is controlled by D2D users, respectively as shown in Fig. 2.9.

The D2D communications establishment uses the following techniques:

- Peer Discovery: Open discovery allows a device in proximity to be discovered by another device, while restricted discovery does not allow the discovery of a device without the BS's permission. Additionally, D2D users can identify each other using network discovery methods by sending beacon signals with identification and channel state information for pair grouping. It can be a network and device-centric based discovery.

- Synchronization: To get time and frequency and synchronization, periodic broadcasts are sent by BS and can be used by D2D devices from the same BS to synchronize. The right time slot and frequency are used by D2D users for discovery, communications, and energy efficiency.

- Mode Selection: During D2D communications, two devices can choose cellular communication, if direct communication is noisy or has more interference to get low latency and spectral efficiency.

- D2D Connectivity: Two devices to connect, the transport condition must be satisfactory, one device must check the request from the BS if direct communication throughput is higher than the cellular communication and the other device in range. A D2D radio bearer is chosen by the BS, one of the devices initiates connection and checks if traffic between devices has passed through gateway tunnels. Even though the BS is not involved in the direct communication, it still controls the radio resources.

- Resource Allocation: It dedicates the creation and maintenance of D2D direct links, only resource blocks that are not in use by the cellular devices are used. The industrial, scientific, and medical bands resources are not used in outband mode as they might be allocated to other D2D users in the proximity. There are some challenges to D2D communications that are brought by architectural changes and security requirements such as MIMO, physical layer, macro cell, and device tiers security (Agiwal et al. 2016).

2.4.1 Overview of Security and Privacy in D2D Communications

UE security is paramount to D2D communications security as the UE connects to the network through a ng-RAN, participates in D2D services, and uses supporting technologies, increasing the attack vector which can lead to compromising the security of the network and the UE (Haus et al. 2017). In D2D communications, the UEs manage auditing and logging activities usually managed by a centralized entity, a security mechanism should be in place to protect these activities. Additionally, the UE uses device discovery for detecting its peers, by broadcasting messages over wireless channels, which could also lead to location and privacy attacks (Gandotra et al. 2017). However, not relying on the BS for communication, the single point of data leakage is prevented, but D2D users still need to be protected from unauthorized access to UE data. Due to the self-management of D2D users, implementing security and privacy is difficult and it is going to be even more challenging in 5G, due to 5G's heterogeneous nature.

Moreover, ProSe are vulnerable to privacy invasion, which is a key concern in 5G and D2D communications. So, attacks like location tracking, data leakage, and unauthorized access to user's sensitive data need to be addressed. To meet the security requirements of 5G enabled D2D communications, authentication and authorization mechanisms must be implemented to secure the communication together with availability and dependability to guarantee QoS and QoE. The threats and security requirements analysis are presented in chapter 4. D2D applications require a concise abstraction in relation to the network architecture and service provisioning. The next section discusses NS abstraction.

2.5 Network Service Abstraction of D2D Applications

2.5.1 Mapping of Existing Related Works

Some applications from legacy systems will be used to support the next generation mobile network. The author believes that to link the applications to mobile network architecture, there should be a well-defined classification and mapping of D2D applications in terms of which part of the network architecture they are linked to. Therefore, abstracting NS at different network levels to enable full utilization of security and network applications should be in synch with the multi-level architecture and as discussed in this thesis. The European Telecommunications Standards Institute (ETSI) defines a NS as a group of NFs specifying E2E function to support network infrastructure to deliver customized services to end users (ETSI 2014). D2D communication and CCN are also classified as NS since they are used for specific connectivity services to the user.

The classification and mapping of the D2D communications network in legacy systems and 5G can be grouped into applications, infrastructure services, and network connectivity abstraction levels.

Application Level

This level is composed of the following OTT applications: live gaming, user requests, multimedia, social networking, local advertising for the UE. It contributes to the efficiency of secure content delivery and access by the UEs. D2D communications require high QoE, high resource efficiency, resource management. With an optimal rate allocation and description distribution performance can be improved between D2D users and the BS (Vo et al. 2018).

Infrastructure Service Level

This level is composed of the following NSs: ProSe, service discovery, resource allocation, content sharing, and delivery to UEs. Network capacity can be managed through resource utilization, content sharing, caching, and distribution. The use of incentives to persuade devices to cache content for others. Additionally, D2D transmission and caching policies can be used to implement mobile content caching and distribution efficiencies. UE helpers can be used to utilize context information for content distribution and dissemination to facilitate service delivery (Chandrasekaran et al. 2018).

Table 2.1: Network Service Abstraction with Modular Framework

OSI Model	**Services**	**Abstraction Level**	**5G Protocol Stack**
Application Presentation Session Layers	OTT applications	Applications	APP
Transport Layer	CCN	Infrastructure Services	TCP UDP
Network Layer	Internet - CCN	Infrastructure Services	IP NAS RRC SDAP
Data Link Layer	Connectivity - D2D	Network Connectivity	PDCP RLC Mac
Physical Layer	Connectivity - Radio Access	Network Connectivity	PHY

Network Connectivity

This level is composed of the following connectivity and communication applications: network routing, spectrum sharing, radio access, peer, and network discovery, mmWave, V2V, and D2D communications. With auto-mobile industry, vehicular connectivity, intelligent transport system benefiting from 5G enabled D2D communications network connectivity (Zhou et al. 2018) such as cooperative vehicular networks using Global Positioning System (GPS) and geographic system data.

2.5.2 Network Services Abstraction in Relations with 5G Protocol Stack

The development of D2D applications for 5G can benefit from the proposed NS abstraction, by using abstraction, the applications can be mapped to one or more abstracted levels. The three levels enable the meticulous development for connectivity and security solutions that are interoperable and easily integrated on different levels. The proposed abstraction levels can be used with X.805 security framework (Zeltsan 2005) to evaluate security in D2D communications network presented in chapter 4, aligning it with modular architectures and systematic analysis. As shown in Table 2.1, NS abstraction levels are mapped with Open System Interconnection (OSI) reference model, 5G protocol stack, network layers, and NSs. These include Applications (APP), Transmission Control Protocol (TCP), IP, User Datagram Protocol (UDP), NAS, Radio Resource Control (RRC), Service Data Adaptation Protocol (SDAP) Packet Data Convergence Protocol (PDCP), Radio Link (RLC), Medium Access Control Layer (Mac), and Physical Layer (PHY). This enables the development of abstracted solutions at multi levels of the network architecture supporting multiple connectivities, service delivery, and an integrated security framework as proposed in this research.

2.6 Content-Centric Networking (CCN)

In 5G, end users will be able to access and share content faster at the edge due to efficient content distribution and retrieval processes. They will participate in content delivery while on the move with their UEs in cellular or D2D mode. Content sharing, delivery, and distribution in D2D communications can be achieved efficiently by using CCN delivery techniques. CCN addresses some of the 5G requirements that are difficult to satisfy with current IP networking. CCN architecture (Jacobson 2009) can be used as a single protocol that is capable of handling mobility, security, unify layer interfaces and integrate wired interfaces with wireless APs. It combines computing, storage, and networking into one paradigm hence enabling ubiquitous virtualized service logic and caching functions. Inherently its security enables the authentication of user's request and response messages, while the content caching is location independent. Moreover, flat architecture is applicable without the use of specific gateway, connectivity of applications and devices implemented via C-RAN, enabling services' deliverance of high bandwidth and low latency requirements (Ravindran 2019).

However, ICN architectures such as CCN have their security challenges as it is vulnerable to attacks like Denial of Services (DoS), interest flooding, signature key retrieval and cache pollution, (Ahlgren et al. 2012), (AbdAllah et al. 2015). Privacy is also an issue in CCN whereby ID authentication and management are the core blocks in designing a secure network. ACL and encryption mechanisms can be deployed to address some of these issues (Tourani et al. 2018), (Nunes & Tsudik 2018). In addition, evaluation of the security requirements for a concise security approach is a good starting point as demonstrated in chapter 4.

This research focuses on CCN (Jacobson 2009) as it is one of the popular and efficient ICN architectures proposed for mobile content delivery. CCN characteristics are hierarchical naming, content caching, and named data object routing. The content provider's name is used in the routing made possible by the hierarchical naming principle. Moreover, CCN is a content-oriented communication model driven by content interests and data objects whereby the content requests and replies are identified by data/object name which is a Uniform Resource Identifier (URI) like a string of variable length name segment, that can be the provider's domain name and content folder, for example, (*sp.domain.com/videostreams*). The requests are routed directly towards the provider and replies towards the requester using the same route while the data is being cached along the way (Jacobson 2009). The cached content can identify requests and reply with the requested data if it is matched, otherwise it forwards the requests. A content requester asks for content by sending their interest to all connected nodes, any node with that data can receive the interest message can reply with a data object packet message. With the interest and data identifying content by name, the nodes interested in the same content can participate by sharing the transmission.

Additionally, a node in CCN can forward an interest packet from a consumer in a hop-by-hop manner using its data name. The CCN separates the identifier and locator role, each data object is identified by Named Data Object (NDO). In this research, the NDO is referred to as content or data interchangeably, NDOs are published at nodes with routing protocols helping in the distribution of data (Loo & Aiash 2015). The request message is routed based on a data name instead of hostname and the architecture uses a transport protocol called CCNx for content delivery (Jacobson 2009). A CCN protocol is embedded in the nodes which consist of functionality such as the Forwarding Information Base (FIB) for guiding interest messages towards data for specific stored content, Pending Interest Table (PIT) stores data about the forwarded interest which are not satisfied yet by content object and Content Store (CS) stores the replicated cached content object which can be reused (Ahlgren et al. 2012).

Generally, in a mobile network, the communication channel is protected by using security protocols, which require entities to trust each other to deliver correct data over the channel. However, ICN architectures provide data integrity and origin verification, independent of the data source, enabling ubiquitous caching with integrity and authenticity (Ahlgren et al. 2012). These approaches are primarily applicable to any accessible content that does not require access authorization or delivery of encrypted content. However, ACL mechanisms are needed in CCN to deal with authorization.

2.6.1 Integration of CCN with Mobile Cellular Network

As mentioned in chapter 1, to solve the problem of increasing traffic in 5GC, besides D2D Communications, the integration of CCN into the mobile network can support offloading of traffic, access, and sharing of content efficiently. It can also reduce data latency in C-RAN and improve QoS and QoE. Moreover, the cache store and content server can also save costs of inter operators' communication, content provisioning, and consumer usage fee. Even where the content producer is different from MNO or SP, they can pro-actively push and cache popular content to the cache-enabled intermediate proxy devices or the UEs in advance, then the UE can get the content from the ideal device via D2D communications (Yang et al. 2015). 5G entities including the UE can be integrated with CCN functionalities to enable CCN content dissemination and allow UE to request and access the content.

Integrating CCN with a mobile network is intended to turn the network infrastructure into being content-aware. Since the NDO and its payload consist of routing and security functionalities fields, they can improve network routing and security. CCN can also perform intelligent content-based operations such as popular content sharing, in-network content caching, and cache space utilization of forwarding nodes (Ahlgren et al. 2012) hence supporting efficient traffic offloading and content delivery.

With CCN, 5G can improve content dissemination and management. This is achieved by embedding the CCN protocol into the UE, BBU pools, edge routers, and 5GC entities (Chandrasekaran et al. 2018), (Zhang et al. 2019) or integrated as a NS or NF (Ravindran 2019), (Ravindran et al. 2017) to enable CCN functionality. The caching, content management, and CCN/IP protocol conversation is managed by BBU Pool. The IP-based entities embedded with the CCN protocol help in protocol translation to enable continuous communication in the network and during E2E communication between entities. Also, CCN integration in mobile network (Kim et al. 2017), will allow 5G to rely on its elasticity, enhancing user and control plane functions. The extension of NFs to include CCN as a function can enable the implementation of CCN PDU sessions thus providing CCN application and network slice as a service, part of CCN enablement in 5G (Ravindran 2019). Based on the CCN functionalities, there are two types of messages used in content dissemination and they are as follows:

- Interest Message: Sent by a consumer with an interest in the content, it can be satisfied by the content provider or cache hit node in the BBU pool or the D2D device.
- Data object message: Sent back by the content provider or cache node replying to interest message in NDO chunks (Jacobson 2009). This cycle relies on the functionality of FIB, PIT, and CS.

The participating entities include the following:

- UE: The end user device requesting the content and delivering the content to other UEs.
- RRH: Connects the UE to the BS and forwards the interest packet to BBU Pool.
- gNB: Is responsible for secure connection of the UE to the network.
- BBU Pool: Connects the 5GC and ng-RAN.
- 5GC: Generates data and administers security within the network. The edge routers, gateways, content, and security servers reside in the 5GC. It might also store the original content.
- Content Producer: Publishes the content under namespaces and could be the MNO/SP.

ICN functionalities in a mobile network are also discussed in (Ravindran 2019), (Chandrasekaran et al. 2018), (Zhang et al. 2019), the CS in BBU pool is tasked with content registration, recording all the cached content of the UE, the content owner in coverage, and maintain the metadata of each NDO. When a recorded data matches a request, the BBU pool establishes a D2D link between a content requester and content owner, then the content owner can reply to the request from the content requester through cellular or D2D communications. The CS can be virtualized for cache size allocation management as well as content caching optimisation and provide efficient UE distribution and delivery. The CS in UE stores cached content after the initial access of the content.

The UE generates content requests initiated by the application layer while the CCN layer produces interest packets. The RRH establishes a physical connection with the UE, demodulates the radio frequency signal, and extracts the baseband signal. The RRH is not involved in any CCN functionality as it is not designed to deal with higher-layer data processing. The BBU pool forwards the inbound packets from external networks to the relevant RRH. It is also tasked with extracting the requested data object name and other information, which are used in matching the cached content as well as looking for forwarding routes. So, if the request cannot be satisfied at the BBU pool, it is then forwarded to 5GC or DN for the content provider to satisfy. The BBU pool is also capable of performing other duties such as processing the content request of another CCN-based packet or an IP-based packet accordingly.

Furthermore, 5GC is configured to operate the CCN protocol, since the MNO is responsible for generating data in the network, content servers are usually located in the 5GC as MNO/SP services and DN as third-party SP services. For functionality purposes, the 5G protocol stack is aligned with CCN and IP protocol , with non-content dissemination services such as IP-based services still processed in the normal way. Embedding the CCN protocol and functions in UE, BBU pool, and 5GC enables the extraction and resolution of data-name as well as matching and forwarding UE requests. The author believes integrating 5G enabled D2D communications with CCN will enable the offloading traffic from 5GC and push content to end users. Therefore, this research considered proposing security mechanisms that address both entities and data security.

2.6.2 Analysis

Sections 2.5, 2.4, 2.6 presented an overview on D2D communications, security, and privacy and discussed network services in 5G. It also gave an overview of the CCN and the significance of its integration with D2D communications. Since D2D communications network is defined as a network service and underlay technology for 5G, it is vital to explore its functionality, security, and integration with CCN to achieve 5G objectives.

2.7 Summary and Overall Evaluation

D2D communications will benefit 5G as the underlay technology, but with 5G standardization yet to be finalized, the security of 5G enabled D2D communications requires an extensive investigation to address security concerns. The abstraction and mapping of network services can be applied with different frameworks to assist in addressing service and security issues in 5G efficiently.

As discussed in this chapter the related work attempt to address the security of 5G enabled D2D communications, however, the security of NSD is still not addressed fully. With 5G enabling new use cases and using D2D communications to offload traffic and push content to end users, it is a big challenge. For this research to provide resilient and an integrated security solution for NSD in 5G enabled D2D communications, it benefited from related work such as 3GPP 5G standard (3GPP 2020*f*), C-RAN architecture (Checko et al. 2015), the concept of integrating ICN in 5G (Ravindran 2019) and mobile network (Chandrasekaran et al. 2018), (Zhang et al. 2019). The full integration of the mobile network with virtualization has broadened the spectrum of 5G but also increased its attack vector which requires multi-level security techniques to provide robust security solutions. The research also benefited from a multi-level security approach in (Aiash et al. 2014).

But the study in this research is different from that of related work in many ways. We considered an integrated system model that aligns with the 5G objectives, end users' content access, and D2D communications. The related works tackled security of 5G supporting technologies separately, our research considered an integrated solution. The research also proposes a host-centric and information-centric hybrid solution that addresses the entities and data security. Unlike the related work, this research conducted a comprehensive security evaluation of the system model using X.805 security framework (Zeltsan 2005) while protocol security properties analysed using two security taxonomies (Lowe 1997), (Menezes et al. 2018). Moreover, the proposed security protocols are analysed using ProVerif and applied *pi* calculus (Blanchet et al. 2020) and performance evaluation is based on Markovian method (Stewart 1994) and network simulation (Nsnam 2021).

The next chapter 3 discusses the approaches and methods used in this research to address the security issues and how the proposed solutions were developed.

Chapter 3

Research Approaches and Methods

3.1 Introduction

The related work to date has been explored in the literature review chapter 2 to enable the evaluation of existing problems and limitations to present suitable solutions to address the defined problems with an intention of improving 5G's security. This chapter presents methods and approaches used to introduce the system and threat models for identifying threats and proposing security solutions for 5G enabled D2D communications system. It starts by recalling the research question and proposed work in section 3.2. Section 3.3 presents the methodology followed to address the research question. Section 3.3.5 validates the verification methods and tools. The security properties taxonomies are explained in section 3.3.8. Section 3.3.9 discusses the performance evaluation methods. The chapter is summarised in section 3.4.

3.2 Proposed Work

We propose to investigate the security and privacy of NSD in 5G enabled D2D communications, evaluate security requirements and develop novel approaches to support the secure service delivery in D2D communications. We introduce a service delivery framework to compliment the security framework and mechanisms proposed in this research.

3.3 Methodology

This section explains the methodology followed to answer the research question to achieve the objectives of this study. Like any other computer system, 5G communication is complex with a variety of requirements as discussed in the chapters 2 and 4. To meet these requirements, specification, design, verification, and evaluation have a vital role in the development and analysis of 5G security process. With this type of research, there are different approaches that can be followed to meet those requirements and achieve the research objectives.

The commonly used approaches are real experiment, simulation, and analytical modelling or a combination, which all require testing, validation, and analysis. Firstly, a real experiment requires the use of a testbed to carry out the implementation and testing of our proposed security solution. It provides better information about the communication system using realistic conditions. However, building and managing a testbed is expensive and complex, especially when implementing and testing communication in mobile networks.

Secondly, discrete event simulation tools such as NS-3 (Nsnam 2021) and OMNET++ (Varga 2010) can be used to simulate the network activities. A discrete network simulator is built to represent a computer network simulating the functionality of entities and the communication channels in a network. Simulation provides scalability with repeatable results and can simulate complicated scenarios. It requires a series of long simulation runs to have an elaborative and correct analysis. Since a mobile system consists of complex characteristics and various parameters, simulations take time. Even though it is cheap with easier performance processes, currently it is very difficult to simulate a complete stand-alone 5G communication system as the only available models can only simulate E2E communication with mmWave module as radio technology and LTE as a core network, same for D2D communication (Mezzavilla et al. 2018). Moreover, at the moment there is no simulation tool that can capture all the entities, communication processes, and security parameters.

Thirdly, analytical modelling is based on a mathematical description of the proposed solution

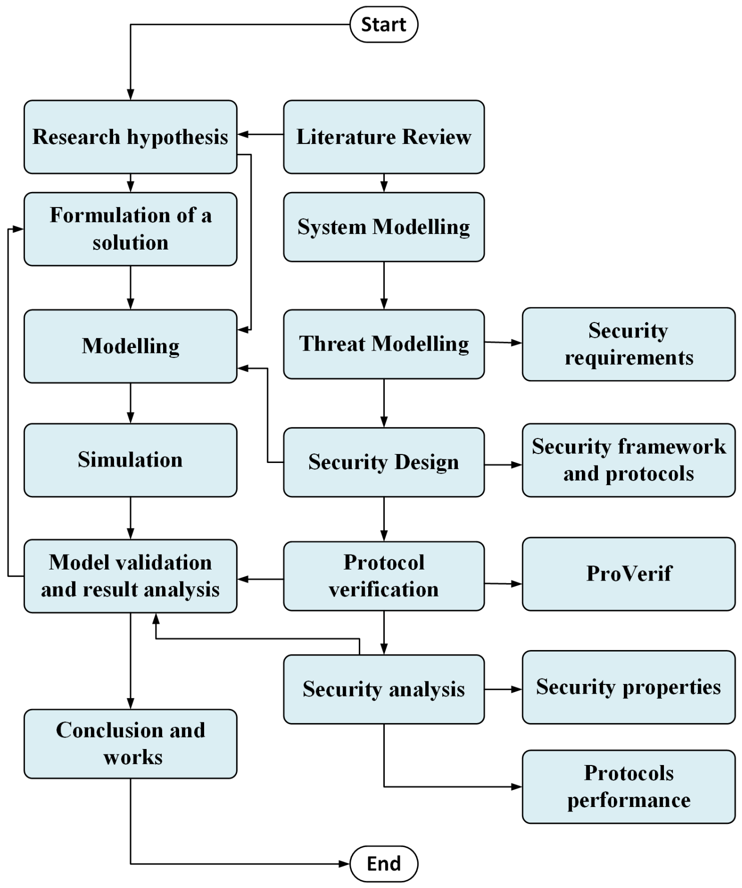


Figure 3.1: Research Methodology

using configurable computer states, mathematical theories, and techniques such as queuing and probability. This enables the modelling of a system with transition between states and provides approximate solutions applicable in computer systems. Numerical methods are applied to the state model using tools and analytical processes such as like MATLAB (MATLAB 2019) and Markov chains process (Stewart 1994), respectively. This approach provides accurate results, obtainable formulas, and faster computations. Even though it relies on assumptions for approximate results, it is the preferred method for quick and accurate computations validation. However, it is not realistic as an experiment or flexible as a simulation.

For this research, considering the resources availability, system complexity, and other constraints, which include the research's objectives and lack of a testbed, we use the combination of simulation and analytical approaches as shown in Fig. 3.1. This methodology consists of methods and approaches that use mathematical theories and computer tools to define, model, and analyse a computer network communication system (Basin et al. 2018), (Hussain et al. 2018). It starts by defining a system model and performing threat modelling to investigate the security issues. Then designing and simulating security protocols using a protocol proof verifier to model and verify the protocols to address security issues in 5G enabled D2D communications as presented in the following chapters. As stated in chapter 1, this methodology was proposed after identifying the constraints in different studied areas of this research in the literature review.

Conventional simulation approaches cannot be used for complete coverage of complex systems of 5G and D2D communications as modules such as the control plane and ProSe function are still not available. Therefore, a mathematical model based on formal methods together with simulation based on an abstract representation of the proposed protocols is applied to improve the security of 5G enabled D2D communications. This approach is also used in the validation of the developed solutions.

3.3.1 Network Services Delivery (NSD) Framework

To investigate the security of NSD in 5G enabled D2D communications network, we present a service delivery framework based on C-RAN architecture (Checko et al. 2015), NS abstraction in section 2.5, D2D communications (3GPP 2014) and CCN (Jacobson 2009). It focuses on the entities involved in the communication and how the services are accessed by the UE and shared with other UEs but not how the data is stored or accessed on the application level. The NSD framework consists of system, access, and delivery models to illustrate the delivery of services. Additionally, to define the entities of the system model to give an overview of 5G enabled D2D system model and security architectures. The NSD framework is discussed in chapter 4.

3.3.2 Threat Modelling

In order to discuss security threats in 5G enabled D2D communications, a clear adversary model to evaluate security and privacy protection mechanisms is required. In our threat model, the attack is based on a Dolev Yao (DY) model (Dolev & Yao 1983), an adversary that formally models the attack against communication, assumed to be the communication channel.

Attacker's Capabilities

The adversary is capable of intercepting, tapping channels, and eavesdropping on the transmitted messages. In addition, the adversary can create, read, modify, capture, block, remove, replay, and send messages on the wireless and wired communication channels on the network. Which enables the attacker to initiate the attack discussed in section 4. By listening to signalling messages, setting up fake BSs to impersonate SNs, the attacker can compromise secure entities like USIM as well as other entities in SN and HN.

For instance, the adversary can compromise UEs and other entities by revealing the secrets between UEs as well as applying her own public functions like encryption, hashing, and signing on values known to her. By impersonating any entity, she can become a legitimate user capable of initiating communication and responding to interest messages sent by legitimate UEs. Might also try to repudiate its malicious behaviour hence preventing data sharing among users. Furthermore, the attacker can encrypt and decrypt messages with known keys. However, if the attacker does not know the correct decryption key for a given ciphertext, then the attacker cannot gain any information from the ciphertext. Also, no attacks are considered as the result of the implementation environment. In addition, unbounded message lengths, unbounded number of fresh nonces, and protocol sessions allowed by the threat model.

3.3.3 Solution Approach

After the introduction of the system model and defining the involved entities, the next step is to investigate the security issues that could occur in 5G enabled D2D communications. Then introduce our security framework which is a hybrid solution that integrates host-centric and information-centric security schemes to address the security issues in 5G enabled D2D communications network. This security framework includes authentication and authorization mechanisms verified using formal methods techniques integrated into t=a complete security solution.

3.3.4 Network Services Security (NSS) Design

With the NSS framework, a security model needs to identify what can be used to protect the entities and data of the proposed system model in a layered manner. We need to identify security protocols to be developed to provide security on different levels of 5G enabled D2D communications network. The security model relates to network access, service level, and D2D level security. Network access security is concerned with protecting the initial connection between the users and the network while service-level security is concerned with restricting access to the services from the end users. Additionally, D2D level security is concerned with restricting service and sharing between two end users. The security protocols are to be used with the security model to address the security of the system model presented in chapter 5.

Next is to verify the proposed security protocols and check if they are vulnerable to any security threats from the proposed system model point of view. Therefore, the next section presents methods used in verifying these protocols.

3.3.5 Security Protocols Verification

In order to measure the effectiveness of security protocols in form of security guarantees, validation methods such as Unified Modelling Languages (UML) and Specification and Description Language (SDL) can be used to check protocol specification against undesired states and behaviour. But to check the claimed security properties of a protocol, methods that use mathematical logic or model checks are required such as Temporal Logic of Actions (TLA) (Lamport 1994) or Burrows-Abadi-Needham Burrows Abadi Needham (BAN) logic (Burrows et al. 1989) to determine the relationship

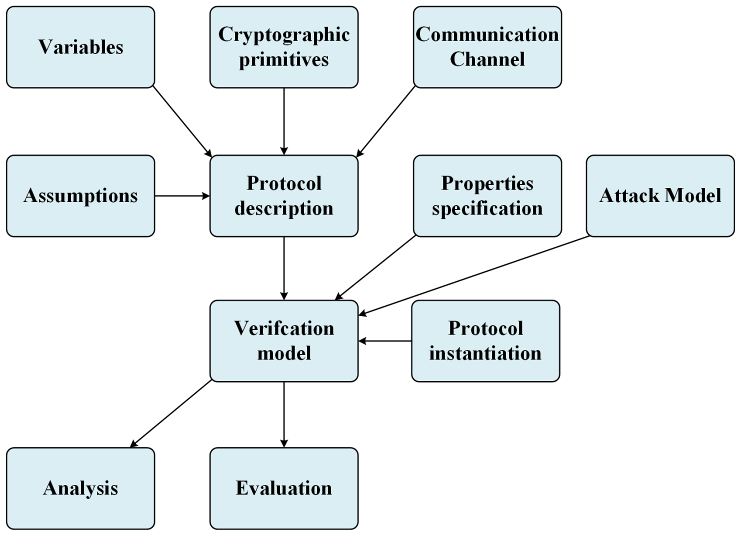


Figure 3.2: Cryptographic Protocol Modelling

between parties based on theorem proofs and verification logic. However, in the BAN logic method all parties are assumed honest and only authentication properties considered (Boyd & Mao 1994), hence, leaving out secrecy analysis. Additionally, formal methods can be used to verify the secrecy and authentication properties of a security protocol relying on automated tools. Therefore, formal methods are applied to determine the security guarantees that should be met by 3GPP security protocols and the proposed security protocols in this thesis.

Formal Methods and Automated Protocol Verifiers

Formal methods and automated tools can be used to achieve verification of cryptographic protocols by applying different approaches such as symbolic and computational modelling as shown in Fig. 3.2. A symbolic model like DY assumes the perfect cryptography to allow the cryptographic primitives to be represented as symbolic operations. When a property is proved in a DY model, it means the protocol is accurately modelled at an abstract level with no attacks. The computational model (Goldwasser & Micali 1984) uses complexity theory, messages as bitstrings, and cryptographic operations are probabilistic algorithms on the bitstrings where a polynomial-time probabilistic turing machine is the attacker. The proof of security is based on reducing the security properties to computationally hard problems.

The analysis of security protocols for mobile network is complex and challenging whereby most protocols are utilized before being formally verified leading to various attacks being found in full operational protocols using formal methods with automated verification tools such as Automated Validation of Internet Security Protocols and Applications (AVISPA) (Armando et al. 2005), ProVerif (Blanchet et al. 2020) and Tamarin (Meier et al. 2013). This shows why it is necessary for protocol formal verification for security functionality in mobile systems. The initial protocols specified for legacy systems were manually verified using TLA (Lamport 1994) and enhanced BAN logic (Boyd & Mao 1994).

Unfortunately, mobile network protocols like those discussed in this thesis consist of complex properties that are challenging for most verification techniques. For instance, 5G uses of SQN and re-synchronization process which is an issue for tools that use a bounded number of sessions reasoning and manual proof checks (Basin et al. 2018). In addition, these protocols are stateful and have numerous loops which require inductive reasoning. The AKA protocols also use primitives like XOR, which are hard to reason symbolically or manually because of their algebraic properties like associativity (Basin et al. 2018). In this study, ProVerif (Blanchet et al. 2020) was chosen as the protocol verifier over other tools such as AVISPA and Tamarin as it is commonly used for its precision operation, good running time performance, wide functionalities, under active development, and still supported. It uses an intuitive and expressive modelling language, while the analysis technique has been formally defined and its proven soundness fits our objectives. For those reasons, ProVerif and applied *pi* calculus are found suitable for protocols' formal verification.

3.3.6 ProVerif and Applied Pi Calculus

Applied Pi Calculus

Formal methods are mathematical model techniques used in the verification of systems by performing mathematical analysis. Applied *pi* calculus is a specification language that uses formal methods and notations (Abadi et al. 2017), a widely used algebraic method for specifying and analysing security protocols with automated tools. It adds a symbolic application of functions and equations. A process calculus can be used to describe concurrent computation while applied *pi* calculus process

Table 3.1: ProVerif Syntax and Semantics

M, N ::=	*terms*
x, y, z	variable
a, b, c, k, s	name
f(M1, . . . ,Mn)	constructor application
D ::=	*expressions*
M	term
h(D1, . . . ,Dn)	function application
fail	failure
P, Q ::=	*processes*
0	nil
out(N,M); P	output
in(N, x : T); P	input
P | Q	parallel composition
!P	!P replication
new a : T; P	restriction
let x : T = D in P else Q	expression evaluation
if M = N then P else Q	conditional
event(M);P	event

is a sequence of operations with a finite sets T of functions and their arity, names N, variables V, and an equational theory Σ (Abadi et al. 2017).

ProVerif

ProVerif (Blanchet et al. 2020) is semi-automated tool that analyses security protocols using applied *pi* calculus, with the DY model, allows user-defined equation theories to verify protocol's security properties. Cryptographic primitives defined by rewrite rules and equations for satisfying the finite variant property are supported (Blanchet 2016), which excludes associativity. ProVerif's focus on the case of unbounded sessions and uses precise horn-clause abstraction, the abstractions' theory is very efficient. With the user-defined equation theories, ProVerif is sufficient to model XOR (Küsters & Truderung 2009). Applied *pi* calculus is used as a formal language to describe and model security protocols, by studying concurrency and process interaction. Furthermore, ProVerif and applied *pi* calculus formally make a relationship between mathematical representation and taxonomy of security properties using cryptographic primitives. ProVerif has been used to formally analyse the security properties of security protocols in (O'Hanlon et al. 2017), (Hussain et al. 2018), (Zhang et al. 2020).

The cryptographic primitives are modelled as functions, and terms represent messages built over an infinite set of names `a, b, c, . . .`, variables `x, y, z, . . .` and function symbols `f1, . . . , fn`. Functions symbols are presented as cryptographic operatives used on messages. The set of reduction rules shows the effect of applying function symbols to terms. The syntax and grammar of ProVerif process language are explained in Table 3.1, more details can be found in (Blanchet et al. 2020) and its structure shown in Fig. 3.3. ProVerif is more precise than the tree automata abstraction in AVISPA due to its focus on the case of unbounded sessions and horn-clause abstraction. It can also verify some equivalence properties.

Modelling Protocols in ProVerif

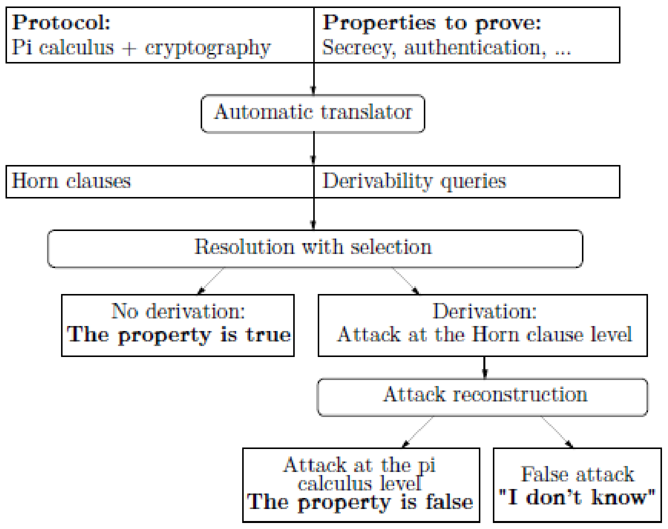


Figure 3.3: Structure of ProVerif (Blanchet, 2020)

ProVerif modelling of a protocol can be divided as follows: declarations, process macros, and main process. The declarations formalize the behaviour of cryptographic primitives. Process macros enable the definition of sub-processes for trivial development as the protocol is encoded as the main process, with the use of macros (Blanchet et al. 2020).

Declarations

Protocol modelling in ProVerif requires variables, constants and cryptographic primitives. It includes a free name or global variable and constructor with their corresponding destructor and definitions of the cryptographic primitives. The definition of process macros is explained as follows:

- User defined types **type** *t*, a type name for example nonce bitstring etc.
- Free name **free** *n : t.* Whereby *n* is a name and *t* is a type, free names are used to define global variables.
- Channel **channel** *c.* or **free** *c*: channel, syntax are synonyms, free channel is known to the adversary.
- Private channel **free** *c*: channel [**private**], private channel is not known to the adversary.
- Constructor **fun** *f(t1,..,tn) : t.* With *f* being a constructor of arity *n,* and *t* is the return type of its arguments.
- Private constructor **fun** *f(t1,..,tn) : t* [**private**], constructors not known to the adversary, used to store password tables.
- Destructors manipulate the terms created by the constructors and to convert the cipher-text back to its original form. The definition of destructor is as follows:
  reduc forall $x_{1,1} : t_{1,1}\,, \ldots.., x_1, n_1 : t_1, n_1; g(M_{1,1}\,, ..., M_{1,k}\,) = M_{1,0}\,;$ . . . . .
  forall $x_{m,1} : t_{m,1}\,, \ldots.., x_m, n_m : t_m, n_m; g(M_{m,1}\,, ..., M_{m,k}\,) = M_{m,0}\,.$
  above g is a destructor of arity k, the terms $M_{1,1}\,, ..M_{m,k}$ built after applying constructor to variable $x_{1,1}\,, ...x_1, n_1$ of type $t_{1,1}\,, ..., t_1, n_1$ (Blanchet 2016). The protocols use private and public channels c for exchanging messages and while constructors and destructor model cryptographic primitives are used in the protocol.

Processes

The brief definition of processes `P` and `Q` is that the null process 0 does nothing, `P` | `Q` is the parallel composition of processes for a protocol parties running in parallel. The !P replication as the infinite composition `P` | `P` |... . `new n : t; P` is the name restriction binding name `n` of type `t` within `P`. While process `in(c, x : t); P` waits for message of type *t* from channel `c`, then carries on as `P` with the message in bound to variable `x`. In that occurrence of `x in P` refers to the received message. The process `event(M); P` executes `event(M)`, and then executes `P`. Events annotate processes and mark the reached level by the protocol but are not affected by them. Events are used to specify correspondence assertions. The principals are modelled as applied *pi* calculus processes with events.

3.3.7 Security Properties

This section presents the security properties used in the modelling of the protocols in ProVerif and during the evaluation of the security requirements of the proposed protocols.

ProVerif Security Properties

ProVerif supports secrecy, reachability, correspondence assertions, observational equivalence, and authentication as properties specified as queries (Blanchet et al. 2020).

Secrecy: A secrecy property is specified as a query of the attacker's knowledge `attacker(M)`. When the fact `attacker(M)` is derived from the horn clauses, then the attacker may have the knowledge of `M`. But if the fact `attacker(M)` is not derived, there is no way that the attacker can gain the knowledge of `M`.

Reachability: The `query attacker(K)` is also used to debug the model of the protocol to check a particular branch is reachable or not. `query k: bitstring; event(endServer(k))`.

Authentication: The authentication properties are specified as correspondence assertions in the form of `event(e1(M)) event(e2(M))`. When the clauses conclude `event e1` with having `event e2`, then `event e1` can only be derived when `event e2` holds, hence correspondence assertion is proven.

Correspondence assertion is used to check the relationship between two events defined in the same sub process or in different sub processes, defined as follows:
`event e(M1, ...,Mn); P`

- **Event Correspondence**: The basic correspondence assertion is queried as:
  `query x1 : t1, . . ., xn : tn ; event (e1(M1, . . . ,Mj) )` $\Longrightarrow$ `event (e2(N1,. . . ,Nk) )` ., the `M1, . . . ,Mj ,N1, . . . ,Nk` are terms the application of constructors has built to variables `x1, . . . , xn` of types `t1, . . . , tn` while events `e1` and `e2'` are declared.
- **Injective Correspondence**: These assertions query one-to-one relationship:
  `query x1 : t1, . . . , xn : tn ; inj-event (e(M1, . . . ,Mj) )` $\Longrightarrow$ `inj-event (e'(N1, . . . ,Nk) )`. The correspondence asserts that, for each occurrence of the `event e (M1, . . . ,Mj)`, there is a definite earlier occurrence of the `event e'(N1, . . . ,Nk)`.

Type Conversion: Constructors declared data cannot be declared private, type converter is used to convert the type of output as to avoid mismatch: `fun tc(t) : t0 [typeconverter]`, type converter `tc`will take input of type `t` and then return type `t0`. Type converter is used to convert the type bitstring to id and keys or nonce to bitstring.

Equations: Certain cryptographic primitives such as XOR cannot be encoded as destructors because their algebraic properties need relations between terms. Instead, they are modelled as equations, `equation forall x1 : t1, ...., xn : tn; M = N`. The `M, N` being terms built to the variables `x1, . . . , xn` of type `t1, . . . , tn` from applying constructor symbols.

Formal Modelling of Cryptographic Primitives

The cryptographic primitives are modelled in ProVerif (Blanchet et al. 2020) as follows:
Symmetric and Asymmetric keys: Encryption and decryption are encoded as a constructor and destructor.
Symmetric Key: `type key.`
```
fun senc (bitstring, key): bitstring.
reduc for all m: bitstring, k: key;  sdec (senc(m, k), k) = m.
```
Asymmetric key: `type skey. type pkey.`

```
fun pk(skey): pkey. fun aenc(bitstring, pkey): bitstring.
reduc forall m: bitstring , k: skey; adec(aenc(m, pk(k)),k) = m.
```

Hash functions: The constructor takes as input, and returns, a bitstring:

```
fun h(bitstring):bitstring.
```

Digital signature: The equation for digital signature is

```
check(x,sign(x,sk(y)),pk(y))= True.
type sskey. type spkey.
fun spk(ssk): spkey. fun sign(bitstring, ssk ): bitstring.
reduc for all m: bitstring , k: sskey; getmess (sign(m,k)) = m.
reduc forall m: bitstring, k: ssk; checksign(sign(m,k), spk(k)) = m.
```

MAC: The MAC is formalized by a constructor without an associated destructor or equation.

```
type mkey. fun mac(bitstring, mkey): bitstring.
```

XOR: With XOR function due to the intrinsic equational properties XOR is represented as an equation

```
fun xor(bitstring, bitstring) : bitstring.
equation forall m1: bitstring, m2: bitstring; xor(m1, xor(m1,m2)) = m2.
```

Attacker Modelling in ProVerif

The attacker knows the free names by default. However, names can be declared private. Private names are not prior knowledge to the attacker. A constructor is declared as data, the attacker can construct and deconstruct data. Any party, including the attacker, can compute `sdecrypt(X,k)` if the attacker has obtained `X` and `k`, so `m` can be obtained if `X` is `sencrypt(m,k)`. In a symbolic model, the attacker can learn nothing from `sencrypt(m,k)` if the attacker does not have `k`. To have the capabilities as discussed in section 3.3.2, the attacker's computation abilities in ProVerif are defined as follows:

For each constructor f of arity n:

`attacker` $(x1) \wedge \ldots \wedge$ `attacker` $(xn) \implies$ `attacker`$(f(x1, \ldots, xn))$

`For each destructor}` g, `for each rewrite rule` $g(M1, \ldots ,Mn) \implies M$

`in def` (g): `attacker`$(M1) \wedge \ldots \wedge$ `attacker`$(Mn) \implies$ `attacker`(M)

Name generation: `attacker(a[ ])`

Initial knowledge: `attacker(`$pk(skA$ `[ ]))`, `attacker`$(pk(skB$`[ ]))`

The attack on a protocol can be explained using the attack derivation (abstracts) or attack trace (semantics). In ProVerif the derivation explains the actions made by an attacker to break the security properties of the protocol, it uses abbreviations of an internal representation of names or terms. Which is a numbered list of steps each corresponding to a process or the attacker's action. The attack trace represents the real attack as an executable trace of the considered process. The trace is a sequence of input and outputs on the public channel and of events in relation to the process. The input, output, or event are followed by their location at $\{n\}$ in a process, referring to the program point at the beginning of the process. As mentioned earlier when ProVerif is given `query attacker (M)` where M is the message transmitted on the channel c, it intends to prove that a property is unreachable by showing `not attacker (M)`, there for the `RESULT not attacker (M) is true` whereby the attacker does not have term M that is the attacker does not have `(M)`. With the query `query x1 : t1, . . . , xn : tn ; event (e(M1, . . . ,Mj) ) ==> event (e'(N1, . . . ,Nk) )` tests the relationship between events. The correspondence asserts that, for each occurrence of the event, there is a definite earlier occurrence of another event.

Now that formal analysis, verification methods, and security properties are explained, the next section discusses the security requirements that should be met by the proposed protocols.

3.3.8 Security Protocol Properties Analysis

The security protocols in this thesis should meet certain security requirements. The security properties are informally defined before being formalized for a protocol, the taxonomies (Lowe 1997) and (Menezes et al. 2018) are adopted, referred to as set 1 and set 2, respectively. The protocols should have some desirable security properties from a different point of view as defined in set 1 and 2.

Security Requirement Set 1

They are as follows:

- Aliveness: A protocol guarantees to an agent a in role A aliveness of another agent b if, whenever a completes a run of the protocol, apparently with b in role B, then b has previously been running the protocol.
- Weak agreement: A protocol guarantees to an agent a in role A weak agreement with another agent b if, whenever agent a completes a run of the protocol, apparently with b in role B, then b has previously been running the protocol, apparently with A.
- Non-injective agreement: A protocol guarantees to an agent a in role A non-injective agreement with an agent b in role B on a message M if, whenever a completes a run of the protocol, apparently with b in role B, then b has previously been running the protocol, apparently with a, and b was acting in role B in his run, and the two principals agreed on the message M.
- Injective agreement: Is defined to be non-injective where additionally each run of agent a in role A corresponds to a unique run of agent b.

Security Requirement Set 2

They are as follows:

- Mutual Entity Authentication: This is achieved when each party is assured of the ID of the other party.
- Mutual Key Authentication: This is achieved when each party is assured that no other party aside from a specifically identified second party gains access to a secret key.
- Mutual Key Confirmation: This requirement means that each party should be ensured that the other has possession of a secret key.
- Key Freshness: A key is considered fresh if it can be guaranteed to be new and not reused through actions of either an adversary or authorized party.
- Unknown-Key Share Resilience: In this attack, the two parties compute the same session key but have different views of their peers in the key exchange. In other words, in this attack, an entity A ends up believing she shares a key with B, although this is the case, B mistakenly believes the key is instead shared with an entity $E \neq A$.

- Key Compromise Impersonation Resilience: This property implies that if the Intruder compromised the long-term key of one party, he should not be able to masquerade to the party as a different party.

These requirements, together with formal analysis, verify the proposed security protocols as presented in chapters 6, 7 and 8. The next section discusses the evaluation process and methods used to evaluate the protocols' performance.

3.3.9 Security Protocols Performance Evaluation

Even though the main focus of the research in this thesis is not performance evaluation but in order to show the effectiveness of the overall performance of the proposed solution, a performance evaluation is conducted. We evaluate the performance of a security protocol using analytical and simulation modelling. Detailed evaluation metrics and results are presented in chapter 9.

Analytical Modelling

Analytical modelling is used to get intuition about the protocol performance measurements, applying numerical methods and Markov chains process (Stewart 1994), (Bodei et al. 2005*b*), (Nottegar et al. 2001), (Abadi et al. 2017). It relies on factors assumptions and theories that are translated in the model. With the lack of a suitable simulation tool and since the protocols in this thesis are modelled and verified using ProVerif that is based on processes and applied pi-calculus, analytical modelling based on the Markov chain model is used. The mathematical representations of the system model are based on queuing theory and probability is a key for analytical modelling. Detailed definitions and evaluation results are presented in section 9.2.

The analytical model to evaluate the performance of the proposed protocols follows these steps.

- Specifications: Using syntax to define the protocol process and messages.
- Enhanced operation semantics: Specialised semantics and grammar used in specification of a protocol and transition behaviour of a system.
- Quantitative measures: Quantifying cryptographic operation and message exchange with a cost on the system based on quantitative properties like length and speed.
- Cost derivation: Any quantitative measures that is derived from transition properties such as cryptographic procedure including encryption and decryption.
- Reward structure of the protocol: Associates a value with any state passed through in a computation process such as a protocol.

Simulation Modelling

The proposed security protocols were modelled in an automated protocol proof verifier and network simulator for effectiveness against attacks and overall performance, respectively. The protocol modelling and verification are simulated in ProVerif tool (Blanchet et al. 2020) to make sure that certain security properties are met, while network performance is simulated in NS-3 (Nsnam 2021) to model the security protocols and measure the network impact of proposed protocols in E2E simulation deployment setting to evaluate throughput and latency. NS-3 is an object-oriented and

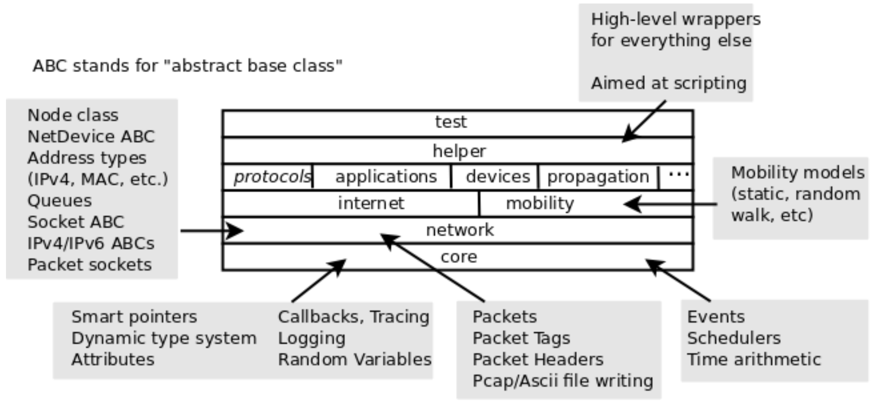


Figure 3.4: NS-3 software organisation (Nsnam, 2021)

event-driven simulator developed in C++ but also uses python, simulation modelling is conducted in both programming languages. However, writing the code in C++ is more convenient with a lot of reference material, written under the ns-3 namespace, using global defaults. NS-3 has an open and extensible architecture, it has been used to model and evaluate different wired and wireless networks and protocols (Kodali & Kirti 2020), (Sbai & Elboukhari 2019), (Malnar & Jevtić 2020), (Mezzavilla et al. 2018). However, it is complicated to model security protocols in NS-3, so we had to use a combination of modules and script modification to achieve our objective. Detailed definitions and evaluation results are in section 9.3.

3.3.10 Simulation Environment Settings

In order to perform protocol simulation, both ProVerif and NS-3 tool are installed and configured on a Ubuntu Linux virtual machine in VirtualBox environment installed on a windows computer.

Implementation environment setup is as follows:

- Windows 10: Processor Intel i7 - 2.40GHz, 16GB RAM and 250GB disk space
- VirtualBox
- Ubuntu 64bit operating system
- ProVerif 2.01
- NS-3.33

NS-3 Modules

The NS-3 is made up of mmWave and LTE modules that are used to programme and run a successful simulation, consisting of key abstractions such as node, application, Internet, net device and topology helper . The NS-3 programme organisation is shown in Fig 3.4.

- Node: represents the basic device (`numUe`, `numEnb`, `remoteHosts`) capable of computation to which various functionalities can be added to and it is managed by `NodeContainer`.

- Application: represents a user program responsible for generating an activity that needs to be simulated using `ClientApps` for client and `ServerApps` for server.
- Internet: a medium through which a data is transmitted between entities, `remoteHost` with bandwidth and propagation loss as configurable properties.
- Net Device: a network interface card is installed on a node to enable communication capability to work with the channels controlled by `NetDevicesContainer` such as `enbmmWaveDevs`, `uemmWaveDevs` and `internetDevices`.
- Topology Helpers: support connections between the elements created by the user such as `Nodes`, `Internet` and `NetDevices`.

The following topology helper are used.

- NodeContainer: provides the user with various ways to create nodes and manage them in form of a network.
- PointToPointHelper: `PointToPointHelper` is used to configure `PointToPointNetDevice` and `PointToPointChannel` objects to establish connection with `DataRate` and `Delay` attributes, respectively.
- NetDeviceContainer: holds the `NetDevices` created with devices and channels configured using `Install` methods enabling data transmission.
- InternetStackHelper: protocol stacks such as TCP, IP or UDP are installed on nodes.
- Ipv4AddressHelper: used to associate IP addresses with the devices on the nodes as an input by the user.

Implementation

To model and implement the proposed protocols, we adopt various models and modify them to suit our simulation models, the models are made up of modules written in C++ (Nsnam 2021), (Mezzavilla et al. 2018), (Banerjee et al. 2020). In order to simulate 5G network communication, the nodes, net device, applications and topology helpers were modified to represent communication of the proposed security protocols entities based on 5G architecture. The NS-3 E2E simulation structure with mmWave, eNB and UE radio stacks is shown in Fig 3.5 (Mezzavilla et al. 2018). The modelling starts by defining the variable set up with the command line argument and declaring local variables to control simulation parameters. The full code is in Appendix L. Nodes are created for each role in the simulation `numUe` for UE, `numEnb` for gNB/SMF and Severs `remmoteHosts` with `pgw` a gateway node. The size of the messages is defined as (M1, M2, M3, M4....) representing the cryptographic quantitative values. The simulation stop time is configure with `stopTime`.

```
uint32_t numUe = 1;
uint32_t numEnb = 1;
uint32_t serverNodes = 4;
uint32_t stopTime = 2400;
```

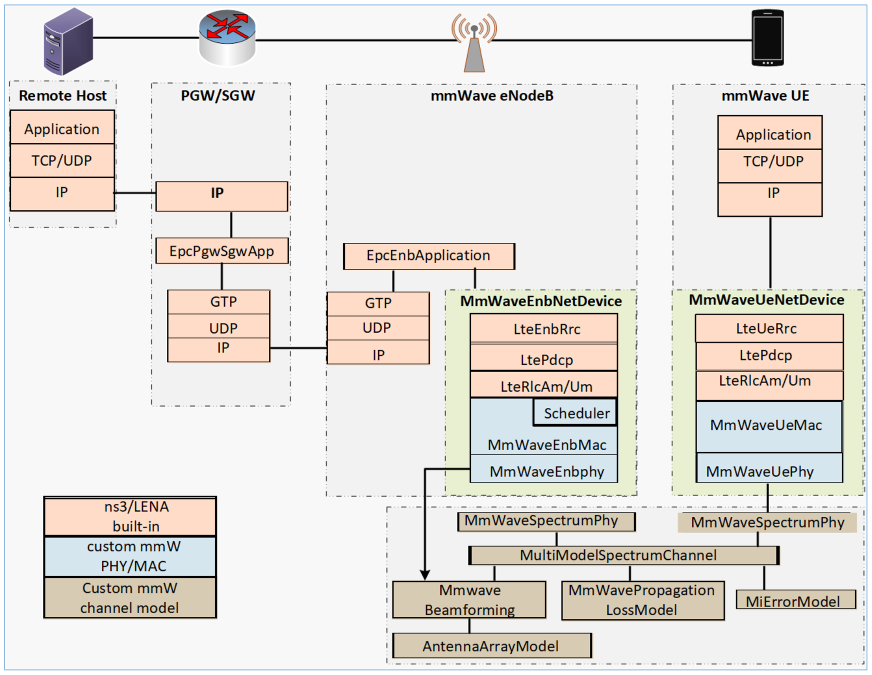


Figure 3.5: NS-3 mmWave end-to-end simulation structure

```
CommandLine cmd;
cmd.AddValue ("UE", "number of UE", numUe);
cmd.AddValue ("SEAF", "number of gNB", numEnb);
cmd.AddValue ("AUSF", "number of AUSF nodes", serverNodes);
cmd.AddValue ("ARPF", "number of ARPF nodes", serverNodes);
cmd.AddValue ("t", "simulation stop time (seconds)", stopTime);
cmd.AddValue ("M1", "Size of message 1 ", M1);
```

The UE, gNB, and servers are created with certain properties and attributes measuring metric parameters. The `MmwaveHelper` and `epcHelper` are necessary for defining the actual mmWave and EPC hardware.

```
NodeContainer ueNodes;
NodeContainer enbNodes;
enbNodes.Create (numEnb);
ueNodes.Create (numUe);

NodeContainer remoteHostContainer;
remoteHostContainer.Create (1);
Ptr<Node> remoteHost = remoteHostContainer.Get (0);
InternetStackHelper internet;
internet.Install (remoteHostContainer);
```

```
 NetDeviceContainer enbmmWaveDevs = mmwaveHelper->InstallEnbDevice (enbNodes);
NetDeviceContainer uemmWaveDevs = mmwaveHelper->InstallUeDevice (ueNodes);
```

In this simulation, packets movement and node mobility are configured using `MobilityHelper` with `uemobility` and `enbmobility` for the UE and gNB, respectively. The devices positions are set with `enbPositionAlloc` and `ListPositionAllocator`. The internet stack, routing, client, and server applications are configured to send and receive predefined data packets as a security message exchange between parties.

```
uint16_t dlPort = 1234;
uint16_t ulPort = 2000;
uint16_t otherPort = 3000;
ApplicationContainer clientApps;
ApplicationContainer serverApps;

UdpServerHelper dlClientHelper(port);
clientApps = authenticateA(clientApps, time , user, SEAF);
}
clientApps = authenticateB(clientApps, time , user, enbNodes.Get (0), SEAF);
}
clientApps = authenticateC(clientApps, time, UE, SEAF, ARPF, AUSF);
}
serverApps.Start (Seconds (0.0));
clientApps.Stop (Seconds (stopTime+1));
```

The `sendMessage()` function is used to create messages in `UdpClients`, while `authenticate()` and `authorise()` functions simulate the verification of messages. The `serverApps` and `clientApps` are used to start and stop the application.

```
ApplicationContainer authenticate(ApplicationContainer appContainer, double time,
Ptr<Node> user, Ptr<Node> gateway , Ptr<Node> device ){
appContainer = sendMessage(appContainer, time, user, device , M1);
appContainer = sendMessage(appContainer, time, gateway, device,  M2);
appContainer = sendMessage(appContainer, time, device, user, M3);
return appContainer;
}
```

In order to generate pcap files for analysis and a xml file for viewing in NetAnim to animate the simulation, the following configuration is done.

```
snprintf(saveFilePrefix, 50, "5GAKA_%dx%d_", numUe, numEnb, serverNodes);
if (enablePcap){
p2ph.EnablePcap (stringbuilder(saveFilePrefix,(char*)"_users"), numUe, 0);
p2ph.EnablePcap (stringbuilder(saveFilePrefix,(char*)"_devices"), numEnb,0);
p2ph.EnablePcap (stringbuilder(saveFilePrefix,(char*)"_gateway"), remoteHosts, 0);
}
if(enableAnim) {
AnimationInterface anim (stringbuilder(saveFilePrefix,(char*)"-animation.xml"));
```

```
Mandatory
for (uint32_t i = 0; i < enbNodes.GetN (); ++i)
{
anim.UpdateNodeDescription (enbNodes.Get (i), "UE");
anim.UpdateNodeColor (enbNodes.Get (i), 255, 0, 0);
}
```

The simulation is completed with the following

```
uint32_t bytes_received = 0, totalPacketsThrough;

for (uint32_t i = 0; i < serverApps.GetN (); ++i){
totalPacketsThrough = DynamicCast<UdpServer> (serverApps.Get (i))->
GetReceived ();
bytes_received += totalPacketsThrough ;
}

std::cout <<"Total packets received ("<< "UE="<< numEnb <<", SEAF="<<
numEnb <<", AUSF="<< serverNode1 << ", ARPF="<< serverNode2 << ") :
"<< bytes_received << std::endl;
return 0;
}
```

3.4 Summary

This chapter explored the proposed approaches and methods required to address the research questions. This research attempts to define a NSD framework that incorporates the D2D communications and CCN based on C-RAN architecture, to investigate the security and privacy of an integrated system such as 5G. In addition, it intends to propose security protocols, verify, and formally analyse the protocols using formal methods to integrate them using the security framework introduced in chapter 5. It also intends to evaluate the protocols' security properties and performance. The next chapter introduces a system model used to access services in 5G enabled D2D communications network. It discusses D2D discovery, communication process and presents different D2D application scenarios. Finally, it investigates threats and vulnerabilities against the proposed system model and conducts a comprehensive and systematic analysis.

Chapter 4

Network Services Delivery Framework

4.1 Introduction

With 5G promising to provide services close to the edge and improving end user's QoE, it requires efficient service access and delivery models. Its integration with other architectures and services can be provisioned as 5G NS. As aforementioned in chapter 2, the system model adopts C-RAN, CCN, and D2D communications to enable content dissemination in 5G. The applications presented in this chapter explore different scenarios that can be applied to D2D communications. A comprehensive security investigation is conducted, which explores security and privacy issues in D2D communications and service-oriented networks. This chapter introduces the system model, investigates the security threats against NSD, and analyses the security requirements of the system model based on the threat model and security properties presented in chapter 3. Some of the work in this chapter is also presented in (Edris et al. 2021*c*).

The rest of the chapter is structured as follows. Section 4.2 introduces a NSD framework. In section 4.3, an investigation on security and privacy threats in 5G enabled D2D communications is presented. The security evaluation based on x.805 security framework is presented in section 4.4. The security solution approaches are presented in section 4.5. The chapter is summarised in section 4.6.

4.2 NSD in 5G enabled D2D Communications

4.2.1 System Model

The system model in Fig. 4.1 consists of the UE, BBU pool, RRH, HN, and the SN. The UE is registered to the HN and receives roaming services from the VN. For the CCN integration, CCN protocol can be embedded into the entities and 5GC (Kim et al. 2017), (Zhang et al. 2019) or the control and user plane enhancement can be implemented with CCN functions within 5GC and extend the interfaces to support CCN PDU sessions (Ravindran 2019). The UE must be registered with MN/SP and subscribed for the services. With that the UE requests to connect to the HN,

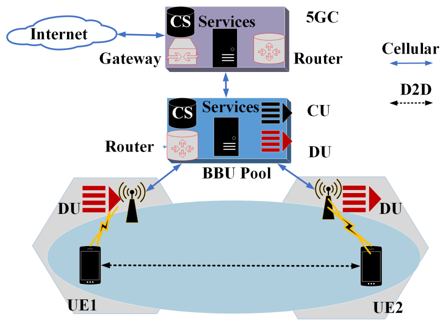


Figure 4.1: System Model

the primary authentication procedure is performed to authenticate the UE to the network then the UE requests to access the services depending on the Service Level Agreement (SLA) and the QoS agreement. After a successful network access authentication, a secondary authentication might be required for UE and SP communication, which authorizes the UE to access more services and perform extra activities such as data caching and sharing.

The SP is responsible for services subscription, authorization, and service provisioning. As mentioned earlier, the SP could be the MNO or third-party SP with their own infrastructure or using the MNO's infrastructure. The SP might want to hide their visibility or deny the UE from accessing the services. The communication channels between UE, SN, HN, and D2D devices are vulnerable to attacks, also that HN and VN could eavesdrop on D2D communication. This research is interested in content access and retrieval process, which deals with content discovery and delivery. Moreover, the MNO controls the user's access to the network, connection establishment, resource allocation, and security management while the SP controls services authorization.

4.2.2 Service Access and Delivery Model

The process of content dissemination and service delivery starts with UE generating an interest message which is forwarded to the BBU through RRH, then BBU starts a CCN forwarding process to match the request content name (Zhang et al. 2019). If the targeted content exists in BBU pool

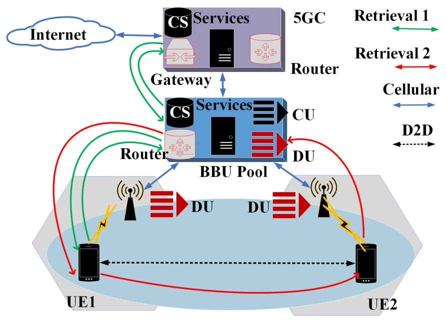


Figure 4.2: Service Access and Delivery Model

CS, then it is sent back in a reverse path traversed by the interest message to fulfil the initial UE request. During the content forwarding, the involved entities can choose to cache content replica or not, subject to the level of permission such as cache authorization. The content retrieval in D2D communications involves cache satisfaction at BBU, UE, and 5GC. Two scenarios of content retrieval are considered, scenario 1 where the content is being matched at the 5GC, and scenario 2 where the content is being matched locally by another UE in proximity as shown in Fig. 4.2. However, if the content request is not matched locally from UE in coverage or BBU, the interest is forwarded to 5GC, if it is matched in CS of 5GC then the content is pushed back towards the UE. In case the interest is not matched at this level it is forwarded to the internet, using the same routing, and forwarding process. The content is being cached by various entities as it is sent back to the UE. Therefore, the traffic burden is reduced on backhaul if the request does not go to the 5GC and the content retrieval is achieved at fronthaul or by D2D links as the BBU can discover cached content by associated devices and content transmission through D2D communications (Jin et al. 2017). Concurrently the security requirements must be fulfilled at the network, service, and D2D levels by performing authentication and authorization procedures. It is assumed that primary authentication between UE and the HN has been performed and the HN has already agreed on the communication terms.

4.2.3 D2D Discovery and Communication Process

D2D communications can be initiated to satisfy a service request by establishing a communication session securely and efficiently. The D2D communication between two UEs in cellular coverage is controlled by BS, whereas the out-of-coverage is controlled by the D2D devices. Generally, the D2D communication process in this thesis is like any D2D communications network with a few changes. So during the discovery stage, the UE broadcasts a request message to discover other UEs, when a UE receives a request message, it sends a response message back to the sending UE. The broadcast or response message in this process includes UEs' IDs, a Generic Public Subscription Identifier (GPSI) for outside HN communication. With the link setup stage for a D2D connection between two UEs, each UE sends a verification request to its gNB, with the GPSI of the target UE being received in the discovery stage. After verification, ECIES is used to support security vectors exchange, authentication, authorization procedures, and secure data transmission using the proposed protocols in this research presented in chapter 8.

D2D communications Network Assisted

The gNB can initiate D2D connectivity via SMF even though the UE might not be aware of any other UE in proximity. When the SMF in 5GC receives IP or CCN data packets, it is analysed to consider the link as a D2D link or not. This process starts from step 1 below and it is assumed that the UE can discover other UEs in proximity hence the UE can initiate D2D connectivity if it wants to share content with another UE as stated in step 3. The generic D2D connectivity process is explained below:

1. When the SMF receives a session request packet, it checks if the transmitter and receiver are within the same cell and the channel condition can support D2D communication then it initiates the D2D connectivity request to the gNB. The request messages include a UE ID, service ID, and data name.

2. Then gNB starts a strategy list of communication patterns, Physical Uplink Shared Channel (PUSCH), Physical Uplink Control Channel (PUCCH), power indicator, scan spectrum, and time, after analysing the request messages from SMF and it sends it to UE1.

3. The UE1 hoping to make D2D connectivity with UE2 sends regular device information in the Physical Random-access Channel (PRACH) with binary code, UE ID, data name, and the UE can also detect received information from other UEs.

4. After matching the targeted UE2, the initiating UE1 sends the D2D connectivity request to the gNB.

5. Then gNB analyses the D2D connectivity request, chooses the best communication pattern for the D2D pair based on the conditions of the channel and cell resource utilization.

6. When the D2D connectivity request fails, the normal cellular communication mode is adopted and the communication between the UEs goes back to the conventional cellular mode.

7. Conversely, if the D2D mode request including orthogonal or reused pattern is permitted, the gNB establishes a strategy list including communication patterns, power indicator, scan spectrum, and scan time and sends it to UE1.

8. After completing the channel measurement such as normalized interference intervals, UE1 provides an update to the gNB.

9. The gNB allocates the spectrum resource to UE1 and lets UE2 recognise the same channel. Training sequences at the allocated channel are sent by the UE1 to help UE2 to get the link quality. Then UE2 sends a confirmation message back to UE1 after the link quality is suitable and UE1 confirms to gNB.

D2D communications Non-Network Assisted

This is like MANET however, D2D link uses a reserved cellular licensed spectrum during an out-band communication, also referred to as public safety network. This is a direct communication between UEs with controlled link establishment by the UEs and it is explained below:

1. The UE1 hoping to make a direct communication transmits beacon signals with low frame rate to reduce signalling overhead.

2. The UE2 in proximity then responds by sending acknowledgement messages.

3. The received messages are evaluated based on metrics, such as Signal-to-Interference-plus-Noise Ratio (SINR).

4. The UEs can start direct communication with each other when a received signal is above the predetermined metrics threshold with a good QoS.

The beacon signals are sent over channels dedicated for transmitting control signals, whereas shared data channels are used for exchanging data between D2D UEs. The beacon signal should include security vectors within the packet frames for securing the D2D links. 5G UE is capable of an authentication process with low signalling overhead.

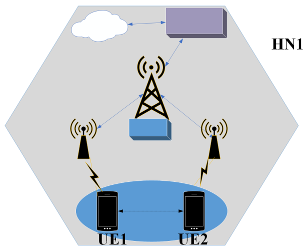


Figure 4.3: Scenario 1 Intra Network Operator and in-Coverage

4.2.4 D2D Application Scenarios

With MNOs and SPs providing NS as an independent and cooperative business model, users subscribe to both for specific NS. The UE connects via the HN and transfers to another network using handover techniques, digital certificates are used to establish trust and service agreements to support UEs. Firstly, the UE and the HN mutually authenticate and agree on a session key for UE and SN communication, network access security through AKA protocols discussed in chapter 6. Secondly, the UE requests service authorization to SP, discussed in chapter 7. Thirdly, the UE can establish D2D connectivity with another UE and perform security procedures as presented in chapter 8 completing the security solutions to address security issues as intended by this research.

The UE may request access via HN or VN in a roaming state, there should be predefined SLA, QoS, and roaming agreement between the UE's HN and VN for seamless connectivity and access. It assumed that two UEs hoping to establish a D2D link, have already been authenticated by their HN and the communication session is established securely and efficiently. There are two types of subscribers considered i.e., the one transmitting the content and the requester of the content, the role of subscribers can change from transmitter to requester and vice-versa depending on cellular coverage, D2D communication mode, and location of the content. This research focuses on four scenarios.

Scenario 1: *Intra Operator, Non-Roaming and Same Cell*
UE1 sharing content with UE2 in an intra-operator and non-roaming scenario as shown in Fig. 4.3, both users subscribe to the same MNO and get served by the same HN. The UEs are in cellular coverage of the BS, D2D communications between two UEs are controlled by gNB. The MNO controls the user's access to the network, initial connectivity establishment, resource allocation,

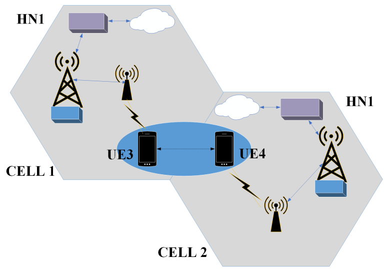


Figure 4.4: Scenario 2 Intra Network Operators and in-Coverage

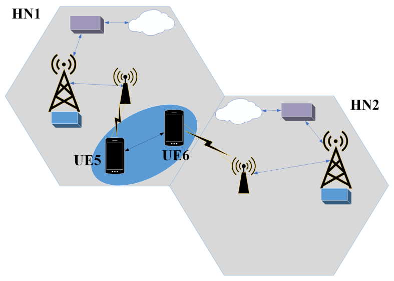


Figure 4.5: Scenario 3 Inter Operators, in-Coverage and Roaming

and security management. The cellular licensed spectrum is shared between the established D2D link with the normal cellular connections under the coordination of MNO.

Scenario 2: *Intra Operator, Non-Roaming and Different Cells*
UE3 sharing content with UE4 in HN, served by the same operator but in different cells, as shown in Fig. 4.4. This is an intra-operator and non-roaming scenario, both users subscribe to the same operators, and they are in coverage.

Scenario 3: *Inter Operators and Roaming*
UE5 sharing content with UE6, UE5 in HN while UE6 is roaming in VN, one SN, and two HN. This is an inter-operator and roaming scenario, both users subscribe to different operators, and UE5 is served by HN and UE6 served by VN in coverage as shown in Fig. 4.5. D2D communication is established through the D2D link but the HN of UE5 can restrict UE5 from sharing content with UE6 by hiding its service visibility or denying access through ACL mechanism and preventing its services from being accessed by UE6. However, when static roaming agreements between VN and HN are not applicable then dynamic roaming would be used to permit access.

Scenario 4: *Inter Operators and out-of-Coverage*
UE7 sharing content with UE8, both UEs are out-of-coverage and need to share content without involving their HNs. This is an inter-operator; it also applies to emergency or disaster situations. Both users, UE7 and UE8 subscribe to different operators as shown in Fig. 4.6.

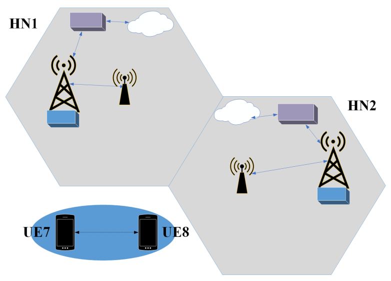


Figure 4.6: Scenario 4 Inter Operators and out-of-Coverage

4.2.5 Analysis

The system model identifies entities that are involved in the access and delivery of services, while the access and delivery model describes the service request by the UE and how the request is handled by the HN/SP and content dissemination back to the requester. The application scenarios discussion entails how D2D users communicate and share content with network assistance or with stand-alone D2D communications. This next section investigates the threats and vulnerabilities of security and privacy in 5G enabled D2D communications and service-oriented networks.

4.3 Security and Privacy for NSD in 5G enabled D2D communications Network

Security and privacy are interrelated issues that will be serious concerns in 5G enabled D2D communications. The UE can participate in D2D based content transmission, distribution, and delivery as well as traffic offloading, which increases the vulnerabilities of participating entities and the data in transit. Moreover, mobile security at edge is another concern, where the services and user's data will be vulnerable. The MNO must authenticate and validate the SP to ensure secure service access and provisioning. If not, it might compromise the security of HN entities and expose them outside the HN (5GPPP 2017). Therefore, security mechanisms must be in place to protect the network, UE, and services from any possible threats. This section explores the security threats and vulnerabilities that might affect NSD in 5G enabled D2D Communications, some of these issues have been discussed in (Haus et al. 2017), (Gandotra et al. 2017), (AbdAllah et al. 2015), (Tourani et al. 2018).

4.3.1 Security Threats

The wireless nature of D2D communications plus its characteristics and architecture presents several security vulnerabilities that put the network at risk to potential threats (3GPP 2010). 5G can still be affected by threats from legacy systems and the security of NS could be compromised by new attacks on different levels of the network. In a cellular network, the BS acts as a central authority but in D2D communications it might play a minimal part hence strong anonymity is provided. However, BSs have access to the data transmitted between the UE and other UEs, which could expose the data to possible attacks in the form of active and passive, local, and extended. Based on our system model in Fig. 4.1, some of the attacks are as follows.

Eavesdropping: D2D messages can be eavesdropped on by authorized and unauthorized cellular users, affecting data confidentiality (Haus et al. 2017). Side channel attacks can be used across network slices targeting the implementation of cryptography or running a code to influence the contents of the cache (Alliance 2016).

Data Fabrication: The unprotected transmitted data can be fabricated or changed by malicious users, which leads to the content being circulated by the unaware infected device to other devices, affecting the integrity of the system.

Control Data Attack: An attacker might try to modify the control data; cryptographic techniques can be used to prevent such attacks.

Impersonation Attack: A legitimate user might be impersonated by a malicious user and communicate to other D2D users through an ID impersonation attack. Network slice instances are vulnerable to such attacks, which could lead to other attacks like location attack (Alliance 2016).

Free-Riding Attack: In D2D communications UEs participate in sending and receiving data willingly but some UEs might not be willing to send data to other UEs if they are in power-saving mode while receiving data from their peers, which decreases the system availability.

Privacy Violation: Privacy affects both D2D communications and CCN, it is important to preserve the privacy of users' data such as their ID and location from being leaked. User's privacy could also be violated if sensitive data in transit is eavesdropped on. If an attacker can listen and intercept transmitted messages, she would be able to extract information and guess the location of the UE. ICN cached content, user privacy, and content names are all targets of privacy attacks. Moreover, a user's subscription information could be leaked by an attacker or a compromised publisher through timing and anonymity attacks (Tourani et al. 2018).

Denial of Service (DoS) Attack: D2D services might be interrupted by making them unavailable to the intended users, an attacker can weaken or block legitimate devices from establishing the connection completely. DoS attack can overload the NS by sending big continuous requests to CCN nodes, domain name queries, and by interest flooding (Tourani et al. 2018). An attacker might exhaust security resources in one network slice so that she can attack other slices (Alliance 2016).

Content Poisoning Attack: It involves filling the content router's cache with invalid content, the content being injected has a valid name matching the sent interest, however, the payload might be fake or have an invalid signature (Ghali et al. 2014*a*).

Cache Pollution Attack: An attacker may weaken the caching activity by requesting more frequently the less popular content with attacks such as locality disruption and false locality (Tourani et al. 2018).

Unauthorized access: An unauthorized device might access some data which was for a specific entity. For instance, in unauthorized cache access, a cached data object from a local device might be access by an unauthorized device (Loo & Aiash 2015).

Cache Misuse: The attacker can utilize on caches capability and use it as storage, hence, enabling her to make their own content available. Also, an attacker can corrupt the cached content turning it into incorrect returned objects for a DoS attack.

False Accusation: A malicious publisher may try to make it look like the requesting device has requested a data object when it has not and might also charge a subscriber for a service that was not requested or obtained.

IP and Location Spoofing: Attackers use malicious code to manipulate the header of IP packets (Lichtman et al. 2018). Might also send fake location information disrupting the D2D connectivity establishment by imitating with artificial locations to confuse the D2D members.

Session Hijacking: The attacker spoofs the IP address of the victim device and guesses the SQN expected by the targeted source device, this is followed by Distributed DoS (DDoS) attack on the victim's device, impersonating the device to carry on the session with the targeted device.

Communication Monitoring: The attacker may have access to the same router the requester is using to receive content then she targets a requester and tries to identify the victim's requested content.

Jamming Attack: Malicious user masquerading as a legitimate subscriber on a shared link, sends many content requests to disrupt the flow of information, and then replies are sent to a destination other than the requester. In 5G jamming is achieved through analysing physical control link channels and signals (Lichtman et al. 2018).

Data Leakage: The UE might be attached to several slices on the network with different security parameters. If the UE cannot separate data from different slices, the separation between

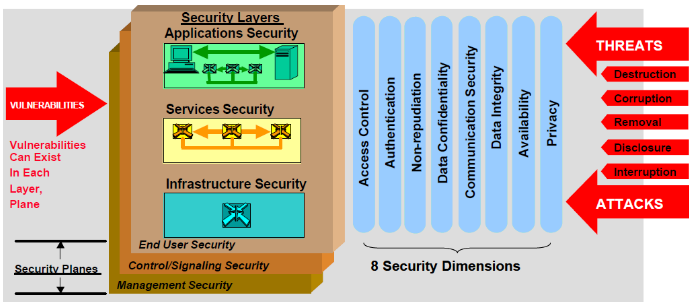


Figure 4.7: X.805 Security Framework (Zeltsan, 2005)

slices could decrease, leading to the UE receiving sensitive data from one slice and then publish that data via another slice (Alliance 2016).

4.4 Security Evaluation of NSD in 5G enabled D2D Communications

To give a systematic approach of the security evaluation for 5G enabled D2D communications, this section applies the X.805 framework (Zeltsan 2005) to analyse the security requirements for delivering NS based on our system model. The NS abstraction is used to map NS with X.805 security layers. The framework was used in (Park & Park 2007), (Loo & Aiash 2015) to evaluate E2E security of systems.

4.4.1 The X.805 Security Framework Overview

Analysing security in any networking system is very complicated, ITU-T developed X.805 standard as a security evaluating tool, it uses a modular method to create a multi-layered framework that assesses possible threats and vulnerabilities in E2E network security based on the eight security dimensions to address security threats effectively. The standard defined three security layers (applications, services, and infrastructure); three security planes (end user, control, and management); and eight security dimensions (access control, authentication, non-repudiation, data confidentiality, communication security, data integrity, availability, and privacy) (Zeltsan 2005). The security layers and security planes are identified according to the network activities as illustrated in Fig. 4.7. In addition, nine security viewpoints are created by applying each security plane to each security layer, whereby each viewpoint has its own distinctive vulnerabilities and threats based on security dimensions.

Security Layers: Infrastructure security layer covers the core building blocks of NS, NF, network slices, applications, and individual communication links such as BSs, RRH, routers, servers, slices, and fibre links. This layer facilitates the security of hosts involved in the data transmission, it prevents attacks from air interfaces and physical links including content servers, gateways, BBU, and D2D connectivity. Service security layer covers services provided to end users such as CCN, IP, cellular, QoS, and location services. Securing this layer is complicated as services may build upon one another to satisfy user requirements such as sharing and delivery of CCN services via

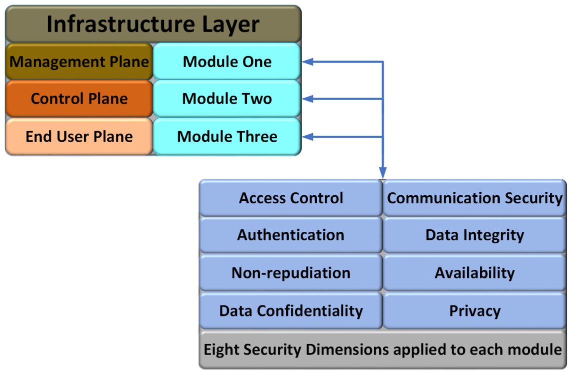


Figure 4.8: Infrastructure Layer with Eight Dimensions

D2D communications. The CCN is related to the service layer while D2D communication is related to the infrastructure layer.

Security Planes: The security plane is concerned with the security of operations and provisioning of the individual mobile network elements, D2D, and cellular communication links as well as securing the functions of NS such as the configuration of UE, BBU, 5GC, and secure content provisioning. In addition, it is concerned with the security of the control data stored on the network elements and in transit for NS such as D2D control link and PDU session. It gives assurances on the security of the user's data on the network elements.

Security Dimensions: The eight security dimensions are used as a viewpoint for vulnerabilities and threats to provide protection against any attack in the form of security controls on each layer, this aligns with 5G SCC.

4.4.2 Security Evaluation using X.805 Framework

To be able to mitigate potential threats and attacks as presented in section 4.3, the security requirements must be evaluated. The security requirements are classified using security layers, associated with the security planes in modular format, each module is analysed using the eight security dimensions. The focus is on the service and infrastructure layers and how the UE is affected when accessing the NS. Modules 1, 2, and 3 are associated with the infrastructure layer whereas modules 4, 5, and 6 are associated with the service layer. The X.805 demonstrates a methodical approach in modular form as shown in Fig. 4.8 and Fig. 4.9. The security goal is to cover the security capability of the framework including the detection and recognition of attacks, protection, audit of the system, and its recovery after the attack. Next is the comprehensive security requirements analysis:

Access Control

This dimension limits and controls access to network elements and services through encryption and authorization mechanisms. Some services lack built-in support to provide ACL. When an entity that is not controlled by the SP publishes content, the SP has no way of applying ACL or knowing which user has accessed the data (Ohlman et al. 2014). Also, the system should be able to revoke the user's privilege if it is detected to be malicious. Attacks such as free-riding should be prevented,

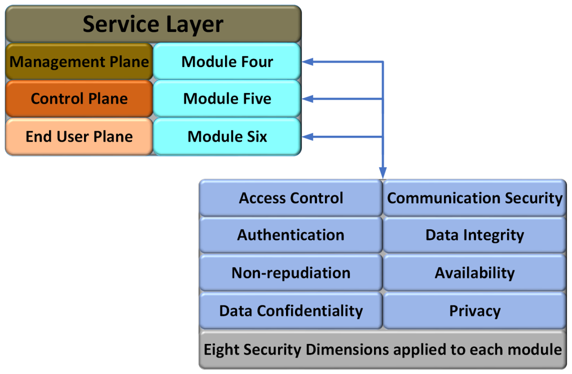


Figure 4.9: Service Layer with Eight Dimensions

and UE should be protected from joining rouge BS. Additionally, the privilege of D2D users should be taken away in time if a user is found out to be malicious or their subscription has expired. Moreover, revocability in D2D communications can be applied to avoid impersonation attacks.

Infrastructure Security Layer - *Modules 1, 2 and 3*: The ACL at this layer is concerned with only allowing authorized UE and network entities to perform activities such as accessing data on the UE and content objects in the network. Without the appropriate access control policies, an unauthorized device might be able to access services that were intended for limited UEs. ACL can permit or deny a SP or a UE the right to perform any action on service or UE during the D2D communications and extra mechanisms should be in place for the UE to control their data flow. ACL must be applied to control the access of data on the network entities.

Service Security Layer - *Modules 4, 5 and 6*: The ACL at this layer is concerned with allowing only authorized BBU and SP to perform management activities of the NS and that the received service originated from an authorized source. In addition, only authorized UEs can access the NS and the services request message originated from an authorized UE before being accepted. The BBU should be able to hide its services or visibility from unauthorized UE after the primary authentication and during handover sessions. ACL must be applied on the BBU and content server such as Role Based Access Control (RBAC) or Discretionary Access Control (DAC) depending on the ID of the subject for authorization, it suits unstructured domains like the proposed system model in this research.

Authentication

This dimension ensures a valid proof of ID is presented in the form of a shared secret, public key, digital signature, or digital certificate (Ahlgren et al. 2012). Authentication evaluates the ID of the entity and verifies if the entity has a secret and private key, it can be applied to an entity and data. Assigning an ID to a secret or key is required during authentication. In a cellular network for the UE to access the NS the UE and network must perform primary authentication using AKA method, to stop attacks such as impersonation attacks, false content injection, and free riding. After a successful primary authentication procedure, a secondary authentication can be performed to ensure that only authorized UEs can access SP services (3GPP 2020*f*).

Infrastructure Security Layer- *Modules 1, 2 and 3*: Authentication at this layer is concerned with verifying the ID of the UE requesting the services, SN, BBU, and content server providing the NS to the UE. The ID of the communicating entities and transmitted data must be authenticated to secure D2D communications. The first step is to authenticate the entities to confirm the D2D peers' IDs then authenticate the data source to confirm if it is from a legitimate user. If device UE_A receives a message from device UE_B, UE_A can verify that UE_B is indeed the sender of the message. For instance, verifying the signature of another UE.

Service Security Layer- *Modules 4, 5 and 6*: Authentication on this layer is concerned with verifying the ID of services, the service entities, and the origin of the NS. The verification of UE trying to access the services should be done by Authentication Authorization Accountability (AAA) servers, which also monitor, manage the subscription and service provisioned to the UE. With service authentication, the external service should also be able to authenticate the UE. The receiver should be able to assess the validity, provenance, and relevance of the data received (Ohlman et al. 2014), to make sure that fragmented data received is complete and not corrupted. Therefore, verifying the producer's identification to ensure that ID and source of cached data can be trusted is a must (Ahlgren et al. 2012).

Non-Repudiation

This dimension is concerned with preventing any device from denying its involvement in an activity on the network. It prevents entities from denying transmission or reception of a service. It also allows the tracking of the source of a possible security violation (Abd-Elrahman et al. 2015). The MNO/SP and devices should be held accountable for their action through monitoring of network activities. For example, a verified content producer should not be able to deny that they are the source of the content or UE should not be able to deny sending an interest message. A digital signature can be used to achieve non-repudiation and other mechanisms should be in place to prove the originality of the data as well as proof of transaction to prevent false accusation attacks.

Infrastructure Security Layer - *Modules 1, 2, and 3*: The non-repudiation at this layer records and identifies entities such as UE, BBU, servers that perform activities on other devices, modify control data, or access UE data in case they deny their involvement in such transactions. The record acts as proof of access or alteration of the control data as well as identifying the origin of control messages and the action that was performed. For example, identifiers can be applied as a solution to bind user related messages to the UE and network for accountability. Packet level authentication (Andersen et al. 2007) can be used to provide network layer authentication and accountability of the data using public key cryptography.

Service Security Layer - *Modules 4, 5, and 6*: The non-repudiation at this layer is concerned with recording of the content producer, UE, BBU, or 5GC entities that performed NS activities, the origin of the control message, and the UE that accessed the services. For example, information about an object's provenance, which indicates who generated or published what content object by name. In addition, the requester who receives content is recorded and charged for the service, the use of out-of-band digital signature (Mao 2004) solves this issue. There should be a strong association between the entity IDs and the use of the NS to prevent spoofing attacks. However, there might not be a direct link between SP and the requester despite the need for auditing of usage. Therefore, the producer and requester must trust the system to account for usage fairly, charges are added according to services accessed periodically (Aiash et al. 2014). The mobile network is capable of tracking and monitoring system utilization and usage as per the service contract.

Data Confidentiality

This dimension ensures the confidentiality of data on the network devices and in transit, encryption should be used to provide confidentiality. Data must be encrypted and should only be decrypted by an authenticated and authorized entity. The confidentiality of the data must be protected against attacks such as eavesdropping and privacy invasion. Moreover, encrypting messages on a wireless channel is standardized in 5G. For example, encryption keys can be applied to encrypt data before transmission using symmetric or asymmetric encryption mechanisms.

Infrastructure Security Layer - *Modules 1, 2, and 3*: The data confidentiality at this layer deals with the protection of the data on network devices, control data, and data transiting the network devices. For example, user's control and configuration data should be protected against any unauthorized access. During service provisioning, data might pass through possibly untrusted segments. Encryption and ACL can be used to provide data confidentiality. Other methods based on Channel State Information (CSI) could be used to avoid data leakage (Haus et al. 2017). Additional use of cryptographic mechanisms like stream ciphers might stop the attacker from reading messages between two D2D users as well as preventing eavesdropping attacks.

Service Security Layer - *Modules 4, 5, and 6*: Confidentiality at this layer is concerned with protecting the NS's control, configuration, and management data such as PIT updates, security settings in network entities, and UE data from unauthorized access and eavesdropping attack. Also, encryption and ACL can be used to provide confidentiality of NS. A content producer should be able to control which subscribers may receive what content in out-of-coverage scenario, however, confidentiality might not be relevant where the producer is offering data to everyone. Group key distribution (Liu et al. 2014) can be used for data confidentiality, whereby the producer pre-distributes keys to all potential requesters. Although an out-of-band approach prearrangements between the publishers and the subscribers might be required (Loo & Aiash 2015).

Communication Security

This dimension ensures that information only flows from source to destination using secure channels. Security is achieved by creating a secure tunnel between the endpoints. In legacy systems, the wireless communication channel was not secured but in 5G this problem will be addressed by concealing SUPI (3GPP 2020*f*). However, point-to-point communication does not apply to ICN, the content is requested without being aware of its location. Moreover, the requester might be receiving chunks of cached data from different sources such as, SP, BBU Pool, and other UEs which makes establishing secure channel complex and unmanageable. Therefore, information and host-centric security methods must be considered to protect intended D2D users when receiving and reading data. Encryption methods could be used for secure routing and transmission of data to authorized users as well as physical layer security by exploiting wireless channel characteristics, modulation, coding, and multiple antennas preventing eavesdroppers (Sun & Du 2017).

Infrastructure Security Layer - *Modules 1, 2, and 3*: The communication security at this layer ensures that UE, control, and management data only flows between entities and communication link that uses secure channels. For example, authentication data such as security context should not be diverted or intercepted as it flows between the source and intended target. Secure communication must be established between the UEs and other entities before sharing any information.

Service Security Layer - *Modules 4, 5, and 6*: The communication security at this layer ensures the management, control, and UE data in transit for NS, only flows between entities using a secure channel, and that data is not intercepted. For example, with the interest messages and service data, the SP registers a service ID to the server or cache node and binds the data under a namespace. The BBU is tasked with monitoring and storing data regarding the interest and data exchange, the BBU can search for malicious devices and select a different route for packets to reach their destination securely, the ICN architecture can reveal misbehaving devices (Priya & Sakthisaravanan 2015).

Data Integrity

This dimension ensures that data is received as sent or retrieved as stored and no data manipulation has been performed by any malicious or authorized users. The compromised data can affect the integrity of the whole network, D2D users should be able to receive correct data without alteration or fabrication. If an attack like message injection or false reporting is initiated, the data's integrity might be violated which could compromise the UE. The integrity of the communication between the UEs and the 5GC entities must be protected, this can be achieved by using hash functions, digital signature, and manifest-based content authentication for content signing.

Infrastructure Security Layer - *Modules 1, 2, and 3*: The integrity at this layer is concerned with protecting the configuration and control data, D2D links, data transiting from the UE and other entities against unauthorized modification or replication, and the data stored on the devices. System integrity can be compromised if an attacker puts bogus subscription data using a malicious server and behave as a bogus subscriber to the UE or BBU, respond to interest with a bogus reply or drop the data completely. Protecting the integrity of legitimate user's data can be achieved with the use of MAC. **Service Security Layer -** *Modules 4, 5, and 6*: The integrity at this layer is concerned with protecting the management and control data of NS against unauthorized modification and deletion. For example, the integrity of interest and service data in transits should be protected as well as the integrity of identifications and security context from authentication procedures. To provide integrity, cryptographic mechanisms such as hash functions and MAC should be applied, however, 5G has no cryptographic integrity protection for the user data plane. Also, integrity protection mechanisms should be applied to ensure that any data modification is detectable.

Availability

This dimension ensures network elements and services are available to legitimate users, authorized users should be able to access the network and services ubiquitously. Network access and services should be available even during any attacks like DoS and free riding. DoS attacks are hard to detect since D2D communication does not rely on centralized infrastructure (Huang et al. 2011), while jamming attack affects communication between D2D users and can be started anonymously (Haus et al. 2017) which can make services unavailable for users. However, due to CCN naturally spreading contents to enable requests being satisfied by alternating sources, it requires a lot of effort to initiate a DoS attack which reduces the impact of the attack, even though it is hard for an attacker to send repeated requests on a single device but it is possible. In 5G, NS should always be available for UEs, and the waiting time to connect or get services should be as short as possible to complement 5G objectives like high data rate and reliability. In addition, devices such as intrusion detection, intrusion prevention systems, and firewalls should be deployed in the network as well as business continuity and disaster recovery plans should be in place to decrease downtime of 5G NS.

Infrastructure Security Layer - *Modules 1, 2, and 3*: The availability on this layer is concerned with ensuring that network devices can receive control data, access UE data, and manage the D2D link. Also authorized personnel and devices should have access to the infrastructure. The network should be protected against DoS and jamming attacks. During a DoS attack, the attacker can target caching and routing planes by creating large amounts of unwanted traffic, which is cached by intermediate devices in the BBU and 5GC, resulting into cache overflow due to overloading of caching plane. Moreover, a DoS attack can be initiated by polluting the content cache in BBU and 5GC's CS, hence returning an incorrect content object (Edwall 2011). To minimize the damage from such an attack, data should not be delivered without a valid subscription from the requester. Therefore, preventing unwanted traffic from bogus requests can improve availability and dependability whereby the system only serves valid and authorized users.

Service Security Layer - *Modules 4, 5, and 6*: The availability at this layer is concerned with ensuring that the network devices participating in a NS provisioning are always available to exchange control data. To ensure that accessing and managing the NS by authorized users cannot be denied. Services must be protected from DoS and jamming attacks. The use of malicious content and subscriptions to overload the system or a subscriber from flooding the producer with

Table 4.1: Infrastructure Layer in Relations with Security Dimensions

Security Dimensions	**Infrastructure Layer**	**Security Mechanisms**
Access control	Authorize UE and network entities to accessing data on the UE and other entities	ACL and passwords
Authentication	Verify the ID of the UE, BBU and server providing the NS to the UE	shared secret, PKI, digital certificate, and digital signature
Non-repudiation	Record UE, BBU, servers that perform activities on devices while accessing the NS	MAC, hash function, and asymmetric encryption
Data confidentiality	Protect the UE, control data and data on other network devices	symmetric and asymmetric encryption
Communication security	Ensures that UE, control and management data only flows between entities on sure channels	symmetric and asymmetric encryption
Data integrity	Protect the control data on the entities, links, and data in transit against unauthorized modification	MAC, hash function, and digital signature
Availability	Ensure that network devices can receive control data, access the UE data, and manage D2D links	intrusion prevention and detection systems, business continuity and disaster recovery plans
Privacy	Ensure that data that can identify the UE and other entities are not available to unauthorized users	symmetric and asymmetric encryption

bogus interest messages should be prevented. Availability of NS guarantees that authorized users can access the services through D2D communications. Moreover, availability and dependability can maintain a satisfactory user experience and enforce the availability of services even during an attack.

Privacy

This dimension ensures that the identifiers, users, and network data are kept private. Privacy in 5G is part of very critical security requirements, elements such as ID and location must be preserved. This is central to the new age of information and users' data privacy has become a very sensitive topic in recent years. D2D users should know what data they are sharing, and the system should collect only the required data for specific services. Encryption can be used to protect the communication and data transmission between entities.

Infrastructure Security Layer - *Modules 1, 2, and 3*: The privacy at this layer is concerned with ensuring that data that can be used to identify the UE, BBU, and 5GC entities or communications link is not available to unauthorized users. Network elements should not be able to provide data revealing the UE's network activities such as UE's location to the unauthorized user and only certain user data should be accessed by authorized personnel. Exposing information from cached data could lead to exposing UE data. The UE might need to communicate anonymously while accessing services to reduce such attacks.

Service Security Layer - *Modules 4, 5, and 6*: The privacy at this layer is concerned with ensuring that data that can be used to identify the NS management systems and communication links is not available to an unauthorized user. In addition, NS should not be able to reveal data such as UE and service IDs. An attacker might be able to gather and obtain data by monitoring the cache transaction even when the requester source is not clearly identified, by analysing the direction of the requests and timings of the transactions (Loo & Aiash 2015). With location-based services being provided to the UE, tracking of the UE is possible and UE privacy might be compromised if the attacker manages to get into the network. Moreover, the UE activities can be exposed to cache owners that they might have no transactions with the UE. It is impossible for the user to request services without revealing their subscription and security information to the SP or the infrastructure. Private information retrieval mechanism (Yi et al. 2013) could be used to preserve the privacy of subscription data, it allows the retrieval of database entries without the user disclosing the entries to the server. The Tables 4.1 and 4.2, summarise security requirements classification based on security layers, security planes associated with the eight security dimensions.

4.5 Security Solution Approaches

With the threat modelling and security evaluation completed, next is to explore possible solution approaches, which must consider the unique characteristics and pre-design security features of 5G, D2D communications, and CCN. For instance, CCN has basic security mechanisms as part of its architectural design such as integrity and authentication while the encryption messages over wireless communication have been standardized and trust enhanced in 5G. In addition, cryptography and ACLs can be deployed at the different layers to achieve security objectives.

Table 4.2: Service Layer in Relations with Security Dimensions

Security Dimensions	**Service Layer**	**Security Mechanisms**
Access Control	Authorize BBU and SP to perform management activities on NS	ACL and passwords
Authentication	Verify the ID of NS, the service entities and the origin of the NS	shared secret, PKI, digital, certificate, and digital signature
Non-repudiation	Record the SP, UE, and BBU transactions and origin of the control messages to prevent deniability	MAC, hash function, and asymmetric encryption
Data confidentiality	Protect the NS's data transiting the network devices from unauthorized access	symmetric and asymmetric encryption
Communication security	Ensures the management, control and UE data for NS passes through a secure channel	symmetric and asymmetric encryption
Data integrity	Protect the management, control, the UE data against unauthorized and deletion of service data	MAC, hash function, and digital signature
Availability	Ensure that the network devices, UE data and D2D links are available to to receive control data	intrusion prevention and detection systems, business continuity and disaster recovery plans
Privacy	Ensure that data that can be used to identify NS is not available to unauthorized users	symmetric and asymmetric encryption

4.5.1 Authentication and Key Management

Authentication is a key factor in securing D2D communications, content delivery, and facilitating content authenticity to resist various attacks. Security frameworks and authentication protocols can be used to perform authentication and secure communications. Whereas data origin authentication can be achieved by using a digital signature for proof of origin and protecting sensitive messages from being tampered with. Cryptographic keys are used to encrypt and decrypt data; therefore, key management plays a vital role in the preservation of data and transmission between D2D devices. This includes the generation, distribution, and storage of keys.

4.5.2 Confidentiality and Integrity

Data confidentiality of NS, D2D messages, and control data can be implemented by using ACLs and cryptographic techniques. Data integrity can be protected by using hash functions and digital signatures, preventing a malicious user from forging data. In CCN, routing misuse is the result of the concept of trustworthiness and integrity based on a trusted computing approach (Anderson 2004).

4.5.3 Non-Repudiation Enforcement

The use of digital signatures and certificates can act as proof of work so that entities do not deny their involvement in transactions. The UE must register to SP for services, whereby the SLA and QoS are agreed upon. Then SP can monitor the services accessed and bill the subscriber accordingly. An efficient auditing system can be used to stop attacks by logging all activities in the network such as distributed audit service platforms (Group 1998). The system should be able to identify the origin of the false message through traceability. The message originator can be verified through the authentication process to avoid data leakage by false notifications from a malicious user.

4.5.4 Secure Naming, Routing, Forwarding and Transmission

The content naming technique is fundamental to ICN, binding a content name and its producer prevents content poisoning attacks. Through secure naming, content source verification is affirmed, secure routing and forwarding techniques are also vital to any network security. Secure naming scheme can be achieved by using RSA and IBC (Tourani et al. 2018), whereas secure routing is essential in D2D communications for protecting relayed messages (Panaousis et al. 2014). Additionally, a secure forward plane and a secure namespace mapping provide secure forwarding, by binding the producer's public key and the content name with the interest packet (Ghali et al. 2014*b*), mapping the producer with their content.

4.5.5 Access Control Mechanism

Authorization enables SP to control the access to its services requested by other entities using Remote Authentication Dial-In User Service (RADIUS) (Lior & DeKok 2013) and diameter (Arkko et al. 2012) protocols which are centralized authorization or through a distributed authorization method (Woo & Lam 1998). This is achieved by applying fine-grained, encryption, attribute-based ACLs, and context-aware schemes, which can be supported by the authentication mechanism to provide different levels of authorization such as access to the network, resources, and services.

4.5.6 Privacy Preservation

The UE might be required to disclose their location for data routing and forwarding but UE might be unwilling to share their location to avoid privacy violation hence slowing content dissemination. Privacy can be preserved by using ID expiration enforcement technique based on homomorphic cryptography (Dijk et al. 2010). In addition, data encryption and anonymity schemes can be applied to improve publisher's and consumer's privacy (Martinez-Julia & Gomez-Skarmeta 2012). Moreover, other methods include physical layer security that defines secrecy capacity at which unintended users cannot decode transmitted message (Leung-Yan-Cheong & Hellman 1978) and obfuscation that degrades the quality of information like UE location.

4.5.7 Availability and Dependability

This approach reduces the defect attacks such as DoS and free riding that make services unavailable to legitimate users. A cooperative mechanism between UEs can be used to prevent selfishness among UE participating in D2D content sharing. Other DoS mitigation may include a change in

Table 4.3: Threats, Attacks and Solutions based on X.805 Framework

Threats	**Attacks**	**Solutions**
Destruction of data and resources	Impersonation, man in the middle, routing, misuse	authentication, confidentiality and integrity
Corruption or modification of data	Content poisoning, false accusation, cache misuse, data fabrication, replay, IP/location spoofing	Non-repudiation, secure naming, authentication
Theft, removal or loss of data resources	Unauthorized, access masquerading, false content injection, data leakage	ACL, message and entity authentication
Disclosure of data	Privacy violation, eavesdropping discovery, monitoring, timing anonymity, unlinkability, traceability	ACL message and entity authentication
Interruption of services	Cache pollution, DoS, free riding jamming, session hijacking interest flooding	ACL, authentication, availability

intermediate cache structure reducing the rate of consumer request through a request of proof work (Tourani et al. 2018) and interest flooding detection method (Wang et al. 2014).

4.5.8 Analysis

The related work solutions can be used on different layers of the network to provide security. In addition, unconventional techniques such as strong cache verification and self-certifying naming methods can prevent forged content. Security on the physical layer enforces security on the upper layers in D2D communications and it is necessary to study network security on the upper layers basing on building security protocols to provide secure communication without undermining D2D communications and 5G functionalities. Therefore, the security threats in 5G require an integrated solution using a hybrid approach that consists of information-centric and communication-centric techniques.

Some of the threats, attacks, and possible solutions are presented in Table 4.3. There is no solution comprehensive enough to cover all 5G security domains and requirements of 5G. An integrated solution should consider threats from legacy systems and 5G new use cases. Moreover, the security mechanisms should be lightweight to avoid high communication, computational overhead, and the negative effect of mobility on security to minimize the burden on the cellular network.

4.6 Summary

This chapter introduced the NSD framework in section 4.2 that proposed a system model and defined the network entities which represent the parties involved in the proposed security protocols in the next chapters. However, before the security protocols are developed, an investigation on the security and privacy along with threats and vulnerabilities of the system model was conducted in section 4.3. Section 4.4 evaluated the security requirements comprehensively using a systematic approach based on X.085 security framework. In addition, the existing approaches and possible new approaches were discussed in section 4.5. Which highlighted the lack of an integrated approach to address security in 5G enabled D2D communications. Therefore, the next chapter 5 introduces the proposed security framework that intends to address the highlighted security issues in this chapter.

Chapter 5

Network Services Security Framework

5.1 Introduction

The general security approach in addressing security issues is the use of cryptography techniques to achieve most security objectives. Cryptographic techniques should increase security reliability in 5G enabled D2D communications, in form of anonymity, unlinkability, privacy, confidentiality, integrity, and authentication. These mechanisms should be lightweight due to the computation and energy consumption constraints in mobile devices. In the past D2D security was mostly applied at the application layer but recently network layer and physical layer security have been applied to achieve security objectives. Whereby communication secrecy can be achieved at the lower layers without depending on higher layer encryption. Additionally, physical layer security can also be achieved by analysing and applying physical characteristics of D2D wireless channels to achieve secrecy capacity, channel-based key agreement, physical-layer authentication, and privacy preserving anonymity (Zhang & Lin 2017). This chapter presents the proposed security framework, model, and protocols for NSD in 5G enabled D2D communications.

Some features in CCN provide some security, for instance, a consumer can achieve trust from the received content, derived from the credential of the publisher using their public key certificate (Zhang et al. 2011). Data integrity is provided intrinsically in CCN due to NDO names containing a hash of the data formatting, integrity is achieved by a publisher signing a content with a private key and verified by using publishers' public key. Data confidentiality is achieved by using symmetric encryption and secure key distribution to legitimate users. As mentioned earlier, DoS has less effect on the CCN because of its name-based routing, as the interests are spread all over the network and the request are aggregated. Security can be addressed on a basic level with secure naming, cache verification, and self-certifying naming methods. Self-certifying is a popular method in 5G due to its ability to handle dynamic content objects and providing an efficient content retrieval implementing one of 5G objectives (Tourani et al. 2018). Some of the work in this chapter is also presented in (Edris et al. Submitted for Publication*b*).

This chapter is structured as follows. Section 5.2 presents an overview of the proposed security framework. Security modelling of the security framework is introduced in section 5.3. This chapter is summarised in section 5.4.

5.2 Proposed Security Framework

According to security investigation and evaluation carried out in chapter 4, the security issues can be addressed using infrastructural and information-centric security mechanisms to protect devices, communication channels, and the NS. BSs assist in establishing D2D connectivity and the distribution of security data like keys and digital certificates, which extends the decentralized security methods into D2D architecture however the BS acts as the trust authority. An approach that integrates both infrastructural and information-centric security services is proposed, the hybrid security framework will focus on the following:

- Information-centric security services: providing data confidentiality, integrity, and availability.
- Infrastructural-centric security services: providing entities authentication, ACLs to the user of network and services.

The proposed NSS framework assumes that network access security has been achieved. This research's main concern is the secure access of services by the UE and sharing between other UEs. The UE should be able to get authenticated and authorized to access, share data hence achieving 5G service security. The verification of authenticity, integrity, and provenance of the NDO must be done prior to the UE being granted access. Another concern is that whether the right data is being published and can be restricted even during out-of-coverage scenario. The UE should be able to share data without involving the HN evening during in coverage scenario, which addresses D2D service security.

After a successful primary authentication with the network, the UE requests access to services of SP via HN, the SP verifies the UE and grants access to UE. The security is implemented by various security mechanisms which should be interoperable with each other. Before addressing security on different levels, the security model must be defined, and its functions specified at each level.

5.3 Security Modelling

The security model enables the UE to access NS securely from the HN and SP at different levels. The 5G root certificate is stored in the USIM and ARPF, 3GPP standard (3GPP 2020*f*) specifies that the UE and ARPF share long-term key and identifiers used in a challenge and response authentication procedure to generate a session key with SEAF as explained in chapter 6. The AUSF is responsible for the mutual authentication of the network and the UE, after the initial authentication the communication channel is left open with HN's gNB until it is disconnected. If the UE moves to a different gNB, re-authentication is required and the generated keys and security information might be reused during the handover handled by the ARPF in both the HN and VN, this falls out of the scope of this research. In this case, the UE gets access to the services after connecting to the network, this can be supported further by D2D communications and the CCN which are host and information-centric respectively hence the hybrid approach of the security framework. Multiple methods are used to address the complexity of this system model as presented in the next chapters.

The security modelling in Fig. 5.1 leverages the security principles by 3GPP (3GPP 2020*f*). It applies the authentication and authorization methods that grant the UE access to services and

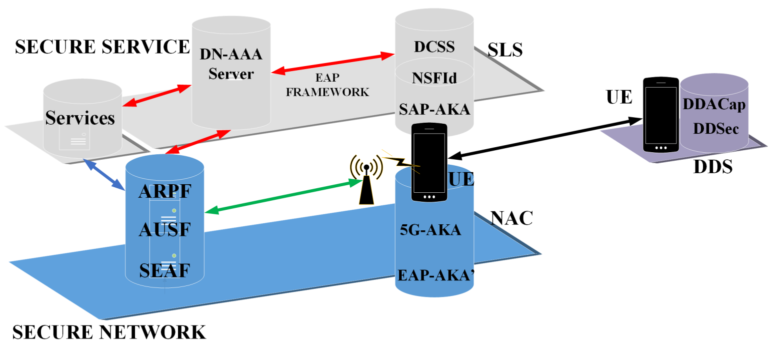


Figure 5.1: Security Model

permission to engage with other UEs. A multilevel framework is proposed to align with the NS abstraction, 5G protocol stack. It consists of Physical Layer Security (PLS), NAC, SLS, and DDS which also align with the NS abstraction in section 2.5. However, this security framework is only interested in the SLS and DDS, as PLS and the NAC have been studied extensively by related work. The PLS provides security for physical layer access, NAC for network access, SLS for services access, and DDS for D2D services sharing.

The framework considers the security link between all the security levels and the security entities, represented as a unified security model. This security framework intends to provide the UE with secure communication, access, and sharing of services with UE from another network without losing their initial network access and without the need of the SN or HN as the central authority in some scenarios.

5.3.1 Physical Layer Security (PLS)

This level of security is concerned with the PLS such as spectrum, resource allocation, interference, and signal. Even though PLS falls out of the scope of this study, a brief overview is given on PLS solutions applied by relying on the characteristics of wireless channels such as interference, signal, and fading. The quality of the attacker's signal can be degraded through keyless secure transmission by using signal design and processing methods. Different studies explored PLS in 5G, detailed reviews on different PLS techniques are presented (Sun & Du 2017), (Wu et al. 2018), (Gao et al. 2018). Techniques and approaches such as artificial noise injection improve channel quality, anti-eavesdropping signal methods align multiple user signals at the eavesdroppers. Secure beamforming enhances the spatial distribution properties of the transmitted signal that leads to the increase in the difference between legitimate users and eavesdroppers' channel quality.

5.3.2 Network Access Security (NAC)

The NAC has been well-defined in 3GPP's 5G security standards (3GPP 2020*f*), (3GPP 2020*g*), (Arkko et al. 2018) and studied in (Arkko et al. 2015), (Basin et al. 2018), (Fang et al. 2018). 3GPP specifies that for mobile subscribers to access the network securely via ng-RAN with their UEs, a primary authentication process is required to provide the NAC. Therefore, secure access is paramount to 5G principles, the security requirements have been defined in (3GPP 2020*f*) and supporting system architecture in (3GPP 2020*g*) to support 5G objectives.

3GPP standardized the security mechanisms that should protect UE and HN connection, subscribers and MNO expect security guarantees such as trust, authentication, confidentiality, and integrity from these mechanisms. The UE and the HN must achieve mutual authentication so that the UE can gain access to the network, services and perform other activities like connecting to SP via the DN function. The UE will be able to access the NS through a multilayer AP securely. The initial security is achieved by running AKA protocol between the UE, the SN, and HN, to achieve mutual authentication between the UE and the HN and generation of a session key for UE and SN secure communication. The 5G standard addresses the most critical security requirements of 5G, by defining two types of authentication procedures: primary and secondary authentication. The primary authentication can be achieved by using 5G-AKA and 5G EAP-AKA' protocols (3GPP 2020*f*).

The network-level authentication is responsible for verifying that the UE is accessing the right network. The 5G AKA protocol messages via radio interface are to be encrypted and implemented according to the security standard. The security context obtained after a successful authentication on this level is reserved for further security procedures on other levels when the UE wants to access other services. The security on this level is concerned with authentication and mobility of the UE whereby handover authentication and re-authentication of involved parties might be required. After this security assurance from NAC, the UE can request to access the other services from 5GC entities in the HN or from 3rd party SPs.

5.3.3 Service Level Security (SLS)

With SLS, the user is verified before being granted access to the services on this level, it checks if the right user is requesting access. As previously stated, this research is focussing on SLS due to new emerging services being promised by 5G, as PLS and NAC have been investigated extensively by related work. With 5G extending the capability of mobile networks, service provision is becoming more complex especially the security aspect. However, the NAC is recalled in the discussion as SLS might use some of the NAC's security context to achieve its objectives. SLS is concerned with service security in HN and SP, by providing authentication and authorization between the UE, MNO, and SP which gives the UE secure access to services provided by SP.

Since 5G is heterogeneous in nature, some of the studies on service security are still relevant to this research. The Service Level AKA Protocol for the mobile network in (Kuroda et al. 2004), (Aiash et al. 2014) addressed security for service access in legacy systems based on IP-based HetNets. After gaining access to the network the UE must be authorized to access the services, the UE and SP must mutually authenticate each other supported by HN through a secure communication channel provided by service level security protocols.

5.3.4 Device-to-Device Security (DDS)

Security for D2D communications in 4G was addressed to some extent as explored in chapter 4 but as explained in chapters 1 and 2 with 5G using D2D communications as an underlay technology that makes it crucial to 5G's functionality. DDS is concerned with service security and how the existing D2D security can be enhanced. After allowing the UE to access the network and services securely, how does the UE deal with the service accessed such as data? The D2D authentication

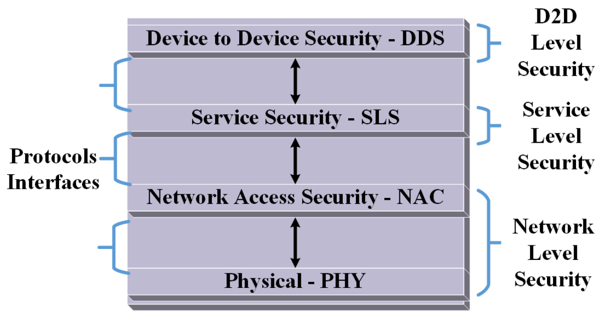


Figure 5.2: Security Framework

and authorization so far have been based on multiple security procedures, whereby every time it disconnects from the network it must get authenticated and authorized again. Moreover, now it is not possible to share the restricted data on UE in a no-coverage scenario. So, what happens to the data when it is on the UE and how can it be shared securely with or without network assistance. Therefore, DDS security will try to address these mentioned security issues.

With the security model defined, next is specifying security protocols for each level as defined in the proposed framework shown in Fig. 5.2, to address security issues faced by the system model as discussed in chapter 4. The framework includes four levels, however, this thesis focusses on three security levels i.e., NAC, SLS, DDS, and seven underlying security protocols, which are discussed in detail in chapters 6, 7 and 8, respectively:

- **Network Access Security**
  1. *5G-AKA Protocol*: To provide authentication and anchor key establishment between the UE and the HN.
  2. *5G EAP-AKA' Protocol*: To provide authentication and anchor key establishment between the UE and the HN.
- **Service Level Security**
  1. *SAP-AKA Protocol*: To provide authentication and anchor key establishment between the UE and the SP.
  2. *NS-FId Protocol*: To provide federated authentication and authorization between the UE and the SP.
  3. *DCSS Protocol*: To provide data caching and sharing security for the UE requesting authorization to cache and share the data accessed from Service Server (SS).
- **Device-to-Device Security**
  1. *DDSec Protocol*: To provide authentication and authorization to share the cached data between two UEs in proximity with network assistance.
  2. *DDACap Protocol*: To provide authentication and authorization to share the cached data between two UE in proximity without network assistance.

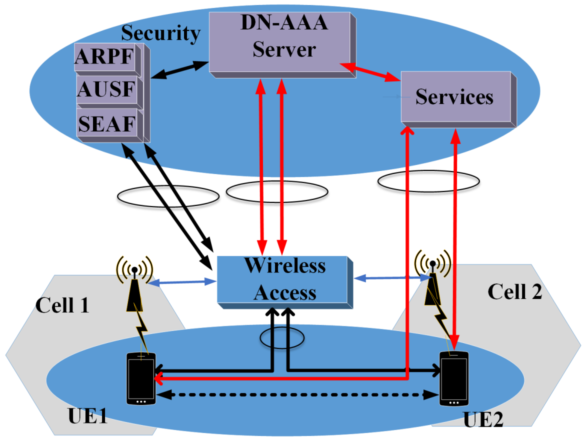


Figure 5.3: Unified Modular Architecture

5.3.5 Network Service Security Architecture

To address the security issues of network access and services, it requires a unified modular architecture as shown in Fig. 5.3 to support the proposed NSS framework. The NSS architecture adopts 5G security architecture entities shown in Fig. 5.4 and are responsible for security procedures such as primary and secondary authentication methods.

- UE: A combination of ME and USIM, it is the end user's device accessing the provisioned services.
- SEAF: Serves in the SN as the anchor for security in 5G, communicates with the AUSF to authenticate the UE and uses the anchor key for all the access scenarios with the UE.
- Security Context Management Function (SCMF): Retrieves the keys from SEAF and derive further keys for securing other communication within the 5GC.

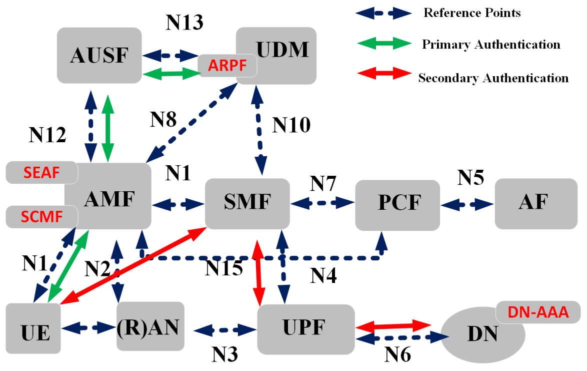


Figure 5.4: 5G Security Entities

- AUSF: An authentication server residing in MNO's HN, supports authentication for 3GPP access and untrusted non-3GPP access. It interacts with the SEAF to authenticate the UE.
- ARPF: A repository residing in a MNO's HN. stores key K, security credentials for UE authentication and computes cryptographic algorithms with UE security credentials as input to create AV. Together with the UDM they support unified authentication procedures with various technologies and allows security context sharing.
- UDM: Generates 3GPP AKA credentials for users, stores user identification data and manages the encrypting of SUPI and decrypting of the SUCI.

The NSS architecture also consists of the following security entities modified according to various security levels:

- HN AAA Servers: Consist of SEAF as the access network or pass-through authenticator, AUSF for authentication from the HN if the services are in a home-controlled environment. ARPF that stores security context from the network perspective that is used for primary authentication.
- External AAA servers: For authorization and secondary authentication in external DNs such as DN AAA. They authenticate and authorize the UE to access the service and share it with other UEs.
- SS: A server storing the content/services provided by the SP that the UE is trying to access.

5.3.6 Security Mechanisms

To achieve the security framework objectives, which are authentication, authorization of service access and data sharing in 5G enabled D2D communications network, multiple security techniques are deployed. The techniques apply cryptographic primitives such as symmetric and asymmetric encryption, one-way hash function, digital signature, and MAC, facilitated by ECIES.

MNO adopted a service authorization model that provides default services, whereby implicit authorization is applied to any subscribed service on a network for the registered UE to access it after a successful primary authentication. Service authorization in 4G was based on a static subscription of a user. In 5G, all UE authorization matrices are stored in the HN and then downloaded to the SN after UE authentication to the SN (3GPP 2020*f*). The downloaded authorization matrix of each UE is used to grant access services provisioned by the SN's SP.

The SP static authorization was able to facilitate interoperability viewpoint, to a business model with limited services provisioned by one or two MNOs. With multiple shareholders in 5G, the UE will be able to access unlimited service from the HN and multiple DNs, hence raising access and security concerns. In addition, the authentication procedure has been decoupled from the authorization process, introducing multiple security procedures as 5G will provide diverse services with network slices. Given the expected diversity of services provisioned and the number of end users, a new service model that supports service delivery through SP based authorization mechanisms to support the new on-demand multilevel and implicit service authorization is needed.

The authentication and authorization mechanisms used in this research adopt AKA and ACL methods to provide security in 5G enabled D2D communications. As presented in chapter 4, the security requirements that are being addressed in this research are aligned with security properties defined in chapter 3 of security mechanisms presented in chapters 6, 7 and 8. Summarised as secrecy, authentication, confidentiality, integrity, and privacy.

Authentication

As mentioned earlier, 5G standard (3GPP 2020*f*) specifies two authentication procedures i.e., primary authentication and secondary authentication based on EAP framework which is an important step for 5G to become an open network platform. The mutual authentication between the UE and the HN is facilitated under primary authentication. It is like that used in the legacy systems, however, in 5G the HN has been given more control during the authentication procedure. The authentication mechanism has an in-built home control application allowing the HN to know whether the UE is has been successfully authenticated to the SN, and take the final decision on authentication decision by agreeing with the message exchange and verification process or not. It is independent of radio access technology whereby it can also be applied to non-3GPP technologies such as IEEE 802.11.

Secondary authentication provides authentication between UE and DN outside the mobile operator domain based on EAP based methods and associated credentials. The UE gets authenticated by DN, establishing data path from MNO's network to DN assisted by HN SMF. The DN might be providing data services such as operator services, Internet access, and multimedia services. The DN function has been mapped onto the 3rd party domain in 5G architecture due to the secondary authentication provided by DN AAA servers. In another applicable scenario, the HN might provide infrastructure services via network slices to other MNO or SP, even when they are in the same network domain, but the service and security provision is handled by another party. Therefore, secondary authentication could be applied to internal DN (Ravindran 2019). The primary and secondary authentications are discussed in chapter 6 and chapter 7, respectively.

Mutual authentication is achieved when both parties confirm each other IDs and agree on a session key. The access security in ng-RAN and 5GC involves mutual authentication between the HN and UE, key derivation for access network, and other security procedures. It provides ciphering, integrity, and replay protection of signalling within 5G. The 5G system supports mutual AKA between the UE and SN, 5G-AKA or 5G EAP-AKA' are mandatory for 5G primary authentication methods, and the only authentication methods supported by UE and SN, for private networks EAP framework should be used as specified in (3GPP 2020*f*).

Authorization

Generally, a mobile network authorizes service access implicitly after authentication. ACL is used to implement permission and access rights to protect a service in the form of an object. When a subject wants access, the subject's name is checked against a list, if it is on the list then access is granted (Sandhu & Samarati 1994). Conventional ACL approaches to provide service access authorization to the system have been proposed in related work, RBAC, DAC, and Attribute-based Access Control (ABAC). Such ACL mechanisms sometimes require additional techniques like Encryption-based Access Control (EBAC) to provide a robust and efficient authorization to complex systems like HetNets. However, these ACL mechanisms are not capable of facilitating an integrated, manageable, and efficient authorization mechanism to support 5G NS provisioning, due to 5G characteristics.

RBAC provides user access authorization based on roles and supports least privilege and separation of duties. However, the use of RBAC based ACL becomes a problem with role explosion as the resources grow or the increase in the number of administrative domains. It complicates the interpretation of complex and heterogeneous 5G applications. ABAC ACLs policies directly associate attributes with subjects, authorization is based on user attributes, an access right is granted by

ACL authority to provide a fine-grained ACL. However, attributes specification and policy management in a domain or across different domains get complicated as the number of devices increases. Hence, not suitable for large-scale distributed 5G enabled D2D communications mobile networks. Capability-based Access Control (CBAC) mechanisms have been proposed as promising solutions to 5G authorization complexity, with CBAC based authorization, an unforgeable token with a set of rights is used to grant access to a resource, with unique reference to an object (Dennis & Horn 1983). Each subject is associated with a capability list indicating what activity the subject is authorized to execute on the object and the access matrix is stored in the metadata of the object (Sandhu & Samarati 1994). The subject presents a capability to the resource owner to get access to an object, the capability is transferable and non-forgeable, its validity and access right authorization can be executed by local SPs or at the edge making it feasible for D2D communications. Although CBAC raises capability propagation and revocation problems in mobile network applications (Anggorojati et al. 2012).

With in-network caching, content objects may not always arrive from their original producer/SP, content security cannot be considered in the traditional mobile network model based on secure and wireless or point-to-point channels (Kuriharay et al. 2015). This implies that content must be encrypted to prevent unauthorized access, invalid disclosure, or modification by unauthorized parties using EBAC. The existing ACL mechanisms represent a good conceptualization of authorization methods for providing access permissions to services. All these ACL policies can be implemented independently or as an integrated ACL solution. Moreover, in (3GPP 2020*f*) it is mentioned that the mobile network authorization methods apply the OAuth 2.0 framework as specified in RFC 6749 (Dick 2012). It also states that client credentials should be used as grants and access tokens shall be in JavaScript Object Notation (JSON) Web Token format and can be hardened with digital signatures or MAC based on JSON Web Signature (Jones et al. 2015).

Federated Delegation

Delegation allows the assigning of access rights to a user by an administrator or another user. The administrative user does not need to have the ability to use the access right, but it requires the user to have the ability to use the access right being delegated (Crampton & Khambhammettu 2008). A federated delegation mechanism can be applied to the capability generation and propagation process for authorization and capability revocation management. The use of ABAC and CBAC with Federated Identity (FId) in a content-aware mobile network could efficiently address challenges in ACL strategy processing of a hybrid security mechanism. By delegating some of the authentication and authorization tasks to other security domains, it supports the 5G security policies and ubiquitous services access in different domains from multiple SPs. Processing validation of capability in the HN and third-party SPs enables a flexible, elastic, context-aware, and fine-grained ACL for D2D communications. Hence, inter-domain delegation with a variety of ACLs techniques enables all-round service security in 5G enabled D2D communications.

5.4 Summary

This chapter introduced the security framework, explained the security levels that are addressed by the related work and those that still need to be addressed. It defined the security model, levels, entities, and the underlying security protocols of the security framework. In addition, authentication, and authorization techniques to be used by the security protocols were presented. The next chapter 6, formally analyses and verifies the primary authentication protocols as specified by 3GPP to find out the security guarantees that are provided by 5G-AKA and 5G EAP-AKA’ protocols.

Chapter 6

Network Access Security

6.1 Introduction

5G security should provide guarantees to users and MNO from the defined security properties like authentication, confidentiality, and integrity. The UE must be authenticated to access the network, this is achieved by applying AKA protocols between the UE and HN to address the network-level security. This chapter presents a formal analysis of NAC protocols on whether they address the security requirements as defined by the 3GPP. For our framework efficiency, it is assumed that these protocols provide the security foundation for the proposed solution later in this thesis. The 5G-AKA and 5G EAP-AKA' protocols are formally analysed, and their security protocols are evaluated for security and privacy guarantees as described in the 3GPP standard TS 33.501 (3GPP 2020*f*). Some of the work in this chapter is also presented in (Edris et al. 2020*b*).

The rest of this chapter is structured as follows. Section 6.2 discusses the related work on AKA protocol. The 5G-AKA protocol is introduced in section 6.3. The modelling of 5G-AKA is presented in section 6.4. In section 6.5, the verification of 5G-AKA is presented. Section 6.6 introduces 5G EAP-AKA' protocol. The modelling of 5G EAP-AKA' is presented in section 6.7. The verification of the protocol is presented in section 6.8. This chapter is summarised in section 6.9.

6.2 Related Work

Different from the related work, this study provides an updated formal analysis of a 5G AKA protocol based on the latest version of TS 33.501 v15.5.0. It formalizes the protocol using two models illustrating all message flows and cryptographic operations presented in section 6.3. The related work omitted the four entities model which shows the function of AV in providing the HN more home control regarding the authentication process as proof of UE participation to the HN. This model shows the role played by the random nonces included in the resynchronization process that was previously sent by the ARPF to AUSF in the previous transaction between the two entities. It gives an up-to-date comprehensive formal analysis in various adversarial settings with different observations of the revised 3GPP specifications.

A malicious actor impersonating another user to SN was found in (Dehnel-Wild & Cremers

2018), this is no longer possible as the 33.501 specifications v0.70 have been revised, now Serving Network Name (SNN) binds UE and 5G to a specific authentication session. It was also based on SUPI not being included in the authentication response between ARPF and AUSF, but it is now included in the 33.501 updated version as shown in the four entities model in section 6.4. Additionally, it is stated in (Basin et al. 2018) that some security goals were not met by the 5G standard except under additional assumptions. However, these findings are also no longer accurate as now, the SUPI and UE/SN session key K_{SEAF} are sent together. Also, the binding of SNN to SEAF and UE as well as AUSF and ARPF solves the problem of SN assigning the K_{SEAF} or SUPI to the malicious UE, this is achieved by including the SNN in response (RES^*)/expected response ($XRES^*$) and the derivation input for K_{SEAF}. Moreover, the analysis in (Basin et al. 2018) based on 33.501 v15.1.0 mentioned that the attacker can observe a 5G-AKA authentication session and replay the SN's message to a subscriber. It was also added that with the subscriber's reply, the attacker can distinguish between the two subscribers in different sessions, this is in relation to MAC and synch failure messages. Which can be used to track mobile subscribers over time, however, it requires assumption like the subscriber using the same BS multiple times to be true.

5G-AKA protocol is still affected by all known attacks except the IMSI-catcher and it is also vulnerable to de-synchronization attack (Koutsos 2019). An attack using a fake BS is unlikely due to the SNN binding to the entities. In addition, since SEAF does not react to unsolicited synch failure messages or send a new authentication request message to the UE before it gets a response from AUSF or times out, then permanent de-synchronization can be avoided. The related work acknowledges the challenges of modelling 5G AKA protocol that is why AUSF and ARPF entities are mostly modelled as one entity HN, SQN is replaced with nonces, and the re-synchronization phase is omitted to simplify the process. Moreover, XOR is either not modelled or simpler algebraic properties are used. The simplification in protocol modelling may lead to some security properties not being analysed and attacks as the one found by our analysis could be missed. The 5G-AKA modelling differences are demonstrated in section 6.3.

6.2.1 5G Identifiers

3GPP has specified different identifiers for UE for different applications. MNO will allocate SUPI as a unique identifier of the UE, SUPI is converted to SUCI while being sent on a wireless link to avoid sending the UE ID in plaintext as specified in TS 23.501 (3GPP 2020*g*). SUPI is located in the USIM and the UDM function. During the authentication process, the VN's AMF may assign a temporary ID to the USIM called Global Unique Temporary Identifier (GUTI), if the user is registering to the network for the first time. Other identifiers used in 5G are:

- SUPI consists of mobile's country, network codes, and subscriber identification number.
- SUCI consists of SUCI type, HN ID, public key ID, routing indicator, protection scheme, and protection scheme.
- GUTI consists of PLMN ID, AMF ID, and 5G temporary mobile subscriber ID.
- GPSI can contain either an external identifier (local and domain identifiers) for use outside the 3GPP system or an international subscriber's directory number to identify mobile phone numbers globally.

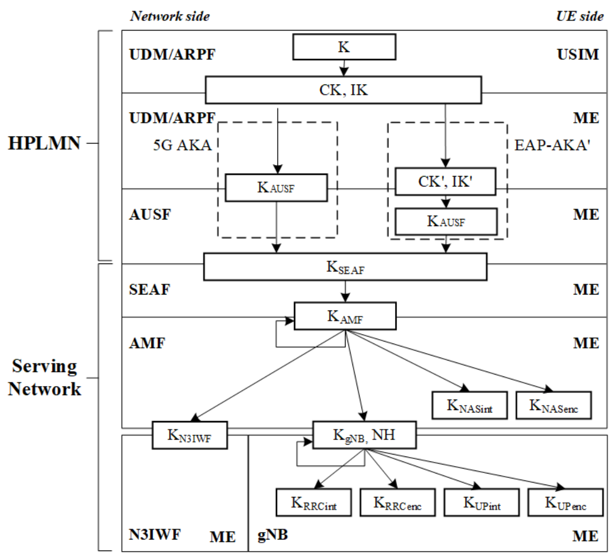


Figure 6.1: 5G Keys Hierarchy and Derivation (3GPP, 2020)

6.2.2 Keys Hierarchy and Derivation

The key K provisioned in the USIM and the 5GC is the primary source of the security context in the same way as in a 4G system. After a successful primary authentication, the SN specific anchor key K_{SEAF} is derived from K. The 5GC and ng-RAN related key requirements are described in the key hierarchy generation in TS 33.501 as shown in Fig. 6.1. It involves the USIM/ME and other entities such as ARPF, AUSF, SEAF using Key Derivation Function (KDF). The Key derivation occurs on both the UE and Network side. The keys related to authentication include key K, Cipher Key (CK) and Integrity Key (IK) that are used to derive other keys such as intermediate key K_{AUSF} and anchor key K_{SEAF} for secure communication between UE and SEAF. K_{SEAF} is used to derive key K_{AMF}, which is used to derive other keys for protecting the confidentiality and integrity of the NAS (K_{NASint}, K_{NASenc}) and RRC signalling (K_{RRCint}, $K_{KRRCenc}$), Up traffic (K_{UPint}, K_{UPenc}), mobility (K_{gNB}), NH (K_{NG-RAN^*}, K'_{AMF}), forward security, non-3GPP access (K_{N3IWF}), which are also used for further derivation of horizontal and vertical keys. The key hierarchy includes K, CK/IK, K_{AUSF}, K_{SEAF}. The K_{AUSF} is derived by ME and ARPF from CK/IK and while AUSF and ME derive K_{SEAF} from K_{AUSF}.

6.3 5G-AKA Protocol

In 5G standard (3GPP 2020*f*), 5G AKA protocol was specified as the main method of AKA between UE and HN, based on the Evolved Packet System (EPS)-AKA protocol (3GPP 2020*b*). As mentioned earlier 5G security authentication mechanisms enable the HN to get notified when the UE is authenticated in each network and to take the final call on the authentication process. After a successful run of the protocol, parties should be able to share and agree on the anchor key K_{SEAF} and derivation of other keys achieved for communication between local network entities. Therefore, the secrecy of K_{SEAF} is crucial for the security of future communications and operations.

6.3.1 5G Architecture Overview

5G standards are still under development, the overview points out the key differences in protocols and reference points between legacy systems and 5G, full details in (3GPP 2016). The 5G system architecture (3GPP 2020*f*) consisting of UE, SN, and HN as presented in chapter 2. The access network consists of the gNB referred to as the SN where the UE attaches to when connecting to the network. The HN is where the database and other security functions responsible for the authentication and storing of the security context reside. The UE and the SN communicate on a wireless channel and the SN and HN communicated on a secure wired channel. The 5G security architecture (3GPP 2020*f*) consists of UE, SEAF and AUSF, and ARPF, as discussed in chapter 5. The UE shares the long-term symmetric key K and other information with ARPF which are used during the authentication process between UE and network. SUPI is encrypted with HN public key into SUCI while in transit and only decrypted by the HN to give HN more security control. The UE and SEAF must achieve mutual authentication and must be in possession of the session key before communicating as this occurs on an insecure wireless channel while the communication between SEAF, AUSF, and ARPF occurs on a secure wired channel.

The 5G-AKA protocol uses the following communication channels between entities as shown in Fig. 6.2; `UE` $\leftrightarrow$ `SEAF (UE` $\leftrightarrow$ `SN)`, `SEAF` $\leftrightarrow$ `AUSF (SN` $\leftrightarrow$ `HN)` and `AUSF` $\leftrightarrow$ `ARPF (HN)`. As specified by the 5G Standard, the communications between SEAF, AUSF, and ARPF are within the 5GC, which makes it a secure connection. Various security properties must be met for E2E interconnection security, the security mechanisms shall satisfy the security requirements such as confidentiality and integrity between source and destination network. The destination network is able to determine the message authenticity of the source network, achieved using standard security protocols.

In addition, channel properties must be met even though the delivery of messages or the right ordering of received messages is not a guarantee. These properties are used for setting up and maintaining IPSec, Datagram TLS (DTLS), or diameter sessions for a secure channel such as SEAF to AUSF to prevent attacks such as eavesdropping and wiretapping (3GPP 2020*f*). The channel between UE and SEAF is considered insecure.

6.4 Modelling of 5G-AKA Protocol

While developing the 5G-AKA protocol, some changes were made in relation to EPS-AKA, the authentication and roaming were mainly based on trust in legacy systems. The HN could not verify if the roaming partner claims of a UE visiting a specific SN were true or not. The primary authentication protocols like 5G-AKA address this problem by increasing HN control (3GPP 2017) and give more guarantees to HN. This is achieved through authentication confirmation, whereby

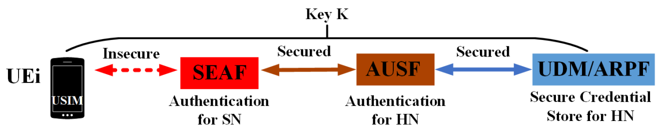


Figure 6.2: 5G Entities Communication

the UDM stores the UE's SUPI, authentication status, timestamp, and the SNN after receiving authentication confirmation from AUSF. The values used for 5G AV are defined in Table 6.1, the value RES has been divided into two halves for backward compatibility. When RES is received by SEAF, only the first half of RES can be verified, due to AV containing $XRES^*$ value, however, the HN can verify both RES and RES^* values. The AV includes a random nonce RAND as a challenge, AUTN as an authentication token to check the freshness and authenticity of the challenge as $XRES*$ is the expected response to the challenge.

6.4.1 Modelling Choices

The 5G-AKA protocol is modelled using four entities (UE, SEAF, AUSF, and ARPF) and three entities (UE, SN, and HN) referred to as model A and model B, respectively. The four entities model illustrates the role played by AUSF calculating $HXRES$, verifying RES, and sending RAND to ARPF as part of AUTS re-synchronization message. Also, the communication channels between the communicating parties are modelled with some security properties and the messages exchange are

Table 6.1: 5G-AKA Notation and Description

Notation	**Description**
HNname/HNID	(MMC, MNC)
SNN	service code:SNID
SNID	SN identifier
Ki/K	preshared symmetric key (UE, HN)
PKHN/SKHN	HN public key/private key
RAND	random nonce challenge
SUPI	(MMC, MNC, MSIN)
SUCI	(MMC, MNC, enc(MSIN))
AUTN	(SQNHN $\oplus$ AK, MAC, AMF)
MAC,XMAC	f1(K, (SQNHN, Rand, AMF))
RES, XRES	f2(K, Rand)
RES*,XRES*	KDF((CK, IK), (Rand, RES /XRES))
CK	f3(K, Rand)
IK	f4(K, Rand)
AK	f5(K, Rand)
HXRES*	SHA-256(Rand, XRES*) one-hash function
HRES*	SHA-256(Rand, RES*) one-hash function
AMF	authentication management field 0-1
KAUSF	KDF((CK, IK), (SNN, Xor(SQN,AK))
KSEAF	KDF(KAUSF, SNN)
SQN	sequence number = Xor(Xor(SQN,AK),AK)
SQNUE	UE SQN
SQNHN	HN SQN
MACS	f1* (AMF, RAND, K, SQNUE)
AK*	f5* (K, Rand)
AUTS	(Xor(SQNUE,AK) || MACS

tagged on both secure and insecure channels. The following communication channels are modelled:

1. `PubSecChannel`, consisting of public channel UE-SN and private channel SN-HN. The insecure channel (UE and SN) and secure channel (SN and HN).

2. `PubChannel` consisting of public channel UE-SN and public channel SN-HN. The insecure channel between (UE and SN) and compromised channel (SN and HN).

In terms of cryptographic messages, the SQNHN is a counter for each UE and the SQNUE indicates the highest SQN accepted by the USIM. It assumes that the attacker cannot follow SQN creation, so it is not known at first. A natural number can be modelled to as a check for freshness, with the possibility of SQN being out of sync during protocol execution, an attacker can trigger AV request hence increasing the SQNHN as decreasing any SQN is not possible in the UE or the HN side (Basin et al. 2018). The mac failure and sync failure messages for re-synchronization and authentication procedures are modelled to an extent as our focus is on authentication and authentication failure, not re-authentication. The concealment of the SQN uses XOR and the public key with ECIES profiles for protection securely provisioned in control of the HN.

For Implicit authentication of SN to UE, the primary AKA binds the K_{SEAF} to the SN, preventing one SN from claiming to be a different SN. The SNN is used as a parameter for a set of derivations from key K to the anchor key and a binding element. The UE and the SEAF shall both construct the SNN, by setting the service code to "5G" and the network identifier to "SNID" of the network the UE is trying to authenticate to and for SEAF to the SN the AUSF is sending authentication data to. Concatenate the service code and SNID. The SNN = (Source code "5G". SNID). The SNN also binds RES^{*} and $XRES*$ to the SN. This ensures that the anchor key is for specific authentication sessions for specific entities. SNN links UE to SEAF and AUSF to SEAF.

6.4.2 Security Requirements and Assumptions

Most of the assumptions and requirements are extracted from TS33.501 (3GPP 2020*f*) and TS 33.102 (3GPP 2020*a*) but there are some that are not. The communication between the SN and the HN is assumed to be transmitted on a secure channel, hence it should provide authentication, confidentiality, integrity, and replay protection. The channel between the UE and SN is vulnerable to eavesdropping, manipulation, interception, and message injection attacks. In a case where the channel between SN and HN is not secure, it is exposed to the same attacks. It is assumed that cryptographic primitives such as the functions f1, f1*, f2 provide integrity as MACs, and f3, f4, f5, f5* provide integrity and confidentiality as a cipher, integrity, and anonymity respectively as defined in (3GPP 2020*a*). Functions f1 and f1* also provide confidentiality to the input data of AV like SQN, while the individual messages use their own cryptographic protections directly.

The parties involved can be compromised if the attacker can gain access to the presumed secure channel (SN and HN) through a compromised SN. It is assumed that the diameter protocol (Engel 2014), (RIFS 2016) can be compromised due increasing capabilities of the adversary. This is supported by 5G characteristics that increase the attacker's vector. Moreover, the attacker may be in possession of genuine or compromised USIMs, whereby, the attacker is able to get access to all secret values on the USIMs, i.e., SUPI, Key K, and SQN, leading to gain access to secrets, key K, SK_{HN}, and SUPI from compromised HN. Initially, it is assumed that the UE credentials such as key K and SUPI on non-compromised entities are secrets shared between UE and HN. The SQN value and the HN private key SK_{HN} are also assumed to be secrets when the protocol starts, the attacker may try to find out how SQN is incremented but not possible to guess it, due to the big size of its bits counter.

The desired security properties for 5G-AKA protocol are authentication, confidentiality, integrity, and privacy as specified (3GPP 2020*f*), interpreted and briefly discussed below:

1. Authentication Properties: The SN that UE is trying to connect to should be authorized by HN. During the primary authentication procedure, the SN should authenticate the SUPI, obtain a non-injective agreement on SUPI with the UE, hence authenticating the UE and UE authenticating SNID through implicit key authentication. The UE is authorized by the SN with the subscription profile from the HN, based on the authenticated SUPI. The UE must get assurance that a successful authentication with SN can only occur with HN authorization, while a UE gets a non-injective agreement on SNN with its HN after key confirmation. In addition, the HN obtain the aliveness of the UE from the SN, which is a non-injective agreement on SNN from the HN's point of view with the UE.

2. Secrecy and Confidentiality Properties: 5G-AKA protocol should guarantee the secrecy of SUPI, K_{SEAF}, K, and SK_{HN}, that the knowledge of the K_{SEAF} in one session is not enough to derive another K_{SEAF} in an old or new session. It emphasises that the same K_{SEAF} should never be established twice.

3. Privacy Properties: Privacy in 5G has been strongly defined in (3GPP 2017), subscription privacy deals with preserving subscribers' information, e.g., identifiers and data. The legacy generation had security requirements for user ID's confidentiality, anonymity, and untraceability but attacks such as IMSI-catchers breached privacy and eavesdropped as passive attack (Rupprecht et al. 2018). 5G has precise requirements on privacy, since SUPI is a subscriber identifier it is sensitive and must remain secret. Moreover, if an attacker gets SUPI, can easily identify a subscriber and their location. In addition, the SQN must remain secret to prevent USIM data leakage attacks, which leaks USIM age and activities leading to further attacks like traceability, location, and monitoring. Since this was a minimal requirement in legacy systems hence should be a requirement in 5G (Basin et al. 2018).

The 3GPP standard security requirements are enough to provide the security guarantees expected by the mobile network systems. The previous work is based on TS 33.501 v15.1.0 while our analysis is based on the latest TS 33.501 version. The main security goals and framework have not changed much, the changes made in the specification updated version are to address some of the issues raised by related work as discussed concisely in the next sections. The protocol should provide mutual authentication and establish the anchor key. The UE ID, SK_{HN} and SQN should stay a secret during the AKA process and K_{SEAF} not revealed to any other party apart from those involved in the protocol run. In addition, the UE should get authorization from HN on SN in real-time, and parties involved in the AKA protocol should agree on the shared key K_{SEAF}.

Even though the diameter base protocol can be secured using TLS, DTLS, or IPsec but it is still vulnerable to several attackers like masquerading, malware, DDoS attacks that can be used for further attacks on the mobile network (Thanh et al. 2014), (Enisa 2018). The trust enhancement in 5G is due to attacks such as key theft, routing attack Signalling System 7 (SS7) (Engel 2014) and impersonation of network nodes and source address spoofing (RIFS 2016) in signalling messages which exploited the trusted domain of the internetwork connectivity in legacy systems. Encryption is enabled in diameter unlike in SS7 but in practice, mobile operators almost never use encryption inside the network, only occasionally on its boundaries. When used it is based on the P2P principle, not E2E, the network security is built on trust between operators. Furthermore, since in diameter protocol every request receives a response and uses the same route, this makes it easier to intercept and gather information. Different solutions have been proposed in this study, an investigation is conducted on how the vulnerabilities in the 5GC can affect the 5G-AKA through formal methods.

Having said that however this research agrees with (Dehnel-Wild & Cremers 2018), (Basin et al. 2018), (Koutsos 2019) that since the security of the authentication depends on successive procedures specified by 5G standard, cannot all be assessed in one study. Therefore, to assess if all the current and future specified successive procedures correctly enforce these requirements, more studies must be conducted on all 5G use cases as 5G is still prone to many attacks including traceability and location attacks.

6.4.3 Protocol Messages Exchange and Execution

This subsection explains the 5G-AKA protocol execution and message exchange based on model A. At the beginning of the protocol run the involved parties know the following information. `UE: (K, SUPI, pkHN, SQNUE); SEAF: (SNN); AUSF:(); ARPF: (K, SUPI, AMF, SQNHN).` The AKA protocol in 5G consists of three phases: (i) authentication initiation and method selection, (ii) the protocol, and (iii) resynchronization. The protocol message (msg) exchange between the participating parties is explained next with some text concisely omitted and reference to notation in Table 6.1.

Phase 1 - The Authentication Initiation and Method Selection:
In this phase, the initializing of the authentication and selection of the method of authentication occurs and the 5G-AKA protocol is selected, Fig. 6.3 shows the message exchange. The SEAF in SN initiates the authentication procedure with the UE after that UE asks to connect to it.

`Msg1. UE` $\rightarrow$ `SEAF: (N1Message)`

Then UE sends authentication request in $N1Message$ with SUCI = `({SUPI},pkHN),HNID)`.

`Msg2. SEAF` $\rightarrow$ `AUSF:(NausfAuthnReq)`

The SEAF receives message 1, adds its SNN then sends $NausfAuthn$ request message containing SUCI and SNID ($SUCI \parallel SNN$) to AUSF in the relevant HN.

`Msg3. AUSF`$\rightarrow$ `ARPF:(NudmAuthnReq)`

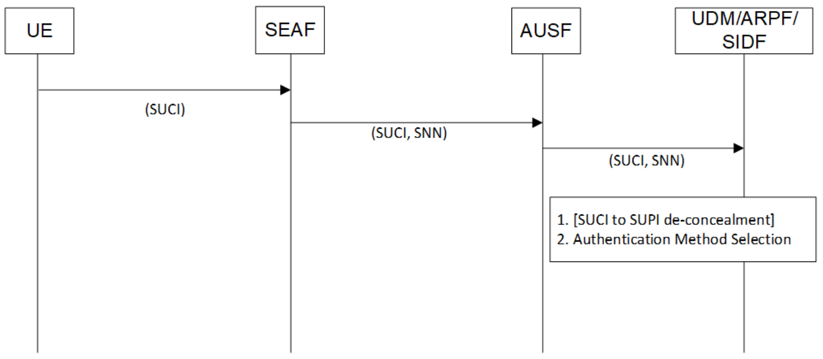

UE
SEAF
AUSF
UDM/ARPF/ SIDF
(SUCI)
(SUCI, SNN)
(SUCI, SNN)
1. [SUCI to SUPI de-concealment]
2. Authentication Method Selection

Figure 6.3: 5G-AKA Initialization Phase Message Flow

When AUSF receives message 2, before using the SNN, AUSF checks that the SEAF is authorized, then sends ($SUCI \parallel SNN$) in $NudmAuthn$ get request message to UDM/ARPF. When the UDM/ARPF receives message 3 it retrieves the SUPI and chooses an authentication method (5G-AKA or 5G EAP-AKA'). It decrypts SUCI to SUPI with the help of SIDF.

Phase 2 - The Protocol:
The 5G-AKA protocol flow in a form of challenge-response between entities as shown in Fig. 6.4.

`Msg4. ARPF` $\rightarrow$ `AUSF:(NudmAuthnResp)`

After retrieving SUPI, UDM/ARPF generates authentication vectors $AV = (RAND, AUTN, XRES^*, K_{AUSF}, SUPI)$ with AMF separating bit set of AMF* in AUTN 0 to 1. First, it generates a RAND and SQN then AV from RAND, the UE's key K and SQN, and calculates $XRES^*$ and K_{AUSF} key for AUSF. The ARPF sends AV in $NudmAuthnResp$ get response message to the AUSF indicating 5G-AKA is to be used.

`Msg5. AUSF`$\rightarrow$ `SEAF: (NausfAuthnResp)`

The AUSF generates the 5G AV from the AV received in message 4, computes $HXRES^*$ a hash of $XRES^*$. The AUSF stores $XRES^*$, SUPI and K_{AUSF}, derives K_{SEAF} from K_{AUSF}, replaces $XRES$ and K_{AUSF} with $HRES$ and K_{SEAF}, respectively. It then removes the K_{SEAF}, sends the AV ($RAND, AUTN, HXRES^*$) in $NausfAuthnResp$ authentication response to the SEAF.

`Msg6. SEAF` $\rightarrow$ `UE:(AuthReq)`

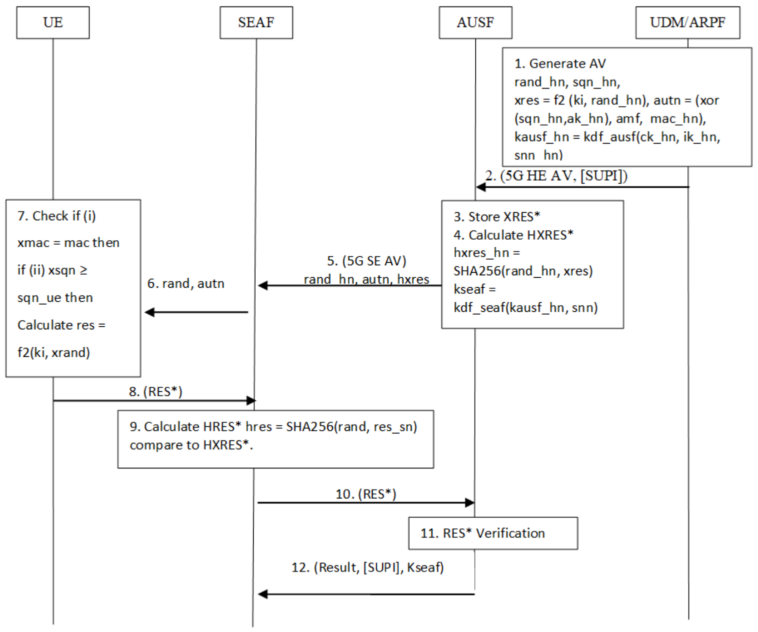


Figure 6.4: 5G-AKA Protocol Phase Message Flow

When SEAF receives message 5, it keeps $HXRES^*$and sends $RAND \parallel AUTN$ to the UE with some of the native security contexts. It includes the ABBA parameter for protecting new security features.

`Msg7. UE → SEAF:(AuthResp)`

When UE receives message 6, it forwards the $RAND \parallel AUTN$ to the USIM, the USIM verifies the freshness of the AV and checks if $AUTN$ is acceptable. First computes AK and retrieves the SQN. Then computes $xMAC$, (i) it checks if $xMAC = MAC$ and also checks (ii) if SQN is in the right range $SQNUE < xSQNHN$. If (i) and (ii) are the expected response, USIM computes RES. And finally, it computes CK and IK. The USIM returns RES, CK, IK to the UE. The UE computes RES^* from RES, calculates K_{AUSF} from CK and IK then K_{SEAF} from K_{AUSF} and checks that the "separation bit" in the AMF field of $AUTN$ is set to 1. UE returns RES^* in $AuthResp$. The UE proves its ID and ownership of Key K by sending RES^* to the SEAF.

`Msg8. SEAF → AUSF:(NausfAuthnReq)`

When SEAF receives message 7, it calculates $HRES^*$ the hash of RES^*, then compares it with the $HXRES^*$ received from the AUSF in message 5. If it matches $HRES = HXRES$, the authentication is considered successful by SEAF from the SN point of view. A RES^* is sent in $NausfAuthnReq$ message to the AUSF. The process is aborted if $HRES$ is not equal to $HXRES$.

`Msg9. AUSF→ SEAF: (NausfAuthnResp)`

When AUSF receives the RES^*, it checks if the AV has expired and if it has then the authentication is considered unsuccessful. AUSF compares the received RES^* with $XRES^*$. That is $RES^* = XRES^*$, the authentication is considered successful by AUSF from the HN point of view. If the authentication was successful, AUSF sends $K_{SEAF} \parallel SUPI$ to SEAF in message 9.

Phase 3 - Re-synchronization:

The out-of-sync SQN is updated in HN during the re-synchronization procedure as shown in Fig. 6.5. At the UE side, if the received $AUTN$ fails the verification, the USIM informs the ME of the reason for failure whether it is MAC or synchronization failure and it passes the AUTS parameter to the UE.

`Msg10. UE → SEAF: (AuthnFail)`

The UE sends $AuthnFail$ NAS message authentication failure to SEAF in message 10 that includes $mac\_failure$ or $synch\_failure$ message with AUTS.

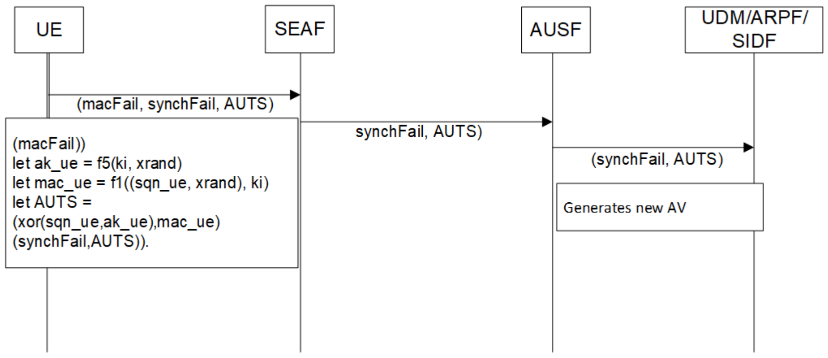

UE
SEAF
AUSF
UDM/ARPF/
SIDF
(macFail, synchFail, AUTS)
synchFail, AUTS)
(synchFail, AUTS)
(macFail))
let ak_ue = f5(ki, xrand)
let mac_ue = f1((sqn_ue, xrand), ki)
let AUTS =
(xor(sqn_ue,ak_ue),mac_ue)
(synchFail,AUTS)).
Generates new AV

Figure 6.5: 5G-AKA Synchronization Phase Message Flow

`Msg11. SEAF` $\rightarrow$ `AUSF: (NausfAuthnFail)`
When SEAF receives message 10 from the UE, SEAF may request re-identification from UE in case of mac_failure or start a new authentication process in case of sync_failure then SEAF sends *NausfAuthnFail* message to AUSF indicating sync_failure with AUTS.

`Msg12. AUSF` $\rightarrow$ `ARPF: (NudmAuthnFail)`
The AUSF sends *NudmAuthnFail* message to the UDM/ARPF, it includes $Synch_failure$ message, AUTS, and RAND that was sent to the UE in message 6 the preceding *AuthRequest* message, and AUTS received in message 12. In HN the ARPF retrieves SQNUE from AUTS, checks if SQNHN is in the correct range and if the next SQNHN would be accepted by the USIM. Therefore, if the SQNHN is in the correct range, the UDM/ARPF generates a new AV. Otherwise, it verifies AUTS, if the verification is successful, the ARPF resets the value of the counter SQNHN to SQNUE. Then UDM/ARPF sends new AV for the UE to the AUSF. The AUSF runs a new authentication procedure with the UE but this falls outside of the scope of this research.

6.5 Formal Verification of 5G-AKA Protocol

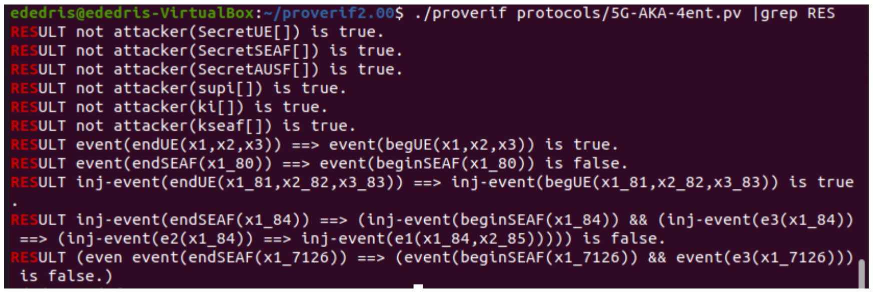
This section follows the same ProVerif process defined in chapter 3. The security protocol modelling in ProVerif is made up of declaration, process macros, and main processes. With queries used for the correctness rectification and secrecy of a protocol. The ProVerif syntax and applied *pi* calculus are used to encode and specify the protocol concisely using a declaration of types, functions, queries, and events such as the following:

- Free names are free variables that are known to the public, globally known whereas bound names are locally known by the process like the `free pubChannel:channel` for communication and [**private**] excludes names from the attacker.
- Types: key, id and nonce.
- Functions: `fun f2(key,nonce):bitstring, fun xor(bitstring,bitstring):bitstring.`
- Queries: `query attacker (SecretUE), query attacker (supi)`, `(ki)`, and `(kseaf)` are used to test the secrecy of supi, ki, and kseaf.
- Events: Querying events uses correspondence assertion to test the relationship between events (authentication). `query x1: id, x2: id, x3: key; event (endUE(x1, x2, x3)) ==> event(begUE(x1, x2, x3)).`
- Process: The protocol is encoded using the main process and process macros `processUE`, `processSEAF`, `processAUS`, `processARPF`. A number copies of the system entities (UE, SEAF, AUSF, ARPF) macros are started with the required parameters as multiple sessions of the roles.

6.5.1 Formal Analysis of the 5G-AKA Protocol

The protocol was simulated using two models on secure, insecure channels and the following processes:

- Model A (four parties' protocol) in Appendix A.1.
  `processUE` as UE, `processSEAF` as SEAF, `processAUSF` as AUSF and `processARPF` as ARPF



```
ededris@ededris-VirtualBox:~/proverif2.00$ ./proverif protocols/5G-AKA-4ent.pv |grep RES
RESULT not attacker(SecretUE[]) is true.
RESULT not attacker(SecretSEAF[]) is true.
RESULT not attacker(SecretAUSF[]) is true.
RESULT not attacker(supi[]) is true.
RESULT not attacker(ki[]) is true.
RESULT not attacker(kseaf[]) is true.
RESULT event(endUE(x1,x2,x3)) ==> event(begUE(x1,x2,x3)) is true.
RESULT event(endSEAF(x1_80)) ==> event(beginSEAF(x1_80)) is false.
RESULT inj-event(endUE(x1_81,x2_82,x3_83)) ==> inj-event(begUE(x1_81,x2_82,x3_83)) is true
.
RESULT inj-event(endSEAF(x1_84)) ==> (inj-event(beginSEAF(x1_84)) && (inj-event(e3(x1_84))
 ==> (inj-event(e2(x1_84)) ==> inj-event(e1(x1_84,x2_85))))) is false.
RESULT (even event(endSEAF(x1_7126)) ==> (event(beginSEAF(x1_7126)) && event(e3(x1_7126)))
 is false.)
```

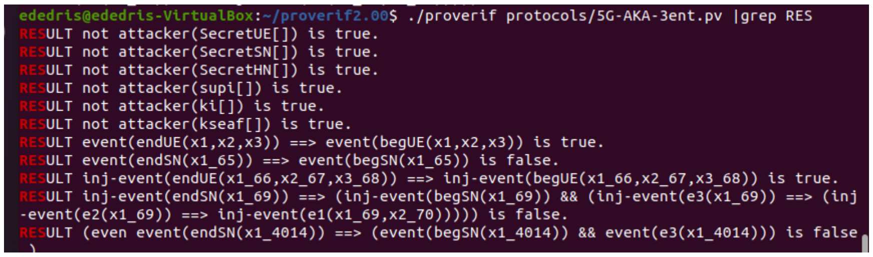
Figure 6.6: 5G-AKA Model A Attack ProVerif Results

- Model B (three parties' protocol) in Appendix A.2.
  `processUE as` UE, `processSN` as SN, and `processHN`as HN

When the protocol was modelled and run in ProVerif, there was no effect on the protocol apart from the emphasis on the message exchange in model A, on the `pubsec channel` and no attack found. However, when the protocol was run on the compromised channel, the authentication did not hold as assumed by the standard. With the possibility of an attacker getting access to all communication channels including those in the HN. The security properties this research is interested in are K_{SEAF}, mutual authentication, and the secrecy of the SUPI and key K. In relation to attacks discussed informally, an attack was found on this protocol for model A and model B, respectively.

ProVerif results of model A as shown in Fig. 6.6 indicates that the secrecy of `secretUE`, `secretSEAF, secretAUSF, supi, ki, kseaf` and authentication of UE to SN holds but the authentication of SN to UE does not hold on both non-injective and injective agreements. In addition, ProVerif results of model B as shown in Fig. 6.7 indicate that the same secrecy properties as in model A hold, authentication of UE to SN holds and but authentication of SN to UE does not hold on both non-injective and injective agreements. The study deep analysis focuses on model A since most related work based their arguments on a model like our model B.

With the `event endSEAF` indicating that the SEAF has completed the protocol, the UE received message 4 and sent message 5, `e1` indicates that the SEAF sent message 4. These events take all parameters of the protocol as arguments: `autn`, and the `rand`, except `e2` that checks if `xsqn = xor (xored_sqn, ak), xmac = f1((xsqn, xrand), ki)` and if `xmac = mac` then if `xsqn = sqn_ue`. If the arguments are true, then RES is sent otherwise it sends either `MAC_failure`



```
ededris@ededris-VirtualBox:~/proverif2.00$ ./proverif protocols/5G-AKA-3ent.pv |grep RES
RESULT not attacker(SecretUE[]) is true.
RESULT not attacker(SecretSN[]) is true.
RESULT not attacker(SecretHN[]) is true.
RESULT not attacker(supi[]) is true.
RESULT not attacker(ki[]) is true.
RESULT not attacker(kseaf[]) is true.
RESULT event(endUE(x1,x2,x3)) ==> event(begUE(x1,x2,x3)) is true.
RESULT event(endSN(x1_65)) ==> event(begSN(x1_65)) is false.
RESULT inj-event(endUE(x1_66,x2_67,x3_68)) ==> inj-event(begUE(x1_66,x2_67,x3_68)) is true.
RESULT inj-event(endSN(x1_69)) ==> (inj-event(begSN(x1_69)) && (inj-event(e3(x1_69)) ==> (inj
-event(e2(x1_69)) ==> inj-event(e1(x1_69,x2_70))))) is false.
RESULT (even event(endSN(x1_4014)) ==> (event(begSN(x1_4014)) && event(e3(x1_4014))) is false
.)
```

Figure 6.7: 5G-AKA Model B Attack ProVerif Results

or `synch_failure` message for authentication failure or re-authentication initiation. This research would like to prove the following correspondence.

```
(*Check authentication of UE to SN/HN and SN/HN to UE *)
query x1: id, x2: id, x3: key; event (endUE(x1, x2, x3)) ==>
event(begUE(x1, x2, x3)).
query x1: bitstring; event (endSEAF(x1)) ==> event (beginSEAF(x1)).
query x1: id, x2: id, x3: key; inj-event (endUE(x1, x2, x3)) ==>
inj-event (begUE(x1, x2, x3)).
query x1: bitstring, x2:nonce;  inj-event(endSEAF (x1)) ==>
(inj-event(beginSEAF(x1)) && (inj-event (e3(x1)) ==> (inj-event (e2(x1)) ==>
(inj-event (e1(x1,x2)))))).
```

However, this correspondence fails to be proved directly in ProVerif because message 4 can be replayed, resulting in several `e2` for a single `e1`. The research also tries to prove the desired correspondence but ends up noticing that `event e2` which has `res` as an argument cannot be executed before `autn` and `rand` has been sent, that is, before `e1` has been executed. Which fails in ProVerif with false.

6.5.2 The Attack Against 5G-AKA Protocol

The ProVerif results indicate there was an attack on the protocol as shown in Fig. 6.6 and 6.7. As discussed in chapter 3, in ProVerif attack derivation represents the attacker's action while the attack trace represents the real attack as an executable trace of the considered process. The derivation and trace are a sequence of steps, inputs, and outputs on the public channel and of events in relation to the process. The attack derivation and trace for Model A is as follows.

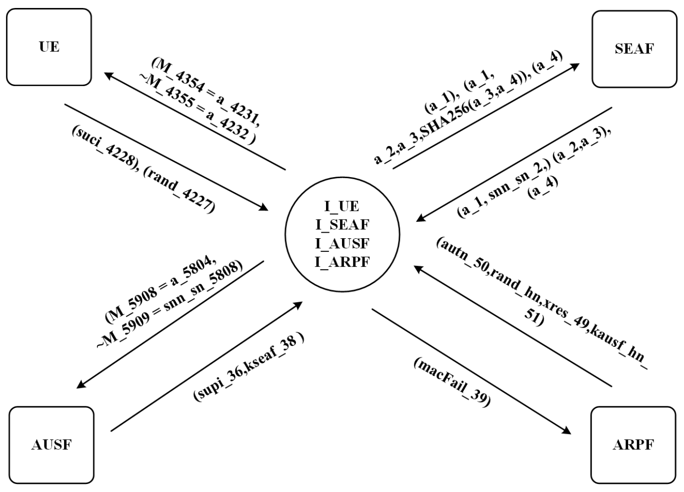

UE
SEAF
AUSF
ARPF
I_UE
I_SEAF
I_AUSF
I_ARPF
(M_4354 = a_4231, ~M_4355 = a_4232)
(suci_4228), (rand_4227)
(a_1), (a_1, a_2,a_3,SHA256(a_3,a_4)), (a_4)
(a_1, snn_sn_2,) (a_2,a_3), (a_4)
(M_5908 = a_5804, ~M_5909 = snn_sn_5808)
(supi_36,kseaf_38)
(autn_50,rand_hn,xres_49,kausf_hn_51)
(macFail_39)

Figure 6.8: 5G-AKA Attack Trace

6.5.3 Attack Derivation and Trace

The attacker I starts by eavesdropping on the communication between entities, impersonates the UE, continuing the protocol with $SEAF$, which completes the protocol with the attacker instead of UE. The attacker's actions are illustrated in Fig. 6.8 and explained concisely in derivation and trace steps below, some text is omitted for simplicity, full trace output in Appendix J:

- When `event(endSEAF(x1_80))==> event(beginSEAF(x1_80))` queried, the attacker's goal is achieved, when he gets `suci_4228` using `attacker(suci_4228)` and `attacker(rand_4227)` for `rand_4227` together with function `SHA256` to obtain `SHA256(rand_4227,x1_4230)`.

- The attacker's goal is also achieved in trace 1 when `event endSEAF(a)` is executed in session `copy a_4234` SEAF has with the attacker at event $\{57\}$. In trace 2, the injective agreement fails when `event endSEAF(a_5801)` is executed in session `a_5800`.

6.5.4 Security Analysis

Protocol Security Analysis

With the discovered attack, 5G-AKA protocol's key K might be leaked because of other attacks such as eavesdropping, hacking of a USIM card, an inside/local attack through the USIM vendor, mobile operator, or through side channels. Additionally, this vulnerability would be used by an attacker without privileged access to impersonate another user to SN in a roaming scenario. Which allows the billing of expensive phone calls or access charges to other legitimate users through eavesdropping on their initial connection (Dehnel-Wild & Cremers 2018). The analysis of the protocol is based security requirements of set 1 (Lowe 1997) and set 2 (Menezes et al. 2018) as presented in chapter 3.

Analysis based on security properties of set 1 is as follows:

- Secrecy: This is achieved since $SUPI$'s secrecy holds. By using xor and anonymity keys to protect the parameters used in derivations of keys in transit and in storage. The use of F1 and F* provides privacy protection of SQN to the data. This achievement also covers the confidentiality and privacy properties of the protocol.

- Aliveness: The aliveness of UE is obtained by HN at the SN, with a non-injective agreement on SNN with the subscribers. But also, the injective agreement on K_{SEAF} with the subscribers, the HN obtains fresh aliveness as a result.

- Weak Agreement: When SN achieves non-injective agreement on $SUPI$ with UE and the key confirmation with $SNID$ as parameter fulfils this requirement. However, the weak agreement does not hold as ProVerif results indicate.

- Non-injective Agreement: The UE obtains non-injective agreement on SNN with its HN after key confirmation of K_{SEAF} and $HNID$ as it is part of $SUPI$. The SN obtain non-injective agreement on $SUPI$ with the HN after $SUPI$ authentication by HN. It gets injective agreements on K_{SEAF} between the SN and UE. Any agreement on K_{SEAF} between the HN and the UE also guarantees that UE is attached to an authorized SN, this is achieved since K_{SEAF}'s derivation includes $RAND$ from HN and SNN from SN. Which assures the UE that SN is trusted but the authentication SN-UE fails due to the changed assumption of the communication channel security.

- Injective Agreement: The UE and the SN's injective agreement on K_{SEAF} is crucial to the protocol's objective and achieving injective agreement on K_{SEAF} for different pairs of parties means K_{SEAF} cannot be derived twice for the same session. The inclusion of $RAND$ in the derivation of K_{SEAF} guarantees an injective agreement on K_{SEAF} between the HN and the UE. The injective agreement on K_{SEAF} based on SNN with the HN assures the UE that SN is known and trusted. The UE obtains the injective agreement on K_{SEAF} with the HN to assure that the session with SN was authorized by the HN. However, the SN fails to achieve the same trust from UE as the SN-UE does not hold.

Analysis based on security properties of set 2 is as follows:

- Mutual Entity Authentication: The UE is authenticated to the SN if $RES^* = HRES$, to HN if $RES^* = XRES$, while the HN is authenticated to UE after verifying $AUTN$. The SNN enforces weak agreement and implicit authentication from HN and UE after a successful authentication and key K_{SEAF} confirmation. Additionally, when $SUPI$, SNN, and $HNID$ are sent to the HN in the first phase of the authentication and proved to hold they enforce this requirement even when some routing data is not encrypted. Moreover, the creation of SNN involves the UE, SN, and AUSF in the HN which also endorses the entity. The SNN links the UE to SEAF and SEAF to AUSF. However, the SN to UE authentication fails to hold.

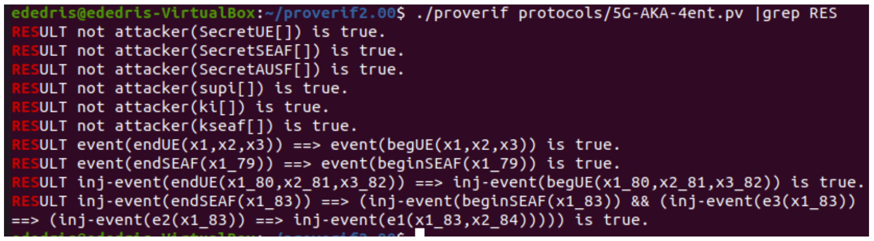
- Mutual Key Authentication: Since the authentication of between UE and HN is based on the secrecy of K_{SEAF}, it also gets implicitly authenticated by including K_{AUSF} and SNN in its derivation parameters.

- Mutual Key Confirmation: The successful AKA roundtrip between the UE, SN, and HN ending with K_{SEAF} confirmation enforce this requirement.

- Key Freshness: ProVerif has no function to check key freshness however during the authentication process as the UE checks if data involved in the authentication process is valid such as checking the freshness the `xSQN > SQN` which also facilitated the derivation of K_{SEAF}. In 5G K_{SEAF} from the previous session cannot be reused in a new session as every K_{SEAF} is linked to a session and SN by SQN and SNN, respectively. And since the secrecy of K_{SEAF} is not violated, it implies key freshness.

- Unknown-Key Share: The reachability property in ProVerif is used to check aliveness. The entities' ID and Key binding prevent this attack. The inclusion $SUPI$, $HNID$ in the authentication procedure and SNN in the KDF of K_{SEAF} links it to SN and since it is derived from key K, which is pre-shared between UE and HN also proving this requirement. The K_{SEAF} is only sent to SEAF after RES verification by AUSF.

- Key Compromise Impersonation Resilience: Since K_{SEAF} is derived from long-term key and both the secrecy of K_{SEAF} and key K hold hence they enforce this requirement. Additionally, knowing key K_{SEAF} generated in one session should not be enough to deduce key K_{SEAF} that has been generated in an old session or that will be generated in a new session. Backward and forward security on keys are not possible (Cohn-Gordon et al. 2016) this is not forward secrecy, no entity or adversary is capable of computing keys of a past session or predicting future keys. Even when the attacker manages to know K_{SEAF} key in another session, K_{SEAF} in each session should be different and confidential. However, forward secrecy and post-compromise



```
ededris@ededris-VirtualBox:~/proverif2.00$ ./proverif protocols/5G-AKA-4ent.pv |grep RES
RESULT not attacker(SecretUE[]) is true.
RESULT not attacker(SecretSEAF[]) is true.
RESULT not attacker(SecretAUSF[]) is true.
RESULT not attacker(supi[]) is true.
RESULT not attacker(ki[]) is true.
RESULT not attacker(kseaf[]) is true.
RESULT event(endUE(x1,x2,x3)) ==> event(begUE(x1,x2,x3)) is true.
RESULT event(endSEAF(x1_79)) ==> event(beginSEAF(x1_79)) is true.
RESULT inj-event(endUE(x1_80,x2_81,x3_82)) ==> inj-event(begUE(x1_80,x2_81,x3_82)) is true.
RESULT inj-event(endSEAF(x1_83)) ==> (inj-event(beginSEAF(x1_83)) && (inj-event(e3(x1_83))
==> (inj-event(e2(x1_83)) ==> inj-event(e1(x1_83,x2_84))))) is true.
```

Figure 6.9: 5G-AKA Model A Safe ProVerif Results

secrecy does not hold because if key K is compromised, the adversary can compute future and past keys. Which requires session key secrecy even when long-term key material is compromised.

Security Consideration

The adversaries nowadays are more sophisticated, and this was ignored in related work (Basin et al. 2018), (Dehnel-Wild & Cremers 2018), (Koutsos 2019). They assumed that security mechanisms in place would protect diameter protocol, the channel, and HN entities but doesn't (Engel 2014), (Enisa 2018), (RIFS 2016), so the author believes that even a secure network can be compromised in several ways which creates a wider attack vector. Since the non-injective and injective agreement on SN to UE fails to hold that indicates that replay attack is possible between SN and UE. Forward secrecy fails as the attacker could get the past and future keys by knowing key K.

The UE's privacy is put at risk if the standard is underspecified, the protocol vulnerability would allow an attacker to impersonate another user to the SN. 5G-AKA cannot provide perfect forward secrecy of K_{SEAF}, this is because a compromised USIM prior to the running of the protocol can lead to the attacker knowing key K, while K_{SEAF} can be revealed, using only K and the message $RAND, AUTN$ sent from SEAF to UE. Perfect forward secrecy could be achieved with Diffie-Hellman key exchange, but its computation is too expensive regarding mobile devices resources usage. It should be noted that authentication relying on the K_{AUSF} in AUSF is not as strong as direct authentication between the ARPF and the USIM.

The results in Fig. 6.9 show model A when the HN environment communication channels are made secure by using a more robust mechanism this might include cryptographic techniques and more secure communication protocols. Full ProVerif description of the secure protocol is illustrated in Appendix A.1. The standard should be strengthened to prevent active attacks on the privacy properties by using encryption and randomness on UE data. The sequence and unlink-ability problems (Koutsos 2019) can be solved after the replayed message has made a roundtrip to the HN, also it should be noted that resynchronization in HN is achieved by either checking if the SQNUE sent in AUTS is greater than SQNHN if not then SQNHN is set to SQNUE. The security mechanism that protects the diameter sessions should also be enhanced.

6.6 5G EAP-AKA' Protocol

This section interprets security properties and models 5G EAP-AKA' protocol as specified in the 3GPP standard, also referred to as EAP-AKA' in this thesis. A formal, security analysis and verification of the protocol is conducted to automatically identify the security properties and guarantees provided by the protocol. The 5G standard (3GPP 2020*f*) addresses the most critical security requirements in 5G and specifies EAP-AKA' as one of the methods used in the primary authentication. There has been a significant amount of research conducted on primary authentication but mostly covers 5G-AKA protocol, hence, why this research is also discussing and formally verifying the EAP-AKA' protocol using similar methods as used on 5G-AKA protocol in section 6.3.

6.6.1 Related Work

The EAP framework supports authentication method defined under RFC 3748 (Vollbrecht et al. 2004), that runs over data link layers without an IP for dedicated, wired, and wireless links for flexibility. The EAP was developed by 3GPP and verified by the EAP WG in RFC 4187 (Arkko & Haverinen 2006). It was later specified as an EAP-AKA method for authentication and session key distribution for 3rd generation mobile network UMTS based on symmetric keys and run in a USIM. EAP-AKA included options for ID privacy, result indications, and fast re-authentication. The RFC 4187 made the use of AKA method for primary authentication possible within the EAP framework, later improved in 5448 (Arkko et al. 2018) with a new EAP method, EAP-AKA'. The changes included a new KDF binding the derived keys with the name of the access network hence protection from binding down attacks. Furthermore, the EAP-AKA' can be applied as an authentication method to gain 5G and non-3GPP network access, specified in TS 33.501 (3GPP 2020*f*) as a primary authentication method. The EAP-AKA' uses CK' and IK' as specified in TS 33.402 (3GPP 2020*c*) and updates the hash function from Secure Hash Algorithm (SHA)-1 to SHA-256 and Hash Message Authentication Code (HMAC) to HMAC-SHA-256.

The EAP-AKA' protocol was specified in (3GPP 2020*f*) and proposed as one of the main methods of AKA between a mobile device and its HN. As it can be used as an alternative authentication method to 5G-AKA in primary authentication. It uses similar cryptographic primitives and fulfils similar security guarantees as explained in section 6.3. The difference is EAP-AKA' uses the EAP framework for messages exchange and the session key is derived differently as explained in the next sections. In addition, 3GPP recommends the EAP framework for the secondary authentication to external DN.

EAP Architecture Overview

The 5G EAP-AKA' uses the same architecture as the 5G-AKA protocol described in section 6.3, it consists of the following three essential parties UE, SN, and HN. The communication process, security procedure, and the entities are also the same as used in 5G-AKA which are UE, SEAF, AUSF, and ARPF as per the security architecture (3GPP 2020*f*). It should protect the SUPI with SUCI on all channels, achieve mutual authentication and agree on session key K_{SEAF}. However,

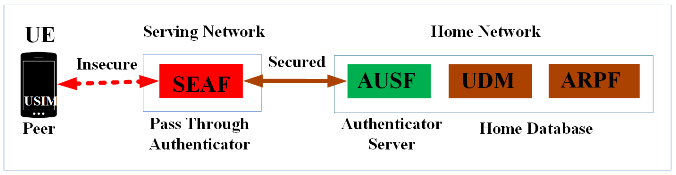


Figure 6.10: 5G EAP-AKA' Entities

the EAP framework in RFC 3748, defines peer, pass-through authenticator, and back-end authentication server roles. With 5G EAP-AKA' the EAP framework is supported with the following entities as shown in Fig. 6.10 (3GPP 2020*f*):

- The UE as the peer.
- The SEAF as the pass-through authenticator.
- The AUSF as backend authentication server with the support of the ARPF and UDM.

The main EAP-AKA' attributes in the EAP-request/response are AT_RAND, AT_AUTN, $AT\_RES, AT\_MAC, AT\_KDF, AT\_KDF\_INPUT, AT\_MAC, AT\_AUTS$. There are some significant changes that have been made in EAP-AKA', as per 5G security specifications (Arkko et al. 2018):

- Network name field: The SNN shall be constructed by the UE and the SEAF, the service code set to "5G" and the network ID to "SNID" for the network the UE is trying to authenticate to and for SEAF/SN to which the AUSF is sending authentication data to.
- The identifiers: SUPI, SUCI, SNID.
- Key derivation inputs: 5G identifiers should not be trackable for privacy preservation and, permanent identifiers like SUPI should not be transmitted outside HN.
- Session identifiers: EAP Type code $\|$ $RAND$ $\|$ $AUTN$, carries the AT_KDF_INPUT attribute, including Network Access Identifier (NAI) for the UE and AUSF. It supports future extension with KDF negotiation via the AT_KDF attribute.

Keys Derivation

The UE ID and the access network ID are used as an input in the key derivation using the at_kdf_input parameters. After a successful authentication between the UE and the HN, K_{SEAF} is derived from K_{AUSF}. The Key derivation occurs on both the UE and network sides. The keys related to authentication include keys: K, CK/IK, CK'/IK' that are used to derive the EMSK(K_{AUSF}) then K_{SEAF} and later used derive other keys to secure communication between the UE and other entities in the network. In addition, the KDF input parameters for CK' and IK' are the same only separated by a 256-bit return output, where the 128 most significant bits are for CK' and the 128 least significant bits are for IK' (3GPP 2020*c*).

6.7 Modelling of 5G EAP-AKA' Protocol

The modelling of 5G EAP-AKA' same as 5G-AKA in section 6.3 with some changes in the notation as shown in Table 6.2. The AV values are the same but 5G EAP-AKA' uses the EAP-AKA challenge/response framework. The anchor key K_{SEAF} binding with SN shall be achieved by including "NAI" into the chain of key derivations parameters which is the SNN. It should be noted that direct involvement of USIM and ARPF in an authentication procedure provides strong guarantees than the one based on K_{AUSF} in the AUSF, equivalent to EAP-AKA' fast re-authentication.

Table 6.2: 5G EAP-AKA' Notation and Description

Notation	**Description**
SNname/SNN	service code:SNID
K	symmetric key (UE, HN)
PKHN/SKHN	HN public/private keys
RAND (AT_RAND)	random nonce challenge
SUPI	(MMC,MNC,MSIN)
SUCI	(MMC,MNC,enc(MSIN))
SUCI/SUPI	User's Network Access Identifier (NAI)
AUTN (AT_AUTN)	(Xor(SQNHN AK),MAC,AMF)
MAC, MAC2 (AT_MAC)	f1(K, (SQNHN, Rand, AMF))
RES (AT_RES), XRES	f2(K, Rand)
CK	f3(K, Rand)
IK	f4(K, Rand)
AK	f5(K, Rand)
CK'	IK, CK, SNN, Xor(SQN,AK)
IK'	IK, CK, SNN, Xor(SQN,AK)
KAUSF/EMSK	KDF((CK', IK'), (SNN, Xor(SQN,AK))
KSEAF	KDF(KAUSF, SNN)
SQN	sequence number
MACS (AT_MACS)	f1* (AMF, RAND, K, SQNUE)
AK*	f5* (K, Rand)
AUTS (AT_AUTS)	Xor(SQNUE, AK) ‖ MACS
h(x)	hash value of message x
{x}{k}	message encrypted with key K

Security Requirements and Assumptions

Most of the assumptions are based on TS 33.501 (3GPP 2020*f*) specifications like those presented in section 6.3 and (Arkko et al. 2018), with UE-SN channel assumed insecure and the SN-HN assumed secure, providing authentication, confidentiality, integrity, and replay protection. It is also assumed that cryptographic primitives such as functions and hash functions provide integrity and confidentiality using derivation keys and MAC, while the individual messages use their own cryptographic protections directly. There is no cipher suite negotiation mechanism in EAP-AKA' but there is one for KDF, the security properties provided by SHA-256 such as mutual authentication, confidentiality, cryptographic binding, and session independence are as good as those of the old EAP-AKA. It is also assumed that SHA-256 is like a pseudo-random function and the pre-shared secret cannot be calculated from any keys by any practically feasible means. As per the 5G standard, EAP-AKA' uses different identifiers in different scenarios to identify the UE. The protocol key strength prevents brute force attacks but does not provide channel binding (Arkko et al. 2018).

The desired security properties for EAP-AKA' protocol are the same as the 5G-AKA protocol explained in section 6.4.2 with UE getting assurance that it is authenticating to SN authorized by its HN. The UE shall authenticate SN with the SNN through implicit key authentication and key confirmation with the secrecy of the anchor key K_{SEAF} ensured. The also EAP-AKA' uses

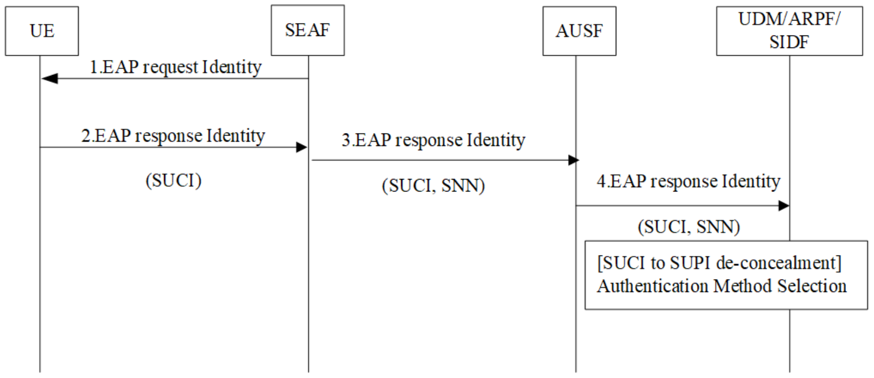


Figure 6.11: 5G EAP-AKA' Initialization Phase Message Flow

diameter-based over IPSec or TLS to provide some security.

6.7.1 Protocol Message Exchange and Execution

This subsection explains the EAP-AKA' protocol message exchange and execution, consisting of three phases and described as follows:
Phase 1: The Authentication Initiation and Method Selection
Starts with initiating the authentication and selecting a method to be used, in this case, it is the EAP-AKA'. The SEAF initiates the authentication with the UE as shown in Fig. 6.11.

`Msg1. SEAF` $\rightarrow$ `UE: (EAP-Request/Identity)`

The SEAF sends ID request to UE.

`Msg2. UE` $\rightarrow$ `SEAF: (N1Message)`

Then UE sends an authentication request message which includes $SUCI$ and $HNID$.

`Msg3. SEAF` $\rightarrow$ `AUSF: (NausfAuthnReq)`

When SEAF receives message 2, it sends $SUCI$ and SNN to AUSF in $NausfAuthnReq$ get request message to AUSF.

`Msg4. AUSF` $\rightarrow$ `ARPF: (NudmAuthnReq)`

The AUSF sends $NudmAuthnReq$ get request message to UDM/ARPF in HN. Before using the SNN, AUSF checks that the SEAF is authorized. When the ARPF receives message 4, it decrypts SUCI into SUPI with the help of SIDF and chooses an authentication method.

Phase 2: The Protocol
The EAP-AKA' flows in a form of EAP challenge/response between entities as shown in Fig. 6.12.

`Msg5.ARPF` $\rightarrow$ `AUSF: (NudmAuthnResp)`

After retrieving SUPI, UDM/ARPF generates AV with AMF^*. First it generates a $RAND$ and SQN then $XRES$ and $AUTN$. Calculates CK, IK, computes CK', IK'. The ARPF sends EAP-Response/AKA' AV ($RAND$, $AUTN$, $XRES$, SNN, $CK' \parallel IK'$, $SUPI$) in $NudmAuthnResp$ get response message to the AUSF indicating EAP-AKA' is to be used.

`Msg6. AUSF` $\rightarrow$ `SEAF: (NausfAuthnResp)`

The AUSF stores $XRES$ and $SUPI$ then sends EAP-Request/AKA'-Challenge in $NausfAuthnResp$ message that includes $RAND$, $AUTN$ and SNN to SEAF.

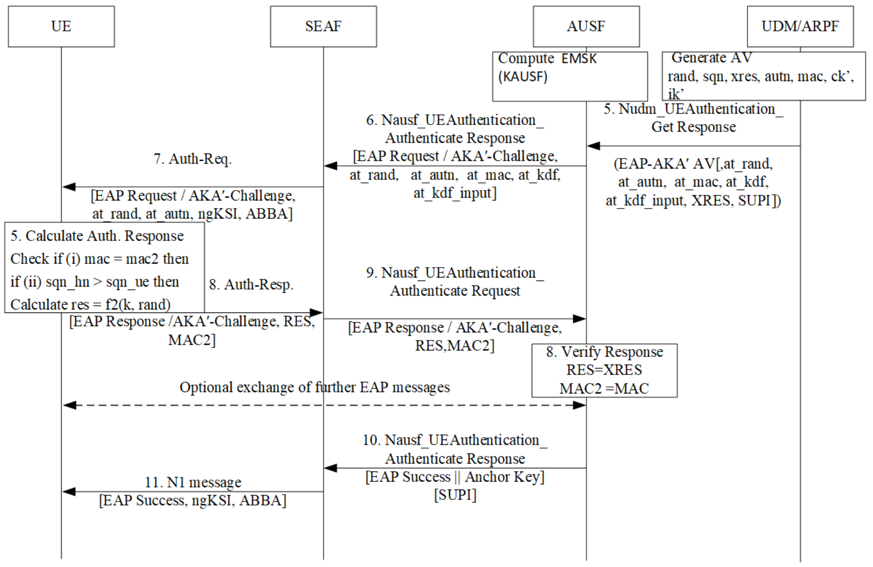

UE
SEAF
AUSF
UDM/ARPF
Compute EMSK (KAUSF)
Generate AV rand, sqn, xres, autn, mac, ck', ik'
5. Nudm_UEAuthentication_ Get Response
(EAP-AKA' AV[,at_rand, at_autn, at_mac, at_kdf, at_kdf_input, XRES, SUPI])
6. Nausf_UEAuthentication_ Authenticate Response [EAP Request / AKA'-Challenge, at_rand, at_autn, at_mac, at_kdf, at_kdf_input]
7. Auth-Req.
[EAP Request / AKA'-Challenge, at_rand, at_autn, ngKSI, ABBA]
5. Calculate Auth. Response Check if (i) mac = mac2 then if (ii) sqn_hn > sqn_ue then Calculate res = f2(k, rand)
8. Auth-Resp.
[EAP Response /AKA'-Challenge, RES, MAC2]
9. Nausf_UEAuthentication_ Authenticate Request
[EAP Response / AKA'-Challenge, RES,MAC2]
8. Verify Response RES=XRES MAC2 =MAC
Optional exchange of further EAP messages
10. Nausf_UEAuthentication_ Authenticate Response [EAP Success || Anchor Key] [SUPI]
11. N1 message [EAP Success, ngKSI, ABBA]

Figure 6.12: 5G EAP-AKA' Protocol Phase Message Exchange Flow

`Msg7. SEAF` $\rightarrow$ `UE: (AuthReq)`
When SEAF receives message 6, it sends $RAND$ with $AUTN$ to the UE in $AuthReq$ message. It also includes ngKSI and the ABBA parameter to enable binding down protection.

`Msg8. UE` $\rightarrow$ `SEAF: (AuthResp)`
When UE received message 7, it forwards the $RAND$ and $AUTN$ to the USIM, which verifies AV freshness and if AUTN is acceptable. It computes AK and retrieves SQN. Then computes MAC_2, (i) it checks if $MAC_2 = MAC$ and (ii) checks if SQN is in range $SQNUE < SQNHN$. If (i) and (ii) are the expected response, USIM computes RES then CK, IK then forwards them to the UE. The UE compute CK', IK'. UE returns RES and MAC_2 in $AuthResp$ message.

`Msg9. SEAF` $\rightarrow$ `AUSF: (NausfAuthnReq)`
The SEAF transparently forwards RES, MAC_2 as $NausfAuthnReq$ message to AUSF. There is an optional exchange of further EAP messages after message 9.

`Msg10. AUSF` $\rightarrow$ `SEAF: (NausfAuthnResponse)`
When AUSF receives RES and MAC_2, it verifies them by comparing RES with $XRES$, if $RES = XRES$ the authentication is considered successful by AUSF and it informs the UDM/ARPF. If not, it sends an error message to SEAF. Otherwise, it derives EMSK (K_{AUSF}) from CK' and IK', then calculates K_{SEAF} from K_{AUSF}. It sends K_{SEAF} and $SUPI$ to SEAF in $NausfAuthnResponse$ message, an EAP-Success. The key K_{SEAF} shall become the anchor key.

`Msg11. SEAF` $\rightarrow$ `UE: (N1Message)`
SEAF sends EAP success message in $N1Message$ with $ngKSI$ and the $ABBA$ parameter. The UE drives EMSK(K_{AUSF}) from CK' and IK', then calculates K_{SEAF} in the same way as the AUSF.

Phase 3: Re-synchronization

This phase updates the SQN on the HN side when SQN is out-of-sync. So $AUTN$ cannot be verified, the USIM informs the ME whether it is MAC or synchronization failure and it passes the

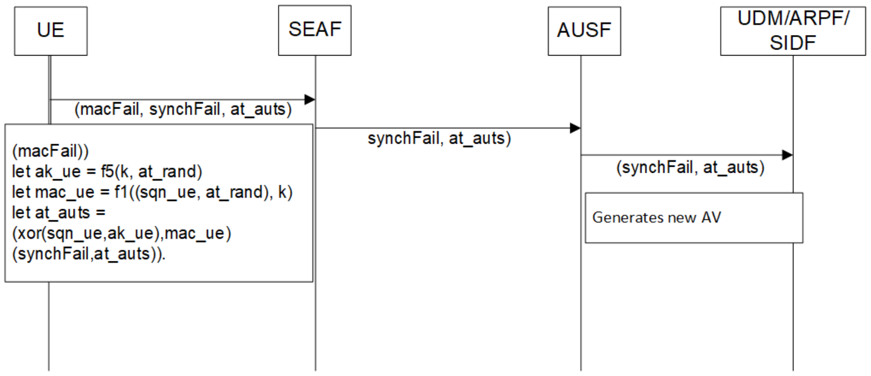


Figure 6.13: 5G EAP-AKA' Synchronization Phase Message Exchange Flow

AUTS parameter to the UE as shown in Fig. 6.13.
`Msg12. UE →  SEAF: (AuthFail)`
The UE sends *mac_failure* and *synch_failure* with AUTS to SEAF in *AuthFail* message.

`Msg13. SEAF →  AUSF: (NausfAuthFail)`
When SEAF receives message 12, it requests the UE for re-identification for mac_failure or initiate new authentication process for *Sync_failure* then sends *Synch_failure*, *AUTS* in *NausfAuthFail* message to AUSF.

`Msg14. AUSF →  ARPF: (NudmfAuthFail)`
Then AUSF sends *NudmfAuthFail* consisting of *Synch_failure*, *AUTS* and the *RAND* sent to the UE in message 6 to the UDM/ARPF. The ARPF gets *SQNUE* from AUTS, it checks if *SQNHN* is in the correct range and if the next *SQN* generated using *SQNHN* would be accepted by the USIM. If *SQNHN* is in the correct range, the UDM/ARPF generates a new AV otherwise, it verifies AUTS and upon a successful verification, the ARPF resets the value of the counter *SQNHN* to *SQNUE*. The UDM/ARPF sends new AV for the UE to the AUSF. The AUSF runs a new authentication procedure as per the specification.

6.8 Formal Verification of 5G EAP-AKA' Protocol

This section follows the same formal verification process as that defined in chapter 3. The modelling of 5G EAP-AKA' protocol is like that in section 6.3 and used same ProVerif processes as used in section 6.5 with some changes in the ProVerif code such as `fun PRF(key, key, bitstring, id):key`,

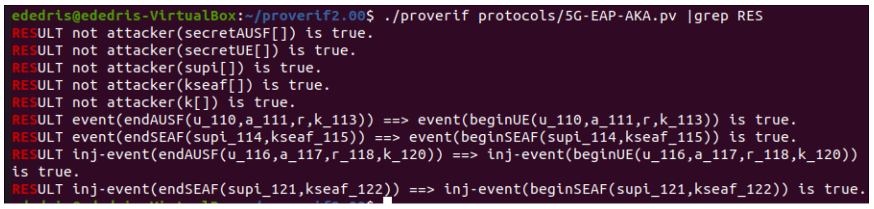

```
ededris@ededris-VirtualBox:~/proverif2.00$ ./proverif protocols/5G-EAP-AKA.pv |grep RES
RESULT not attacker(secretAUSF[]) is true.
RESULT not attacker(secretUE[]) is true.
RESULT not attacker(supi[]) is true.
RESULT not attacker(kseaf[]) is true.
RESULT not attacker(k[]) is true.
RESULT event(endAUSF(u_110,a_111,r,k_113)) ==> event(beginUE(u_110,a_111,r,k_113)) is true.
RESULT event(endSEAF(supi_114,kseaf_115)) ==> event(beginSEAF(supi_114,kseaf_115)) is true.
RESULT inj-event(endAUSF(u_116,a_117,r_118,k_120)) ==> inj-event(beginUE(u_116,a_117,r_118,k_120))
is true.
RESULT inj-event(endSEAF(supi_121,kseaf_122)) ==> inj-event(beginSEAF(supi_121,kseaf_122)) is true.
```

Figure 6.14: 5G-EAP-AKA' Safe ProVerif Results

Table 6.3: Secrecy Properties

Properties	UE	SN	HN
SUPI	H	H	H
K	H	H	H
KSEAF	H	H	H

`fun HMAC_SHA_256(key, bitstring):bitstring`. Modelled with model A, the four parties' protocol, `processUE` as UE, `processSEAF` as SEAF, `processAUSF` as AUSF, and `processARPF` as ARPF. EAP-AKA' is verified with `query attacker(secretAUSF), (secretUE), (supi)` and `(kseaf)`. Authentication is queried with `query u:host, a:host, r:nonce, kseaf:key, k: key; event(endAUSF(u,a,r,k))==> event(beginUE(u,a,r,k)).inj-event(endAUSF(u,a,r,k)) ==> inj-event(beginUE(u,a,r,k))`.

Full ProVerif description of 5G EAP-AKA' protocol is illustrated in Appendix B.1. When the protocol was modelled, the authentication on UE and HN holds as assumed by the standard on both non-injective and injective agreements as shown in Fig. 6.14. The SN authorization is checked in the process. The security properties of interest are K_{SEAF}, mutual authentication for the UE and HN, and privacy of communication between entities. The results also indicate that the secrecy of `secretUE, secretAUSF, supi, kseaf` hold as shown in Table 6.3.

6.8.1 Security Analysis

Protocol Security Analysis

The 5G EAP-AKA' protocol should meet certain security properties and the analysis is based on security requirements taxonomies presented in chapter 3, H is used when the property holds, and X is when the property does not hold. The security analysis is like that of the 5G-AKA protocol in section 6.3, Tables 6.3, 6.4, and 6.5, show that security properties hold. The difference is in the challenge/response and RES and MAC_2 and sent for verification to AUSF instead of RES^* in 5G-AKA. The `query attacker` on secrecy, aliveness, weak, non-injective, and injective agreement hold. Also, the mutual authentication, mutual key authentication, mutual key confirmation, key freshness, unknown key share, and key compromise requirements are achieved. This is due to 5G EAP-AKA' being modelled based on the secure 5G-AKA protocol, the difference in the message exchange could not cause an attack on its own.

Table 6.4: Security Properties for Set 1

Properties	$UE-SN$	$SN-UE$	$UE-HN$	$HN-UE$
Aliveness	H	H	H	H
Weak Agreement	H	H	H	H
Non-Injective Agreement	H	H	H	H
Injective Agreement	H	H	H	H

Table 6.5: Security Properties for Set 2

Properties	$UE - SN$	$SN - UE$	$UE - HN$	$HN - UE$
Mutual Entity Authentication	H	H	H	H
Mutual Key Authentication	H	H	H	H
Mutual Key Confirmation	H	H	H	H
Key Freshness	H	H	H	H
Unknown Key Share	H	H	H	H
Key Compromise Impersonation Resilience	H	H	H	H

Security Consideration

The EAP-AKA' faces similar attacks as 5G-AKA like the impersonation attack. However, EAP-AKA' also faces privacy attacks alleviated by using network name binding and its configuration should not depend on the requester's location unless cryptography is applied to the location data. DoS attack can only occur if the SN requests large numbers of authentication runs for UE, but re-synchronization and tracking/monitoring mechanisms should stop this type of attack by limiting the number of authentication attempts (Arkko et al. 2019). The issue of fake BS and non-repudiation is addressed by increasing home control. Even though 5G EAP-AKA' provides even better security than EAP-AKA, it is still affected by some of the 5G-AKA related attacks (Dehnel-Wild & Cremers 2018), (Koutsos 2019), and diameter protocol vulnerabilities (Enisa 2018).

Our analysis on both 5G-AKA and EAP-AKA' illustrates more detailed protocol modelling and evaluation, it discovered that some security goals and assumptions are underspecified or missing. It also showed the properties that are violated due to changing the channel assumption from secure to insecure. That is why the non-injective and the injective agreement did not hold as explained in section 6.5. We can conclude that 5G authentication protocols are still vulnerable to legacy attacks and with an increase in sophistication of cyber-attacks, it is inadequate to assume that the communication channel is secure without anticipating the adversary capabilities.

6.9 Summary

This chapter discussed 5G-AKA and EAP-AKA' for primary authentications. The 5G standard and EAP-AKA' RFC have been interpreted and analysed to identify all assumptions and security properties that need to be met to achieve the AKA in 5G. The 5G-AKA and EAP-AKA' protocols that provide security to the NAC-level have been modelled based on three and four entities models. Formal verification and systematic security evaluation of the protocols were conducted. With the found vulnerabilities, this study recommended some measures that later showed that the protocol can be secured. After formally analysing the primary authentication security protocols and they are considered secure to an extent under certain conditions. Considering how the primary authentication could affect the secondary authentication and service authorization, the next chapter introduces the service-level security and the underlying security protocols.

Chapter 7

Service-Level Security

7.1 Introduction

After primary authentication, the UE would have achieved mutual authentication and key agreement with the HN derived session key and used it to derive other keys to be used in securing communication with other 5G HN entities. With our proposed integrated solution, the UE will be able to get authorized to access services from the SP and get permission to cache and share content with other UEs. This chapter introduces service-level security and its proposed security mechanisms. It proposes secondary authentication, service access, data caching, and sharing authorization protocols. Some of the work in this chapter is also presented in (Edris et al. 2020*c*,*a*, In Press, 2021*a*).

The rest of this chapter is structured as follows. Section 7.2 discusses the secondary authentication framework. The proposed SAP-AKA protocol is presented in section 7.3. Section 7.4 presents the modelling of SAP-AKA protocol. The verification of SAP-AKA protocol is presented in section 7.5. In section 7.6, the FIdM in 5G is introduced. Section 7.7 presents the NS-FId model. Section 7.8 introduces the NS-FId protocol. In section 7.9, the proposed solution of the NS-FId protocol is presented. The NS-FId protocol modelling is presented in section 7.10. In section 7.11, the verification of NS-FId protocol is presented. Section 7.12 introduces data caching and sharing security. The proposed solution of DCSS protocol is presented section 7.13. Section 7.14 presents the modelling of DCSS protocol. The verification of DCSS protocol is presented in section 7.15. This chapter is summarised in section 7.16.

7.2 5G Secondary Authentication

To access the services from the third-party SP the UE must get authenticated by SP via SMF of the HN which acts as a pass-through authenticator. Having achieved primary authentication with the HN, the UE with some of its NAC credentials, and a NAS security connection with the AMF, requests a PDU service session establishment with target SP. The SMF process the request by getting UE's subscription data like SUPI, service agreement from the UDM via AMF, then UE is assigned a GPSI to use outside HN. The SMF also checks the validity of the UE's request in relation to user subscription, local or external service policies. The SMF may also check whether the UE has been authenticated and authorized by the target SP before. Then SMF redirects the UE to the SP to initiate authorization, the SP checks if the UE has been registered if not then it will initiate a secondary authentication. The secondary authentication process is discussed in the following sections. If the UE is already registered, it will continue with the authentication and authorization procedure explained in section 7.6.

The secondary authentication should be used to provide authentication between the UE and the SP, which adopts the EAP framework as specified in (3GPP 2020*f*) with a few modifications. It was specified for secondary authentication for DN, however, in this research, it is used for both internal SP and external SP. This is due to 5G's SBA that enables MNO to act as a network operator, infrastructure provider, and internal SP. In the case of external SP, this authentication procedure will provide optional authentication between UE and SP while the 5G-AKA and EAP-AKA' methods will provide authentication between the UE and the HN. It is assumed that UE would have registered with and the SP and MNO would have a service subscription of the UE. If the MNO and the SP are different, they should have an inter-operator service agreement for the UE.

After a successful authentication, the UE will be assigned a permanent ID, the UE's ID should be different from that used in primary authentication that will be used to request access to services. The ID will be either derived from the IP address or pseudo-randomized name given by the authenticator. Both the UE and SP would have authenticated each other and agreed on a session key to use during communication as well as achieving the authenticity and validity of the content. The UE and SPAAA will agree on a session key and the SPAAA will assign the UE with an external ID EID which is used in the next step of our security framework.

7.2.1 Problem Definition

As mentioned earlier, 3GPP recommends that the EAP framework should be used as the secondary authentication method in a fully active exposure scenario to external networks, however, the EAP has some limitations in achieving this objective. There is a restriction on using 5G security context such as keys and IDs outside the HN with non-3GPP access networks and the EAP framework's requirement of the authentication key AUT_Key to be preshared between the UE and the AAA server before the run of the protocol raises security problems. The Aut_Key is used to derive CK'/IK' and other following keys.

The IDs used in 5G primary authentication are not allowed to be used outside the HN that is why the SAP-AKA protocol uses Generic Public Subscription Identifier ($GPSI$), a publicly know ID which is later replaced by the (EID) created by SPAAA and securely assigned to the UE. The EAP derived keys; K_encr is used to encrypt AT_ENCR_DATA attribute such pseudonym IDs (identity privacy), K_aut issued to encrypt the AT_MAC attribute and K_re only used in the re-authentication process as per 3GPP and EAP specifications. MSK is used to protect the EAP-AKA packets for non-3GPP access interworking function and the EMSK is used in the derivation of 3GPP related access keys to secure the HN. Therefore, the proposed SAP protocol intends to provide mutual authentication, a session key, and an external ID to secure communication between the UE and SPAAA. The SAP solves the issue of not sharing primary authentication keys and security context with an external AAA server as per 5G specification as shown in Fig. 7.1 by using symmetric and asymmetric cryptography. After a successful run of SAP-AKA protocol, key K_{UE3A} is generated to be used by UE and SPAAA to secure their communication, and EID is created and assigned to the UE as its permanent ID. The EID and SPID are used in the derivation of the

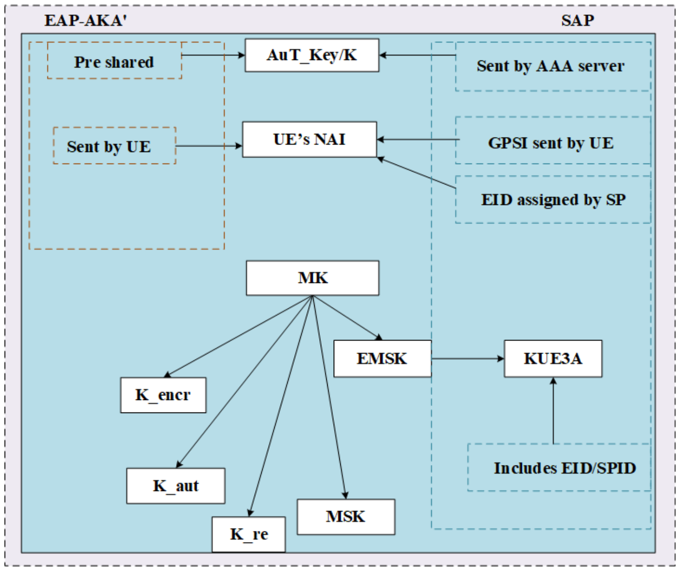


Figure 7.1: 5G EAP-SAP Problem Definition

session key to bind both the UE and SPAAA to the session and the key.

Most of the security assumptions and requirements are based on the specifications in TS 33.501 (3GPP 2020*f*), TS 33.402 (3GPP 2020*c*) and RFC 5448 (Arkko et al. 2018) as mentioned in chapter 6. The desired security properties that SAP-AKA protocol should meet are secrecy, confidentiality, integrity, authenticity, and privacy (3GPP 2020*f*). The UE must be assured that the SP is authorized by its HN, while the UE authenticates SP with the NAI through mutual authentication and key establishment. The HN must achieve a weak agreement with SP after key confirmation. The SP shall be able to authenticate the UE with $GPSI$ and pre-shared information with HN in the registration process. SAP-AKA protocol must ensure the secrecy of key K_{UE3A}, EID, and secure communication between UE and SP without using any of the primary authentication security contexts. Since no security context is shared with third-party SP compromising primary authentication should not compromise the secondary authentication. In addition, user's subscription privacy should also be ensured by providing confidentiality, anonymity, and untraceability.

7.3 Proposed Secondary Authentication Protocol

This section presents the proposed SAP-AKA protocol that leverages the EAP framework (Vollbrecht et al. 2004) as recommended by 3GPP (3GPP 2020*f*). This protocol uses the security parameters and EAP-AKA KDF (3GPP 2020*c*). It is an optional authentication that must be initiated by third-party SP when UE requests its services. The proposed SAP-AKA Protocol should provide mutual authentication and session key establishment between the UE and the SP.

Table 7.1: AT_KDF Parameters

Key	**Input**
MK	$KDF(PRF'(IK'\|CK', "SAP"\|Identity))$
K_encr	$KDF(MK[0..127])$
K_aut	$KDF(MK[128..383])$
K_re	$KDF(MK[384..639])$
MSK	$KDF(MK[640..1151])$
$EMSK$	$KDF(MK[1152..1663])$
K_{SEAF}	$KDF(EMSK, SNN)$
K_{AMF}	$KDF(KSEAF, SUPI, ABBA)$
K_{UE3A}	$KDF(EMSK, (EID, SPID))$

7.3.1 Architecture Overview

SLS recalls the following the entities from the system and security architectures defined in chapters 2 and 5, respectively, to participate in the secondary authentication:

- UE as the peer.
- H-SMF as pass through authenticator, the HN's SMF that communicates with the HN's AUSF/UDM/ARPF via AMF and SP entities such SP authenticator or backend server.
- SPAAA as the authentication server owned by SP. It grants authority and issue the security parameters used by the UE to access the service such as AV and EID.

Keys Derivation

The key derivation is performed according to EAP framework (Arkko et al. 2018) with *at_kdf_input* parameters as inputs (3GPP 2020*c*). The UE and the authentication server compute CK', IK' keys which are used together with PRF', SAP and identity as key derivation inputs *at_kdf_input*. PRF' is a pseudo-random function, SAP is a string indicating the type of protocol and identity is the UE identity used to derive a Master Key (MK). The MK is used to derive K_encr, K_aut, K_re, Master Session Key (MSK) and Extended Master Session Key ($EMSK$) as shown in Table 7.1. The K_encr is used for AT_ENCR_DATA and K_aut for AT_MAC attributes respectively while the K_re is applied during re-authentication if required. The MSK and $EMSK$ are derived after a successful EAP AKA challenge-response run for non-trusted and trusted non-3GPP access networks, respectively.

After a successful secondary authentication process using SAP-AKA protocol, the $EMSK$ key, UE, and SP identifiers are used as input parameters *at_kdf_input* with derivation function $KDF(EMSK, (EID, SPID))$ in deriving K_{UE3A} to secure communication between UE and SP in next stage of service authorization. The MSK is used to derive keys for non-trusted non-3GPP access interworking function. In addition, K_{AMF} is used to secure communication between UE and SMF provided by AMF, derived from K_{SEAF} using SUPI associated with NAI and Anti-Bidding down Between Architectures (ABBA) parameters for forward compatibility as *at_kdf_input* parameters $KDF(KSEAF, SUPI, ABBA)$ (3GPP 2020*f*) during primary authentication run. The K_{UE3A} is derived after a successful SAP-AKA protocol run between the UE and the SP. The key

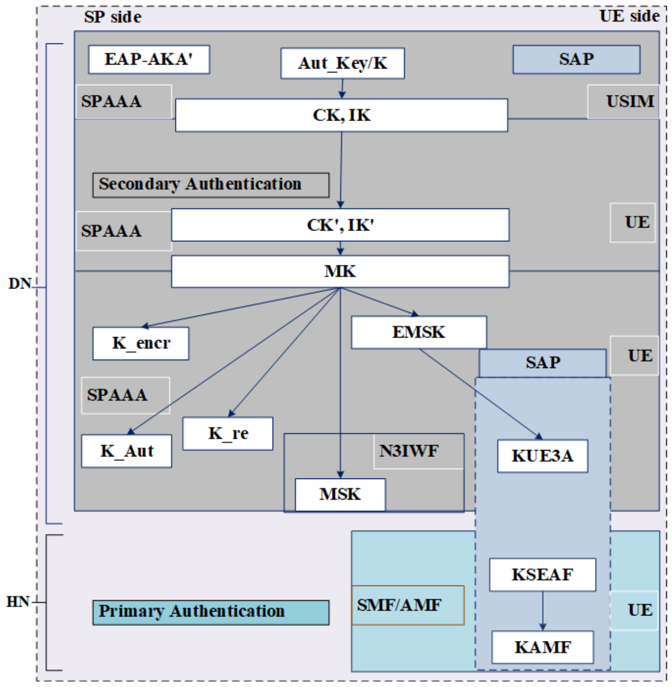


Figure 7.2: SAP-AKA Key Derivation

derivation and hierarchy are shown in Fig. 7.2, where $[0..n]$ denotes the substring from bit 0 to n used in the key derivation (3GPP 2020*f*).

7.4 Modelling of SAP-AKA Protocol

SAP-AKA protocol is modelled with UE, SMF, and SPAAA as entities, the communication between the SP and the UE goes through SMF, then SMF relays the message to UE or communicate to AMF and UDM for message interpretation and subscription data verification, respectively. The SMF performs as an EAP authenticator, it relies on the SP's AAA server to authenticate and authorize the UE's request to establish a service session. The SAP-AKA protocol should provide authentication and a session key for UE to access services from the DN. Its main purpose is to allow the UE 3GPP HN to communicate securely to authorize non-3GPP networks without compromising the security context such as SUPI to SP. The SMF check with UDM for any previous authentication or authorization sessions between UE and SP. If so, they can use the previous keys and trust for generating new session keys. The GPSI is used as UE's initial ID, globally known but it is not used in the AV as it is swapped with EID.

7.4.1 Protocol Message Exchange and Execution

This subsection explains the message exchange and execution of the SAP-AKA protocol, it consists of two phases: (i) service request and (ii) authentication. The protocol message exchange is illus-

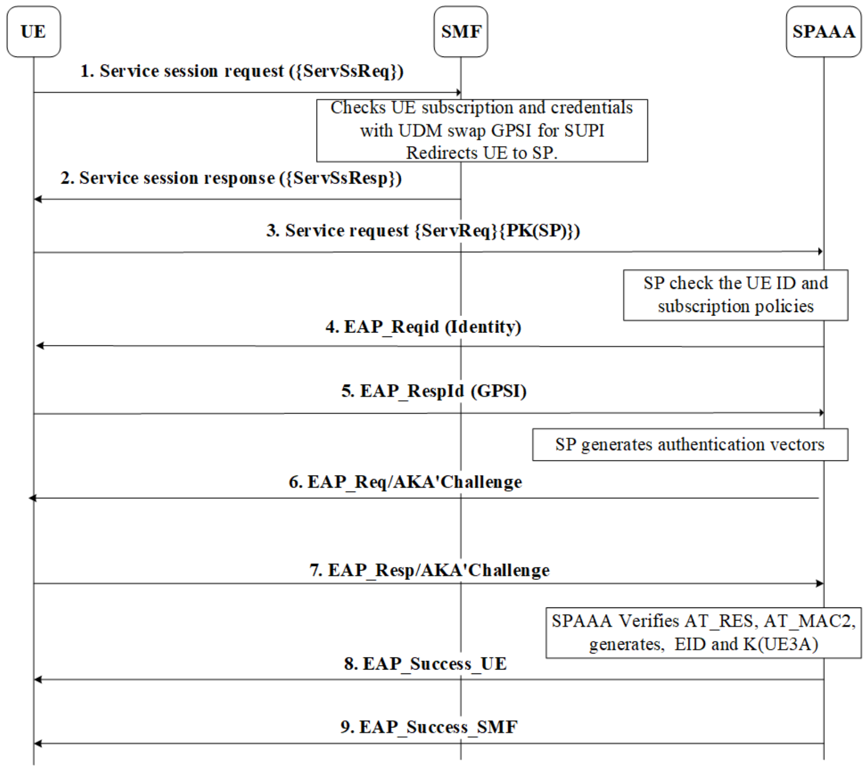

UE
SMF
SPAAA
1. Service session request ({ServSsReq})
Checks UE subscription and credentials with UDM swap GPSI for SUPI Redirects UE to SP.
2. Service session response ({ServSsResp})
3. Service request {ServReq}{PK(SP)})
SP check the UE ID and subscription policies
4. EAP_Reqid (Identity)
5. EAP_RespId (GPSI)
SP generates authentication vectors
6. EAP_Req/AKA'Challenge
7. EAP_Resp/AKA'Challenge
SPAAA Verifies AT_RES, AT_MAC2, generates, EID and K(UE3A)
8. EAP_Success_UE
9. EAP_Success_SMF

Figure 7.3: SAP-AKA Protocol Message Exchange Flow

trated in Fig. 7.3, with reference to notation in Table 7.2 and described as follows:

Phase 1: Service Request

`Msg1.UE→SMF:({ServSsReq})`

After the primary authentication, the UE sends a service session request message $ServSsReq$ via AMF to SMF in 5G HN, which includes the service name $Servname$ and session ID SID to request a service and session establishment. The $Servname$ is the identifier of the service, while SID is used for session management purposes by SMF.

`Msg2.SMF→UE:({ServSsResp})`

The SMF checks the user's subscription data, the primary authentication security status, and the context of the UE in HN's UDM. It checks if the SP provisioning the service to the user resides inside or outside HN and security context available. If its external SP, then SMF retrieves the UE global generic identifier $GPSI$ that corresponds with the UE's permanent Identifier SUPI and sends it to the UE along with the $SPID$ and SP's public key PK_{SP} in a service session response message $ServSsResp$. SMF redirects the UE to SP for authentication and service authorization.

`Msg3.UE→SPAAA:({ServReq}{PKSP})`

Then UE sends a service request message $ServReq$ to SP, it includes service name $Servname$, session ID SID encrypted with SP public key PK_{SP}.

Table 7.2: SAP-AKA Protocol Notation and Description

Notation	**Description**
SPID	SP identity
SID	service/session identity
DNN	service code: SPID (NAI)
NAI	(SID, SPID)
Aut_Key/K	preshared key between (UE, SP)
K_{AMF}	session key for UE and AMF/SMF
RAND (AT_RAND)	random nonce challenge
EID	UE permanent identity
GPSI	UE generic identity
AUTN(AT_AUTN)	(SQNSP $\oplus$ AK,MAC)
MAC,MAC2(AT_MAC)	f1(K, (SQNSP, Rand))
RES (AT_RES), XRES	f2(K, Rand)
CK	f3(K, Rand)
IK	f4(K, Rand)
AK	f5(K, Rand)
CK' IK'	ik, ck, dnn, (sqn $\oplus$ak)) ik, ck, dnn, (sqn$\oplus$ak))
EMSK	KDF((CK', IK'), (SNN, SQN $\oplus$ AK))
SQN	sequence number
PK_{SP}	SP public key
K_{UE3A}	KDF(EMSK, (EID, SPID))
h(x)	hash value of message x
{x}{k}	message encrypted with key K

Phase 2: Authentication

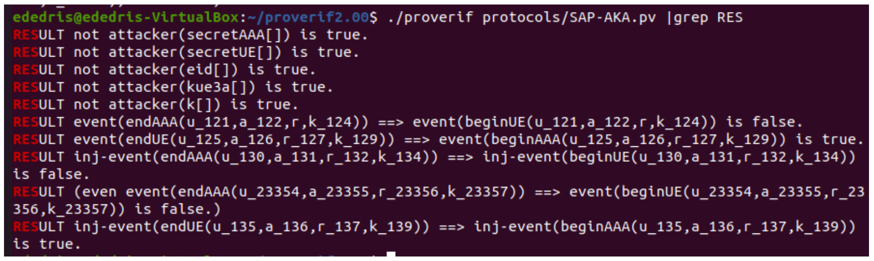
`Msg4.SPAAA→UE:(EAP_Reqid)`
The SPAAA verifies the `ServReq` by checking the details of the *Servname* and *SID* in its database which includes the services and agreement policies with HN, if they are valid then it sends an EAP request message requesting the UE to identify itself, initiating an EAP-AKA message exchange.

`Msg5.UE→SPAAA:(EAP_RespId)`
When the UE receives message 4, it sends an EAP response identity message to the SPAAA, including its generic identity `GPSI`.

`Msg6.SPAAA→UE:(EAP_Req/AKA'Challenge)`
After receiving message 5, the SPAAA checks *GPSI* and policies, then sends EAP request to start AKA challenge with the UE, this includes `(AT_RAND, AT_AUTN, AT_MAC, AT_KDF, AT_KDF_INPUT)`, random number, authentication challenge AUTN, MAC, KDFs, and the input for generating keys.

`Msg7.UE→SPAAA:(EAP_Resp/AKA'Challenge)`
The UE verifies the AUTN, the MAC, checks the token challenge and MAC values, for freshness and message integrity. If they match, it responds with an EAP response message that includes `(AT_RES, AT_MAC2)` a response *RES* to the challenge sent in message 6 and a new MAC2.

`Msg8.SPAAA→:(EAP_Success)`
The SP verifies MAC2 and checks the response from UE received in message 7, if they match it



```
ededris@ededris-VirtualBox:~/proverif2.00$ ./proverif protocols/SAP-AKA.pv |grep RES
RESULT not attacker(secretAAA[]) is true.
RESULT not attacker(secretUE[]) is true.
RESULT not attacker(eid[]) is true.
RESULT not attacker(kue3a[]) is true.
RESULT not attacker(k[]) is true.
RESULT event(endAAA(u_121,a_122,r,k_124)) ==> event(beginUE(u_121,a_122,r,k_124)) is false.
RESULT event(endUE(u_125,a_126,r_127,k_129)) ==> event(beginAAA(u_125,a_126,r_127,k_129)) is true.
RESULT inj-event(endAAA(u_130,a_131,r_132,k_134)) ==> inj-event(beginUE(u_130,a_131,r_132,k_134))
is false.
RESULT (even event(endAAA(u_23354,a_23355,r_23356,k_23357)) ==> event(beginUE(u_23354,a_23355,r_23
356,k_23357)) is false.)
RESULT inj-event(endUE(u_135,a_136,r_137,k_139)) ==> inj-event(beginAAA(u_135,a_136,r_137,k_139))
is true.
```

Figure 7.4: SAP-AKA Attack ProVerif Results

generates session key K_{UE3A} for UE and SPAAA and a permanent Identifier EID for use during service authorization procedure. The session key K_{UE3A} and EID are encrypted with K_enc key in the EAP success message sent to the UE.

`Msg9.SPAAA→UE:(EAP_Success)`

The SP also sends an *EAP_success* message to the SMF that authentication was successful. Then SMF sends a message to UDM to update the UE profile in the HN.

7.5 Formal Verification of SAP-AKA Protocol

This section follows the same ProVerif process as that defined in chapter 3. The modelling of SAP protocol uses ProVerif code like that used for 5G EAP-AKA' protocol in chapter 6 with some changes, full Proverif protocol syntax in Appendix C.1.

7.5.1 Formal Analysis of SAP-AKA Protocol

The SAP-AKA protocol is simulated with processes `procUE(hostU)` as UE, `procAAA(hostA)` as SPAAA, and `procSMF(hostS)` as (SMF). When the protocol was modelled in ProVerif, an attack was found on the authentication protocol between the UE and SP as non-injective and injective agreements as shown in Fig. 7.4. The security properties of interest are secrecy and authentication. The results also indicate that the secrecy of `secretAAA`, `secretUE`, `eid` and `Kue3a` do hold.

The event endAAA means that the SPAAA has completed the protocol, that the UE received message 6 from SPAAA and UE responded with message 7. These events take `at_rand: nonce`, `at_autn: bitstring`, `at_mac: bitstring`, `at_kdf:bitstring`, `at_kdf_input:bitstring` parameters as arguments. It checks the sent `at_mac, at_autn` and computes the `at_rand`. If the arguments are true then `at_res, at_mac2` are sent otherwise it sends authentication failure. The study would like to prove the following correspondence.

```
(*Check authentication between UE and AAA*)
query u: host, a: host, r: nonce, kue3a:key, k: key;
event(endAAA(u, a, r, k)) ==> event(beginUE(u, a, r, k)).
inj-event(endAAA(u, a, r, k)) ==> inj-event(beginUE(u, a, r, k)).
```

The direct proof of this correspondence does not hold in ProVerif because message 7 was sent and message 8 was received hence the failure of the authentication. The study also tries to prove and con-

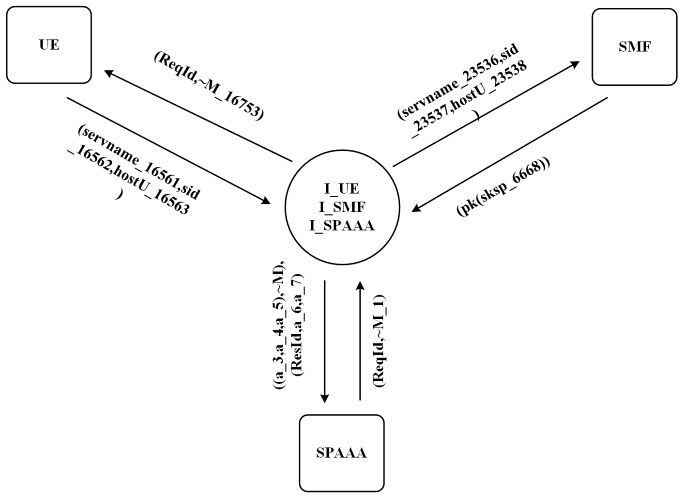


Figure 7.5: SAP-AKA Attack Trace

clude the desired correspondence by noticing that message 7 which has `at_res, at_mac2` as argument that cannot be executed before `at_rand: nonce`, `at_autn:bitstring, at_mac:bitstring`, `at_kdf:bitstring`, `at_kdf_input:bitstring` has been sent in message 6, that is, message 6 has been executed. Which does not hold in ProVerif showing false.

7.5.2 The Attack Against SAP-AKA Protocol

ProVerif produced results indicating there was an attack on the protocol as shown in Fig. 7.4. As attack derivation represents the attacker's action while trace represents the real attack as an executable trace of the considered process as explained in chapter 3. The attack derivation and trace sequence of steps, input, and outputs are explained in this subsection.

Attack Derivation and Trace

The attacker I starts by eavesdropping on the communication between entities, impersonates the UE by continuing the protocol with $SPAAA$, which completes the protocol with the attacker instead of UE. The attacker's actions are illustrated in Fig. 7.5 and explained concisely in derivation and trace steps below, some text is omitted for simplicity, full trace output in Appendix J:

- Steps 1 and 2 indicate that the attacker may have received the SP's key pair in output {7} as the attacker impersonates UE. The input {1}, {3} corresponds with the creation of private key `sksp_16418` and session key `kue3a_16417`. Output {7} corresponds with the creation of public key that the attacker stores in a fresh variable `~M_16501` for later use. Input {55} corresponds to a session of the SP `(copy a_16416)` with the attacker, the attacker sends its copy of service request created at {54} to the server. The server responds with output 57 asking for ID, attacker sends its ID input 58. The session key was obtained at {92} after {91} `(event beginAAA(u,hostA,at_rand_53,k_aut_67))`.

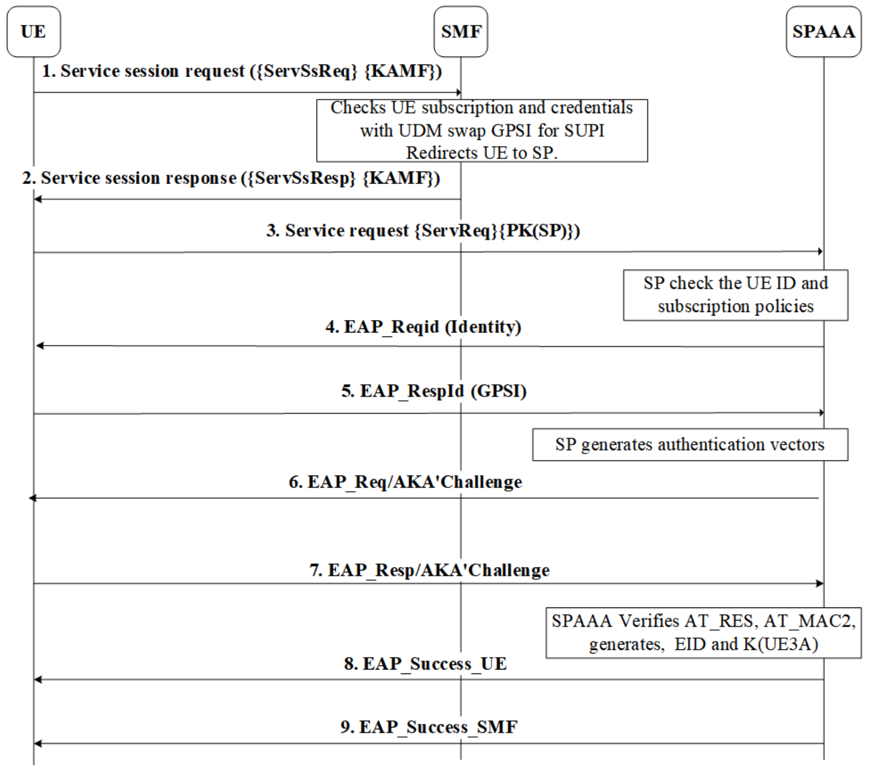
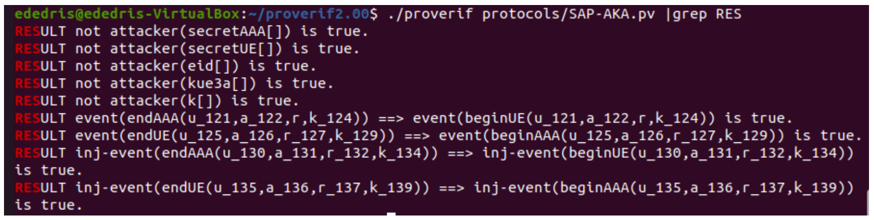

UE
SMF
SPAAA
1. Service session request ({ServSsReq} {KAMF})
Checks UE subscription and credentials with UDM swap GPSI for SUPI Redirects UE to SP.
2. Service session response ({ServSsResp} {KAMF})
3. Service request {ServReq}{PK(SP)})
SP check the UE ID and subscription policies
4. EAP_Reqid (Identity)
5. EAP_RespId (GPSI)
SP generates authentication vectors
6. EAP_Req/AKA'Challenge
7. EAP_Resp/AKA'Challenge
SPAAA Verifies AT_RES, AT_MAC2, generates, EID and K(UE3A)
8. EAP_Success_UE
9. EAP_Success_SMF

Figure 7.6: Improved SAP-AKA Protocol Message Exchange Flow

- Traces 1 and 2 show that correspondence non-injective and injective fail as the attacker's query goal is reached, that is the SP end the protocol thinking it was talking to UE while UE never run the protocol with the SP.

7.5.3 Improved Version of SAP-AKA Protocol

This subsection discusses the changes that were made in modelling of SAP-AKA protocol to address security threats discovered in the old version of the protocol. To improve the protocol instead of



```
ededris@ededris-VirtualBox:~/proverif2.00$ ./proverif protocols/SAP-AKA.pv |grep RES
RESULT not attacker(secretAAA[]) is true.
RESULT not attacker(secretUE[]) is true.
RESULT not attacker(eid[]) is true.
RESULT not attacker(kue3a[]) is true.
RESULT not attacker(k[]) is true.
RESULT event(endAAA(u_121,a_122,r,k_124)) ==> event(beginUE(u_121,a_122,r,k_124)) is true.
RESULT event(endUE(u_125,a_126,r_127,k_129)) ==> event(beginAAA(u_125,a_126,r_127,k_129)) is true.
RESULT inj-event(endAAA(u_130,a_131,r_132,k_134)) ==> inj-event(beginUE(u_130,a_131,r_132,k_134))
is true.
RESULT inj-event(endUE(u_135,a_136,r_137,k_139)) ==> inj-event(beginAAA(u_135,a_136,r_137,k_139))
is true.
```

Figure 7.7: SAP-AKA Safe ProVerif Results

assuming that the exchange between the UE and HN is secure, session key K_{AMF} derived during primary authentication was used. The communication between the UE and SMF/AMF is encrypted with key K_{AMF} within the HN as shown Fig. 7.6. When the protocol was modelled again the secrecy and authentication were found to hold on both non-injective and injective agreements as shown in Fig. 7.7. The implicit authentication between the SP and HN is also checked in the process. The results also indicate `no attacker ([])` is true on `SecretAAA`, `SecretUE`, `eid` and `Kue3a`.

7.5.4 Security Analysis

Protocol Security Analysis

The proposed security protocols should meet certain security requirements and the analysis for the improved version of SAP-AKA is based on the properties of sets 1 and 2 as explained in section 3.

Analysis based on security properties of set 1 is as follows:

- Secrecy: This is achieved since key K_{UE3A} is never revealed to the attacker. By using XOR and anonymity keys to protect the parameters used in derivations of keys in transit and in storage. The use of functions f1, f2, f3, f4, f5 to provide privacy protection of challenges/response of the data. By achieving this property also covers confidentiality and privacy of the protocol.
- Aliveness: The SP obtain the aliveness of the UE at SMF, which is a non-injective agreement on NAI from the SP's point of view with the subscribers. But also, the SP should have injective agreement on K_{UE3A} with the subscribers, which gives recent aliveness as a result.
- Weak Agreement: This is achieved when HN achieves non-injective agreement on EID with UE as it is the ID. Also, the SP achieves weak agreement with HN after the key confirmation as the key includes $SPID$ and $GPSI$.
- Non-injective Agreement: The UE obtains non-injective agreement on NAI with its SP after key confirmation of K_{UE3A}. Moreover, since $GPSI$ also is assigned by HN, an agreement on EID is an agreement on $GPSI$. The HN obtain non-injective agreement on EID with the SP after EID is assigned to UE by SP. The injective agreement on K_{UE3A} from the SP towards the UE, also guarantees that UE is attached to an authorized SP, this is achieved since K_{UE3A}'s derivation includes nonce from SP and NAI. Assuring the UE that SP is trusted and the authentication UE-SP holds.
- Injective Agreement: The injective agreement on K_{UE3A} for different pairs of parties is achieved when the K_{UE3A} cannot be derived twice for the same session. The K_{UE3A} derivation also includes `at_rand`, which gives an assurance on K_{UE3A} from the SP to the UE. The injective agreement on K_{UE3A}, which is bound to $SPID$ provided by the HN assuring the UE that SP is known and trusted. The UE obtain the injective agreement on K_{UE3A} with the SP to assure that the session was authorized by the HN. Also, it achieves the same trust from UE as the event correspondence holds.

Analysis based on security properties of set 2 is as follows:

- Mutual Entity Authentication: The UE is authenticated to SP if `at_res` and `at_mac2` are valid and HN implicitly. Since the $SPID$ and $GPSI$ are included, it enforces implicit authentication upon a successful authentication.

- Mutual Key Authentication: Since the SPAAA sends the `at_kdf:bitstring`, `at_kdf_input:bitstring` strings to UE with key derivation parameter, which fulfils this requirement.
- Mutual Key Confirmation: After the successful AKA roundtrip between the entities ending with SPAAA sending *Success* message and K_{UE3A}, it enforces this requirement.
- Key Freshness: ProVerif has no function to check key freshness, however, during the authentication process the UE checks the `at_autn` freshness and computes `at_rand`. Key K_{UE3A} is a result of the input request that was sent by SPAAA to UE in during the current protocol session, hence the input is fresh and the key is fresh.
- Unknown-Key Share: The reachability property in ProVerif is used to check aliveness. The entities' IDs and `at_kdf: bitstring, at_kdf_input:bitstring` prevent this attack. The inclusion of $GPSI$, $SPID$ in the authentication process and the $GPSI$ in the derivation of `K_aut`, also proves this requirement. Moreover, K_{UE3A} is only sent to UE after the RES and MAC_2 verification by SPAAA.
- Key Compromise Impersonation Resilience: The K_{UE3A} is implicitly authenticated and its secrecy holds. It remains confidential since the key derivation input was sent by SPAAA in a secure communication exchange as defined by RFC 5448. Hence, forward secrecy and post-compromise secrecy might hold. Since SAP-AKA uses EAP framework and EAP-AKA' does meet these security requirements as no other key is involved in the derivation of K_{UE3A}; therefore, deriving past and future keys by the attacker based on `at_kdf:bitstring`, `at_kdf_input:bitstring` is not possible.

Security consideration

The service session establishment procedure falls out of the scope of this research, for IP based and CCN based procedure refer to (3GPP 2020*f*) and (Ravindran 2019), (Ravindran et al. 2017) respectively. Now that the UE has been authenticated to access the SP network, further authorization to access services is required. When the UE registers with the network it shares some data with HN as per 3GPP standard and with SP as per SLA and QoS. The HN gets in agreement with SP if they are different parties. After the authentication, the SPAAA will create a session key for UE and SPAAA.

7.5.5 Summary

This section proposed the SAP-AKA protocol an optional authentication as part of the service authorization procedure for UE requesting access to services provided by SP via its HN. Before other authorization procedures are presented, the next section discusses FIdM in 5G and its role in service authorization.

7.6 Network Service Federated Identity Management

This section explores FIdM and how it can be integrated with authentication and authorization procedures in 5G. It proposes a NS-FId architecture model that complements the 5G system and security architectures, facilitates authentication and authorization with SSO in different scenarios.

The model leverages 3GPP AKA mechanism and OAuth2 framework to provide federated authorization.

It is crucial for 5G to create a chain of trust between the UE, HN, and DN during the authorization process. With 5G supporting multiple shareholders and multi-tenancy on the infrastructure, seamless connectivity and secure access is a big challenge, while the interoperable ACL implementation at different levels might be difficult. ACL mechanisms must be able to identify and authorize the UE requesting access to 5G NS via different access networks. Therefore, there is a need for security contextualization propagation between the MNO and the SP such as a service-oriented authentication and authorization mechanism to facilitate network slicing and service security. Additionally, how will services from multiple providers be delivered to users in different security domains? 5G also is faced with the issue of incompatibility of 3GPP AKA framework with virtualization frameworks for network access and network slices provisioning. This can be solved with Identity and Access Management (IAM) solutions that facilitates FId and SSO via trusted third-party (Norma 2016), (5GPPP 2017) (VirtuWind 2017).

The use of federated authentication and authorization can facilitate adaptable security management, precise UE data tracking, seamless connectivity, and enablement of security management delegation. This would reduce the use of secondary authentication protocol, improve infrastructure resources optimization and third-party trust whereby ID management, authentication, and authorization for the UEs accessing SP services are provided by a third-party provider. Related work discussed the possibility of FIdM in 5G without proposing a robust, unified, multi-purpose solution to complement the complexity of 5G's SBA which will be supported by various technologies that need security guarantees, authentication, and ID protection. Additionally, FIdM was being suggested for network slicing and IoT but not in network access and service authorization where the author believes that FIdM will be an ideal solution in supporting 5G's objectives.

7.6.1 Related Work

With 5G standardization still to be finalized, the integration of FIdM in 5G can benefit NF service provisioning (5GPPP 2017). With MNOs using federated solutions to provide users with NF as a service (Mwangama et al. 2015) but federated network access is yet to be explored. However, FIdM in cloud services, social networks, and IoT has been explored extensively, with the focus on data storage, cloud access, and social media. The FIdM for 5G is not well investigated, so this study intends to address federated security for NS access, services authorization, and network slicing in 5G.

Identity and Access Management

As users roam across different networks, it is becoming a big challenge for MNOs to provide robust ACLs, security, and session continuity but the advancement of IAM technologies can provide security, improve user experience, and privacy. With SSO implementation in 5G, a user can use a single common set of credentials to gain access to multiple services in different security domains for technical interoperability across systems. Whereby the UE relies on a single ID, security association, and trust to get authenticated and authorize to services in the HN and SP. This would allow UE seamless mobility, handover, and connectivity across networks, hence reducing multiple authentication procedures every time the UE requires access to restricted services in multiple domains. IAM system is responsible for the identification, authentication, and authorization of a user accessing services by enforcing ACLs and management of the user's digital ID linked to multiple accounts.

Each account is assigned different ACLs and security contexts. It also manages the access database by updating users' access rights and storing user's information such as ID, keys, and certificates. This is facilitated with SSO, multi-factor authentication, and privileged access management.

Federated Identity Management

With FIdM, users can use the same identification information while accessing resources from multiple SPs in different domains in form of a FId. The SP facilitates user's identification, authentication, and authorization procedures processed and managed by a third-party authority the Identity Provider (IdP). The IdP creates and manages the user's ID data as well as facilitating MNO/SP and user's connection after a successful authentication. The user ID and security context are sent to the SP for access decision-making (Bertino & Takahashi 2010). The IdP and SP must agree on security policies, cryptographic schemes, and authentication methods used in the identification and authentication of the user. However, there is an issue of a single-point failure of a centralized ID management which is solved by enabling the sharing of the IdP's functions among several local IdPs in different SP security domains (Fang & Ye 2018).

After federated authentication, the user's authorization to the services relies on the authenticated credentials. The user IDs are mapped to IdPs in different domains based on PKI, trust, and service agreements. FIdM is enabled by mechanisms such as SSO, SAML (Hughes et al. 2005), OAuth2 (Dick 2012), OpenID (Recordon & Fitzpatrick 2006) and some of the entities used are IdP authentication server, IdP ID database server, SP authorization server, SP federation server, and resource server. In 5G, users can gain access to services in the HN or DN using the federated authentication method and OAuth2 framework providing a secure delegated access. With the UE accessing resources on a resource server on behalf of a user without sharing their credentials with the resource server. The OAuth2 can also ensure that only authorized NFs are granted access to a service offered by another NF in 5G (3GPP 2020*f*), hence enabling FIdM in 5G.

Heterogeneous Network Service Access

5G will require enhanced User ID protection for UEs accessing NS via ng-RAN and pseudo-identifiers will be used to provide anonymity and protection from UE tracking based on IDs. While robust key generation and management are required to prevent key leakage and unknown key share. In legacy systems, the user's ID and access verification are carried out in a controlled security message exchange process within the HN by MNO. However, 5G introduced an ID management domain to support the use of alternative UE IDs during authentication from semi-trusted networks and an IAM domain to address access control based on SCC. Third-party will also be able to authenticate and authorize the UE via third-party and DN functions of the 5G system. Moreover, the management domain was introduced to manage services, security, UE, and virtualized environment domains (5GPPP 2017).

In 5G, the MNO would communicate with the SP via IdP to identify and manage the UE IDs, enabling ID interoperability between MNO and SP and allowing the UE to access services even in cases where roaming agreements between networks do not apply or during D2D scenarios. This would also enforce accountability and non-repudiation by utilizing the linkage between UE and its ID, giving assurance to the VN or SP that UE accessing a service is genuine (5GPPP 2017). UDM will manage users' profiles, so should be flexible and interoperable to support unified IAM and FIdM (Fang et al. 2018).

The use of a single digital ID for multiple profiles would give the user access to multiple systems, increasing mobility availability and management of overlapping digital IDs from different security domains, hence achieving 5G objectives and meeting users, MNOs, and SPs security requirements. The trust between shareholders is used to provide services and implement ID verification, ACL, attributes, and capabilities sharing through the federated delegation process. However, the user would need an ID and a combination of other security inputs such as cryptographic primitives and ACLs for authentication and authorization, while the security level would depend on the security policies. Therefore, FIdM would be ideal in supporting the secure access and delivery of NS in the 5G network.

7.7 Proposed Network Service Federated Identity Model

IAM solutions are needed to support UEs and network devices with different security capabilities and attributes for 5G, legacy systems, and non-3GPP technologies. 5G Infrastructure Public Private Partnership (5GPPP) defined new methods to enable the use of UE IDs in network slicing and IAM solutions (5GPPP 2017) for 5G including FIdM. This study proposes a NS-FId model that leverages 3GPP system (3GPP 2020*g*), security (3GPP 2020*f*), SBA (3GPP 2020*d*) to facilities FIdM in 5G, complementing the relevant 5G standards. The user federated authentication to NS provided by SP would be handled by trusted IdP.

7.7.1 Model Architecture

As 5G is in the final stages of standardization, there are still crucial security issues that need addressing such as authentication, authorization, and IAM mechanisms. Since 3GPP specifies that security parameters and ARPF generated AV must be used within the 5G network and SUPI should not be shared outside the HN. Therefore, to address those mentioned issues a new model is needed, to enable seamless and secure connectivity across multiple domains. For the architecture, the proposed NS-FId model recalls the following entities defined in chapters 2 and 5, these entities might have more than one role:

- UE: End user and principal accessing the service.
- HSMF: SMF function that communicates with the AAA servers in HN and SP entities. Since the IdP and SP cannot interact directly with home security entities
- HNAAA: AAA servers that carry out the primary authentication with UE and it consists of AUSF, UDM, and ARPF.
- IdP: Creates, manages FId, and performs federated authentication with the UE. Issues ID tokens and acts as federated active directory and ID database servers.
- SPAAA: Acts as SP AAA servers, the SP is also part of the transaction as it issues EID, authorization grants, issue access/fresh tokens for service access.
- SS. Server hosting the services, verifies access token to grant the UE access to the protected services.

The proposed model adopts federated AAA, database servers, and 5G security architecture entities to support FIdM, redefine UE's ID parameters, and the sharing of security contexts such as keys and IDs outside of the HN. The user's data is stored in a federated server and UDM is used to process the user's federated identification supporting FIdM and 5G objective of minimal exposure of 5G security context. The main role of the IdP is to perform federated authentication on behalf of the MNO and SP, based on FId that is later used in creating ID and access tokens for service authorization of the UE. The UE must register and authenticate the HN as per the 5G standard and then performs federated authentication and authorization to access the services in HN or DN. Additionally, the following Identity Provider Broker (IdPB) and Service Provider Broker (SPB) are introduced to expand on the proposed model:

- IdPB: Is an ID broker that maps ID attributes from different IdPs to a single user and uses trust relationship to connect SPs to IdPs in different domains (Austin et al. 2013). It also translates tokens into different formats and enables cross-platform federated authentication.
- SPB: Is affiliated with SPs that the user requests access to for a particular service, linking the user to multiple services (Research 2013). It also manages the HN and SP communication with SPs, IdP, or the IdPB. In this model, three phases are defined to complete a FId procedure, and they are presented in the following sections.

7.7.2 Registration

In this phase the UE registers to the MNO in the HN, its subscription and security data are stored in the UDM. The UE's details including ID, access policies, and service agreements are registered to the SP by the MNO. Additionally, the MNO and SP register to IdP, agreeing on the mechanisms to be used such as IDs, PKI, ACL, cryptographic parameters for authentications and authorization procedures. The MNO, SP, and IdP share the necessary security information to their counterparts in form of services, policies, security secrets, IDs, user attributes, and other credentials to purposely support the process.

7.7.3 Authentication

As presented earlier in chapter 6, the primary authentication is the initial stage of the authentication procedure for the UE to access the HN via the SN, it gets authenticated and authorized to use the MNO resources. The second stage authentication comprises two authentication procedures, the first one is the secondary authentication that occurs when the UE requires resources that are provided by the third-party SP rather than the MNO and the SP is not registered to the same IdP as the MNO hence completing the registration phase. The UE, HN, and SP participate in the secondary authentication which is based on the EAP framework as discussed in section 7.3.

The other procedure of the second authentication stage is the federated authentication that occurs between UE, IdP, and SP when the FIdM is enabled managed by IdP, it uses some of the security context and UE's subscription data from the previous authentication procedures such as GPSI and access policies. It is based on FId, the SP generates authorization code that the UE presents to IdP, which creates the ID and ID token that are used in the authorization of process that facilitates SSO relying on the registration information.

Similarly, a Generic Bootstrapping Architecture (GBA) protocol enabling the reuse of existing security procedures such as primary authentication to grant access to application services facilitated

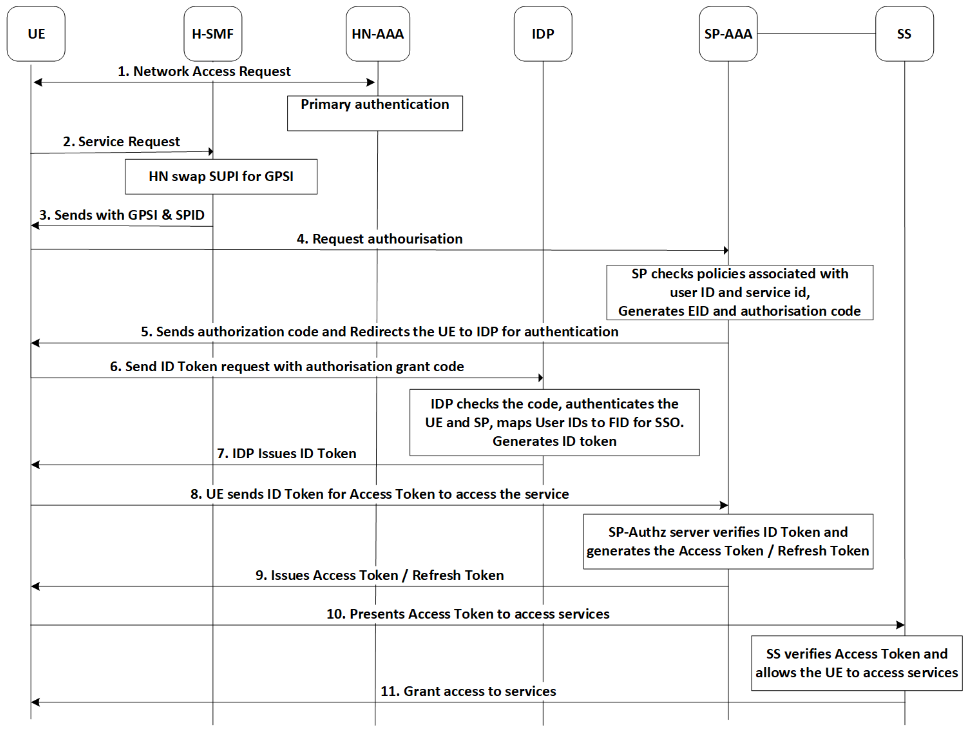

UE
H-SMF
HN-AAA
IDP
SP-AAA
SS
1. Network Access Request
Primary authentication
2. Service Request
HN swap SUPI for GPSI
3. Sends with GPSI & SPID
4. Request authourisation
SP checks policies associated with user ID and service id, Generates EID and authorisation code
5. Sends authorization code and Redirects the UE to IDP for authentication
6. Send ID Token request with authorisation grant code
IDP checks the code, authenticates the UE and SP, maps User IDs to FID for SSO. Generates ID token
7. IDP Issues ID Token
8. UE sends ID Token for Access Token to access the service
SP-Authz server verifies ID Token and generates the Access Token / Refresh Token
9. Issues Access Token / Refresh Token
10. Presents Access Token to access services
SS verifies Access Token and allows the UE to access services
11. Grant access to services

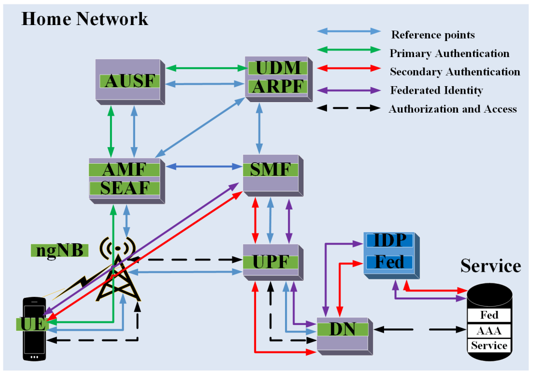
Figure 7.8: 5G NS-FId Model Generic Message Flow

by the application and bootstrapping functions in 5G was developed by 3GPP (3GPP 2020*e*). It verifies if the UE was verified correctly with AV in the UDM, GBA protocol can extend the capabilities of federated protocols in protecting NF to securely expose security context and events to third-party AF using NEF in 5G (3GPP 2020*f*).

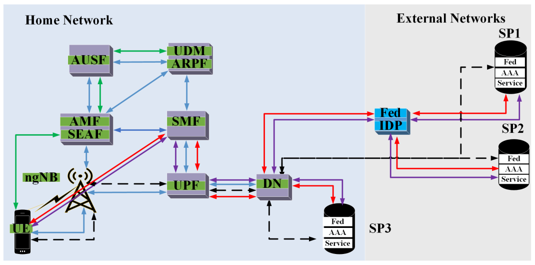
7.7.4 Authorization

With authorization, this study adopts the OAuth2 framework (Dick 2012), it has also been standardized for authorizing the access to NF application services in 5G (3GPP 2020*f*). The OAuth2 is integrated with federated authentication to facilitate an authorization process that uses UE credentials and ACL. The UE is issued with an ID token by the IdP that it presents to the SP which issues an access token in return that UE presents to SS to be granted access to the services. Additionally, the SP can issue a fresh token as an optional token to be used if the access token is invalid, expired or another activity is needed (Dick 2012). Even though requests from the AF can be authorized by the NEF using the OAuth2 method, however federated authentication, and authorization methods with OAuth2 can be more efficient and scalable.

The generic federated authentication and authorization steps of any NS-FId model are initiation, authorization grant, identification, authentication, authorization, and access as illustrated in Fig. 7.8.



(a) Model A



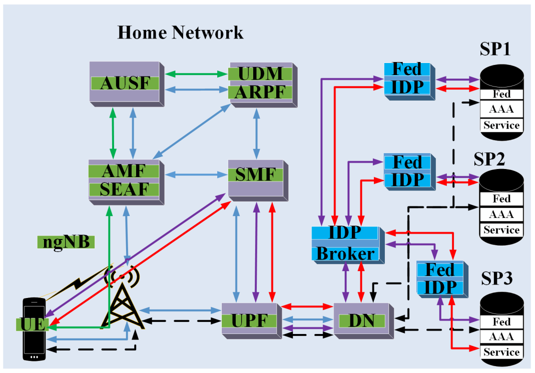
(b) Model B

Figure 7.9: FIdM Models A and B

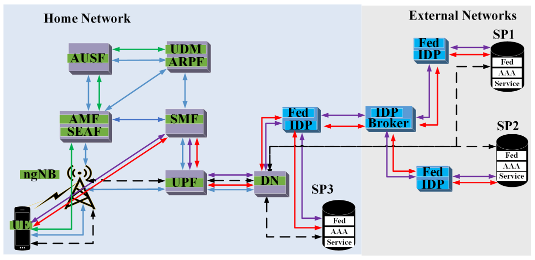
7.7.5 Use case Scenarios

Depending on the network architecture and service provisioned, FIdM in a mobile network can be implemented in many ways supported by the use of trust and PKI mechanisms. MNOs, SPs, IdP, UEs, and other shareholders are accommodated in the models that illustrate different scenarios and they are as follows:

- Model A: MNO is the SP in a network using a single IdP illustrated in Fig. 7.9.
- Model B: Multiple MNOs and SPs in a network using multiple IdPs through IdPB illustrated in Fig. 7.9.
- Model C: Multiple MNOs and SPs across multiple networks using a single IdP illustrated in Fig. 7.10.



(a) Model C



(b) Model D

Figure 7.10: FIdM Models C and D

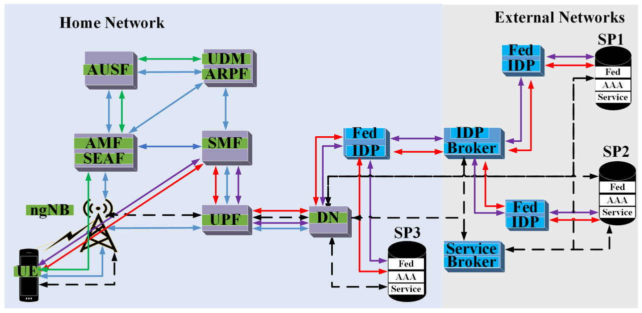


Figure 7.11: FIdM Model E

- Model D: Multiple MNOs and SPs across multiple networks using an IdPB with multiple IdPs and multiple SPs illustrated in Fig. 7.10.
- Model E: Multiple MNOs and SPs across multiple networks using an IdPB with multiple IdPs and SPB with multiple SPs across multiple networks illustrated in Fig. 7.11.

7.7.6 Analysis

Most of the existing IAM solutions are based on assumptions that seldom apply to multiple domains with multiple users and SPs. The entities taking part in the transactions must be trusted, a familiar protocol is used for the data exchange and appropriate credentials are used by users while requesting access to service in different domains. This is not always the case, so new IAM solutions that adopt federated methods and apply FId in a mobile network should be considered, this is due to 5G features such as network slicing, infrastructure multi-tenancy, and multiple shareholders.

MNO should implement FIdM and collaborate with other shareholders outside their network through trust relationships. The use of federated security between domains will provide efficient authentication and authorization for seamless access to services. Additionally, the authentication of the user to the access network and services in HetNets like 5G should also provide SSO. The proposed model defines procedures that provide authentication, authorization, ID protection, interoperability, and management of SSO in 5G. This will address 5G security as a unified multilayered security solution as every level of the system has different security requirements as discussed in section 2.5. Therefore, FIdM with mobile network security can become a business model for service authorization and a security hardening technique.

7.7.7 Summary

This section explored IAM and how the FIdM can be used to support NS access in 5G. It proposed a NS-FId model that facilitates federated authentication and authorization to the user for NS access using FId in 5G. It also introduced different FIdM models that can be adopted to support HetNets, SPs, and users, making a case for FIdM in 5G. The next section introduces a security protocol that uses the model discusses in section 7.6 to secure NS access by UE.

7.8 Network Service Federated Identity Protocol

This section proposes the NS-FId protocol to secure service access and achieve SSO in 5G and SP networks using FId. As explained earlier, mobile subscribers will be able to access the NS, provisioned by MNOs and SPs using a federated security mechanism to facilitate unified authentication and authorization procedures and smooth interaction between shareholders. This requires MNOs to provide robust AKA, fine-grained ACLs, privacy protection and session continuity as users roam across different networks. The author also believes that the use of FId and SSO will be an ideal solution to achieve robust authentication and authorization. Therefore, the NS-FId protocol is proposed to provide a federated multi-level authentication and authorization to the UE for secure access NS in HetNets. It leverages 3GPP AKA mechanisms presented in chapter 6, NS-FId model in section 7.6, and OAuth2 framework (Dick 2012).

Related work discussed FId in 5G (5GPPP 2017) and in HetNets (Targali et al. 2013). (Xu et al. 2018) presented a federated capability-based access control framework for IoT systems. A subscriber can use a single digital identification as FId to obtain access to SP services in different networks handled by IdP with the SP verifying the access request. And as discussed in chapter 5, ACLs have been used to facilitate authorization in mobile systems by granting a user access to a service object, checked against the user's name and some permissions. However, conventional ACL mechanisms alone are not enough to provide the required security for authorization in 5G due to its complex characteristics. The proposed solution adopts an integrated approach by using RBAC and ABAC, EBAC, and CBAC to facilitate the granting of different access rights and attributes to a user as a subject and service as an object. Moreover, EBAC provides an additional layer of security using cryptography. The CBAC approach provides the subject with a capability referring to the object and a capability token granting the subject access to the object (Sandhu & Samarati 1994) enforcing security to the entity but also to the data.

7.9 Proposed NS-FId Protocol

This section proposes a NS-FId protocol that uses the NS-FId Model presented in section 7.7. To access the services from SP, the UE would use the NS-FId protocol for authentication to SP controlled by a trusted IdP, after the authentication, the UE would be assigned FID and achieve SSO. Then the user gets authorized by SP to gain access to services through the home security domain.

7.9.1 System Model

The proposed NS-FId protocol is based on the FIdM model presented in section 7.6 as shown in Fig. 7.9 that incorporates 5G entities as specified in (3GPP 2020*f*) with federated entities which can be implemented in 5GC or third-party network. The entities UE, HSMF, IdP, SPAAA, and SS are recalled, with the same registration process described in section 7.6 used. The user's data on the federated server and UDM is used in the implementation of federated processes. Additionally, the UE's registration, authentication to the network, and authorization to SP follow the 5G standard. When the UE registers to the MNO, it is also gets registered to SP by the MNO and both MNO and SP agree with IdP on security processes and policies such as IDs, PKI, authentication methods, and ACLs. Furthermore, the content producer must register the NDO/service to the SP whereby validation, authentication, and encryption of the NDO is performed.

Table 7.3: Security Token Attributes

Attributes	**Description**
Token ID	security token identifier
Issuer	token issuer and signed its private ke
Issue time	timestamp when the token was issued
Issue sign	digital signature of the token
Subject	UE's ID to with access rights
Service	SP address/URI
Audience	entity that the token is intended for
Nonce	random nonce for authentication
Expiry time	token expiring time
Access right	set of attributes and capabilities
Scope	set of conditions (grant type, offline access, token type)

7.9.2 Authentication and Authorization

As explained in section 7.6, the first stage of authentication is provided by primary authentication method between the UE and HN, granting UE access to the HN as presented in chapter 6. Secondary authentication between UE and SP can be performed at the request of the SP for UE that is not registered to the same IdP as the SP as discussed in section 7.3. The proposed protocol provides the second stage of federated authentication handed by the IdP using some of the UE's data from primary authentication to create FId and ID token for authentication of UE to SP.

For authorization, the OAuth2 framework is adopted for authorization with some additional components. The authorization grant, access permission, attributes, and access tokens are generated based on the user's profile and access policies to assist in the authorization process. With the ACLs, the UE and the service object are assigned attributes and capabilities with encryption integrated into a FIdACap token, generated during the authorization process. The SP creates, manages, and delegates the permissions to the participating entities, while the attributes and capabilities are set up based on the agreed policies with the HN. The security tokens used in this process are also assigned attributes such as access rights, parameters, claims in relation to the subject's and object's attributes and capabilities. The attributes and format of the security token such as FIdAcap, ID, and access tokens vary but the main structure and mandatory attributes are the same as shown in Table 7.3.

7.9.3 Token and Security Policies

The PKI mechanism is used to authenticate the IdP, SP entities, and HSMF, different security policies, and cryptographic primitives are used depending on the security requirement of the process. These may include the encryption of signed hash of messages, nonce, and keys in tokens associated with nonce keys. The access and ID tokens are renewed by the UE sending a refresh token specifying the token in question, the fresh token is verified, and a new token is issued. In addition, the server can revoke any token by sending a revoke token detailing the type of authorization, token ID, content, and token type. The timestamp and expiry date in a token are used to reduce the risk of replay attacks.

Token Verification

The tokens used in the authorization process must go through a federated authorization validation process, whereby the tokens are validated by the authorization and service servers. The attributes that are checked and verified are nonce, signatures, claims, issuer, subject, audience, timestamp, and expiry date. Similarly, the grant type and token ID for renewal and revocation details are verified, as well as the data name, issuer signature, capabilities, and expiry date of the data. A timestamp is used to prevent replay attacks and check the validity of the token. It is evaluated by checking if it is within a given time window relying on the clock of the device. The message timestamp must be equal or slightly off by seconds to the current time/date, while the token lifetime time/date must be after the current time/date.

Keys Derivation

The keys used in NS-FId protocol use the KDF(Nonce, ID) as an input in the key derivation of K_{UE3A} and K_{UESS} using the k(x) parameters. After a successful authentication between the UE and the SP, key K_{UE3A} is derived. Key K_{UESS} is a session key generated by SP for communication between UE and SS.

7.10 Modelling of NS-FId Protocol

This section models the proposed NS-FId protocol using UE, SMF, IdP, SPAAA, and SS as entities. The cryptographic primitives used are symmetric and asymmetric encryption, one-way hash function, digital signature, and MAC, facilitated by ECIES, and security properties assumptions are based on the 5G specifications in (3GPP 2020*f*). The proposed solution should provide a federated authentication and authorization with SSO to UE and it stops the requirement of the secondary authentication procedure every time the UE requests access to SP services. This protocol addresses the concern of sharing the UE's SUPI and other security contexts outside the HN. The UE would be able to access services of different networks even without roaming agreements so long as they fulfil the federated security acquirements. Providing privacy preservation for users' IDs is intended to enforce accountability and non-repudiation in 5G, hence should be included in the modelling. The process includes initiation, authorization grant, identification, authentication, authorization, and granting access.

7.10.1 Protocol Message Exchange and Execution

This subsection gives a concise description of NS-FId protocol execution and message exchange between UE, SPAAA IdP, and SS. The UE would have achieved registration and primary authentication to HN and secondary authentication to the SP if needed. As a result, it is assumed before the start of the protocol run that the UE is in possession of $GPSI$, however, it would only have EID and K_{UE3A} if the secondary authentication method in section 7.3 was performed. The protocol message exchange consists of three phases: (i) service request, (ii) identification and authentication, (iii) authorization, illustrated in Fig. 7.12, with reference to notation in Table 7.4 are described as follows:

Phase1: Service Request

`Msg1.UE`$\rightarrow$ `SMF:({ServReq})`

After a successful primary authentication to HN, the UE initiates a service session by sending service request message $ServReq$ to the HN SMF via AMF, which includes service name $ServName$ and session ID (SID).

`Msg2. SMF` $\rightarrow$ `UE:({RedSp})`

When SMF receives message 1, it checks the user's subscription data, security status, and context with UDM. Check if the SP is internal or external and if it is internal, then SMF retrieves the UE generic ID ($GPSI$) that corresponds with the UE permanent ID ($SUPI$) and pass it on to internal SP related functions. Otherwise, it sends the $GPSI$ to the UE along with the SP ($SPID$), SP public key PK_{SP} in $RedSP$ message for the UE to request access to external SP services. The SMF redirects the UE to SP for authorization in both cases.

Phase 2: Identification and Authentication

`Msg3.UE` $\rightarrow$ `SPAAA:({AuthzReq},{PKSP})})`

The UE sends authorization request $AuthReq$ to SPAAA encrypted with SP public key PK_{SP}, it includes the SID, service name $ServName$, and its $GPSI$.

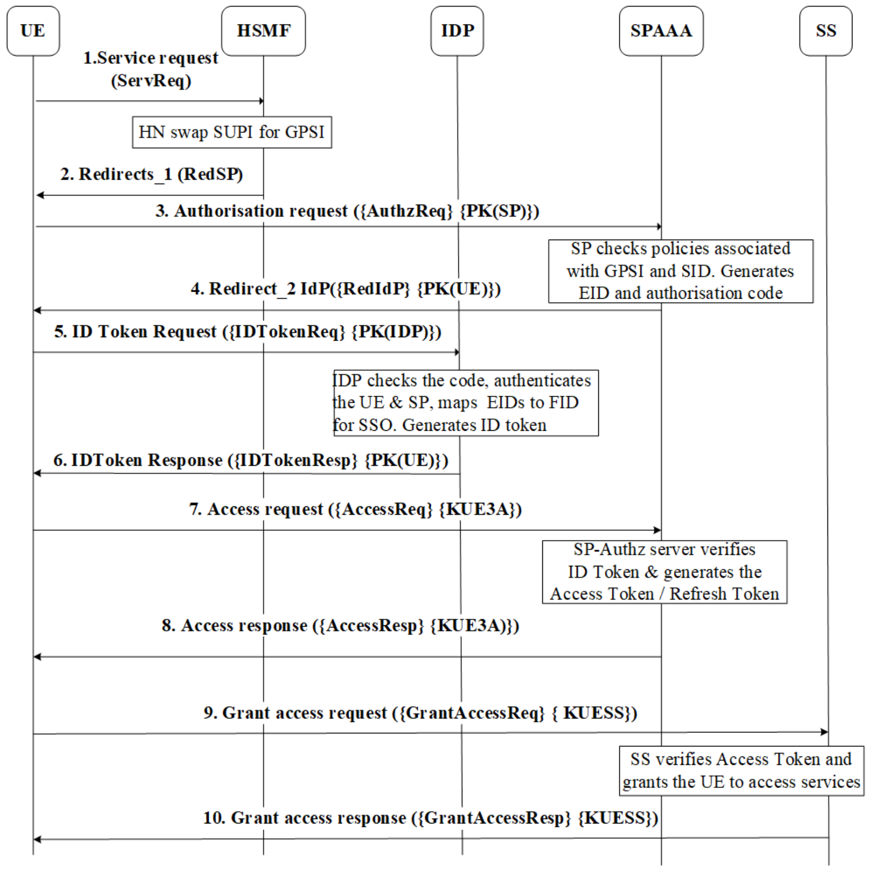

UE
HSMF
IDP
SPAAA
SS
1.Service request (ServReq)
HN swap SUPI for GPSI
2. Redirects_1 (RedSP)
3. Authorisation request ({AuthzReq} {PK(SP)})
SP checks policies associated with GPSI and SID. Generates EID and authorisation code
4. Redirect_2 IdP({RedIdP} {PK(UE)})
5. ID Token Request ({IDTokenReq} {PK(IDP)})
IDP checks the code, authenticates the UE & SP, maps EIDs to FID for SSO. Generates ID token
6. IDToken Response ({IDTokenResp} {PK(UE)})
7. Access request ({AccessReq} {KUE3A})
SP-Authz server verifies ID Token & generates the Access Token / Refresh Token
8. Access response ({AccessResp} {KUE3A)})
9. Grant access request ({GrantAccessReq} { KUESS})
SS verifies Access Token and grants the UE to access services
10. Grant access response ({GrantAccessResp} {KUESS})

Figure 7.12: NS-FId Protocol Message Exchange Flow

Table 7.4: NS-FId Notation and Description

Notation	**Description**
IdPID	IdP identifier
SPID	SP identifier
SID	service identifier
K(X)	session key
R1	random nonce challenge
EID	UE external identifier
FID	UE federated identifier
GPSI	UE generic identifier
IDT	ID token
PK(x)	public key
SK(x)	private key
label	capabilities string
Act	access Token
Ack_1	acknowledgement
Serv	service bitstring
Exp	expiry date
Ts	time stamp
AGrcode	authorization grant code
h(x)	hash value of message x
{x}{k}	message encrypted with key K

`Msg4. SPAAA` $\rightarrow$ `UE:({RedIdP},{PKUE})`
When the SP receives a request in message 3, it retrieves the $GPSI$, *ServName* and SID that includes the HN details and checks the UE's HN service agreement and policies with the SP. If it is valid then it generates authorization grant code *AuthGrant*, UE's (EID), and the session key K_{UE3A} for the UE and SPAAA, sends it to the UE in *RedIdP* messages encrypted with UE's public key that was part of the security context shared between the HN and SP in the service provisioning agreement. The EID and K_{UE3A} are only generated if the secondary authentication method was not performed otherwise they are reaffirmed to the UE. Then SPAAA redirects UE to IdP for a federated authentication procedure.

`Msg5. UE`$\rightarrow$ `IdP:({IDTokenReq},{PKIdP})`
When the UE receives message 4 it retrieves the K_{UE3A}, then sends EID, authorization grant code *AuthzGrant* and a nonce $R1$ to IdP for Federated Identifier FID and ID token IDT encrypted with the IdP public key PK_{IdP}.

`Msg6. IdP`$\rightarrow$ `UE:({IDTokenResp},{PKUE})`
When the IdP receives message 5, it checks the grant code in *AuthGrant*, retrieves the EID and PK_{UE}, whether there is a need of secondary authentication or the security context. The IdP verifies the UE credentials and generates the FID for the UE and IDT as per the security policy. It maps FID with EID, UE profile with MNO/SP attributes to support SSO then sends IDToken response message *IDTokenResp* which includes FID, IDT, hash of the IDT $hash(IDT)$, nonce $R1$, the hash of the whole message $hash(FID, IDT, hash(IDT), R1)$ and encrypts the whole message with UE's public key $PKUE$. Both hashes are signed with IdP private key SK_{IdP}.

Phase 3: Authorization
`Msg7. UE` $\rightarrow$ `SPAAA:({AccessReq},{KUE3A}})`
When the UE receives message 6, it retrieves the FID, send access request message $AccessReq$ for access token AcT/refresh tokens from the authorization server of the SP. The message includes the IDT and hash of the IDT $hash(IDT)$ signed with IdP private key SK_{IdP} and encrypts the whole message with K_{UE3A}.

`Msg8. SPAAA` $\rightarrow$ `UE:({AccessResp},{KUE3A})`
When the SPAAA receives, message 7, the authorization server verifies the IDT, checks IDT is valid with the right parameters, if it does then it generates an access token AcT/fresh token. Then sends an access response message $AccessResp$, which includes the AcT/refresh token, session key K_{UESS} for UE and SS, hash of AcT $hash(AcT)$ and hash of whole message $hash(AcT, hash(AcT), K_{UESS})$ signed with SP private key $SKAAA$ and encrypted with the shared key K_{UE3A}.

`Msg9. UE` $\rightarrow$ `SS:({GrantAccessReq},{KUESS})`
The UE sends grant access request message $GrantAccessReq$ that includes AcT and hash of AcT $hash(AcT)$ received in message 8 to SS encrypted with K_{UESS}.

`Msg10. SS`$\rightarrow$ `UE:({GrantAccessResp},{KUESS})`
The SS verifies the AcT and then sends a grant access response $GrantAccessResp$ granting UE access to the service $SERV$ (DATA) signed hash of the service name $ServName$ encrypted with K_{UESS}.

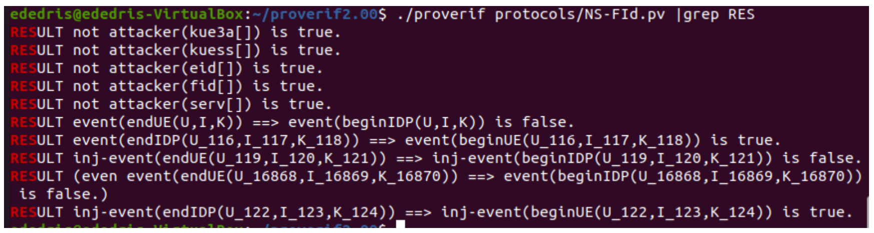
7.11 Formal Verification of NS-FId Protocol

This section follows the same ProVerif modelling process as defined in chapter 3. The protocol modelling is like that illustrated in section 7.3 with some changes and full ProVerif protocol syntax in Appendix D.1.

7.11.1 Formal Analysis of NS-FId Protocol

The protocol was simulated using secure, insecure channels and processes `procUE` as UE, `procSMF` as SMF, `procIdP` as IdP, `procSPAAA` as SPAAA, and `procSS`as SS.

When the protocol was modelled, an attack was found on the public channel between UE, IdP and SP, hence the protocol is insecure. The security properties we are interested in are the secrecy of EID, FID, K_{UE3A}, $PK_X(SK_X)$, IDT, AcT, and authentication of UE to SP via IdP. ProVerif



```
ededris@ededris-VirtualBox:~/proverif2.00$ ./proverif protocols/NS-FId.pv |grep RES
RESULT not attacker(kue3a[]) is true.
RESULT not attacker(kuess[]) is true.
RESULT not attacker(eid[]) is true.
RESULT not attacker(fid[]) is true.
RESULT not attacker(serv[]) is true.
RESULT event(endUE(U,I,K)) ==> event(beginIDP(U,I,K)) is false.
RESULT event(endIDP(U_116,I_117,K_118)) ==> event(beginUE(U_116,I_117,K_118)) is true.
RESULT inj-event(endUE(U_119,I_120,K_121)) ==> inj-event(beginIDP(U_119,I_120,K_121)) is false.
RESULT (even event(endUE(U_16868,I_16869,K_16870)) ==> event(beginIDP(U_16868,I_16869,K_16870))
 is false.)
RESULT inj-event(endIDP(U_122,I_123,K_124)) ==> inj-event(beginUE(U_122,I_123,K_124)) is true.
```

Figure 7.13: NS-FId Attack ProVerif Results

results in Fig. 7.13 shows that secrecy `(kue3a[])`, `(kuess[])`, `(eid[])`, `(fid[])`, `(serv[])` and authentication UE to IdP holds but IdP to UE fails.

The `event endIdP` indicates that the protocol has been completed by IdP, the UE received message 6 and sent message 7, while `event beginIdP` indicates that message 6 was sent by IdP. The parameters of the protocol *AuthzGrant*, $R1$ and EID are taken as an argument by these events, with the IdP verifying the grant codes and responding to nonce with an ID token. The ID token is only sent if the arguments are true, otherwise, an authentication failure for re-authentication initiation is sent. This study would like to prove the following correspondence.

```
(*Check authentication between UE and IdP*)
query U:host,I:host,K:pkey;
event(endUE(U,I,K))==> event(beginIdP(U,I,K)).
inj-event(endUE(U,I,K)) ==> inj-event(beginIdP(U,I,K)).
```

The sending of message 5 before message 6 proves the correspondence directly, hence, it holds in ProVerif. This research also tries proving and concluding the desired correspondence by noticing that event which has IDT as an argument can only be executed if *AuthzGrant*, $R1$ and EID are sent first. That is, the request of the IDT is executed with IDToken Response that includes the FID. This gives true results in ProVerif, hence holding.

7.11.2 The Attack Against NS-FId Protocol

ProVerif results indicate there was an attack on the protocol as shown in Fig. 7.13. The attack derivation and trace are explained in this subsection.

Attack Derivation and Trace

The attacker I starts by eavesdropping on the communication between entities, impersonates the IdP, continuing the protocol with UE, which completes the protocol with the attacker instead of

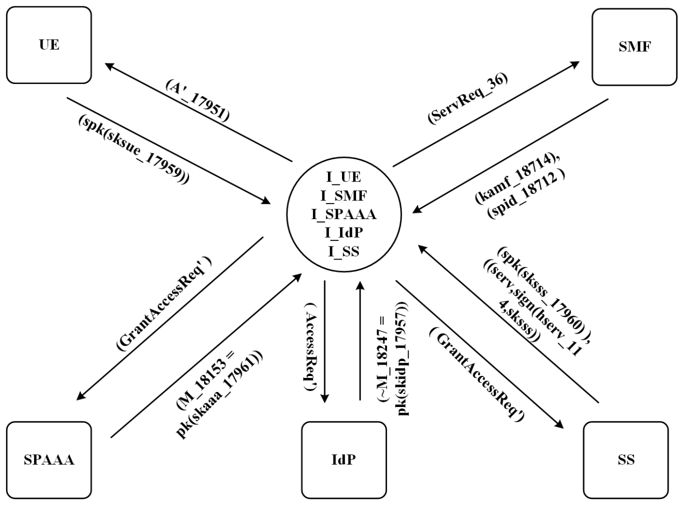


Figure 7.14: NS-FId Attack Trace

IdP. The attacker's actions are illustrated in Fig. 7.14 and explained concisely in derivation and trace steps below, some text is omitted for simplicity, full trace output in Appendix J:

- Steps 1 and 2 indicate that attacker has some term `A'_17951`, `attacker(A'_17951)` input {25}, so `event endUE(hostU[],hostI[],pk(skidp[]))` may be executed at {34}. The input {1}-{8} corresponds with the creation of public/private keys. Output {11} corresponds with the creation of public key that the attacker stores in a fresh variable `~M` for later use. Input {25} corresponds to a session `(copy a_17956)` the UE has with the attacker, where the attacker gets UE ID.

- In trace 1, attacker's goal is achieved when `event endUE(hostU,hostI,pk(skidp_17957))` is executed in session `copy a_17956` the UE has with the attacker at event {34}. In trace 2, the injective agreement fails when `endUE(hostU,hostI,pk(skidp_22772))` is executed in session `copy a_22770` that is the UE end the protocol thinking it was talking to IdP while UE never run the protocol with IdP.

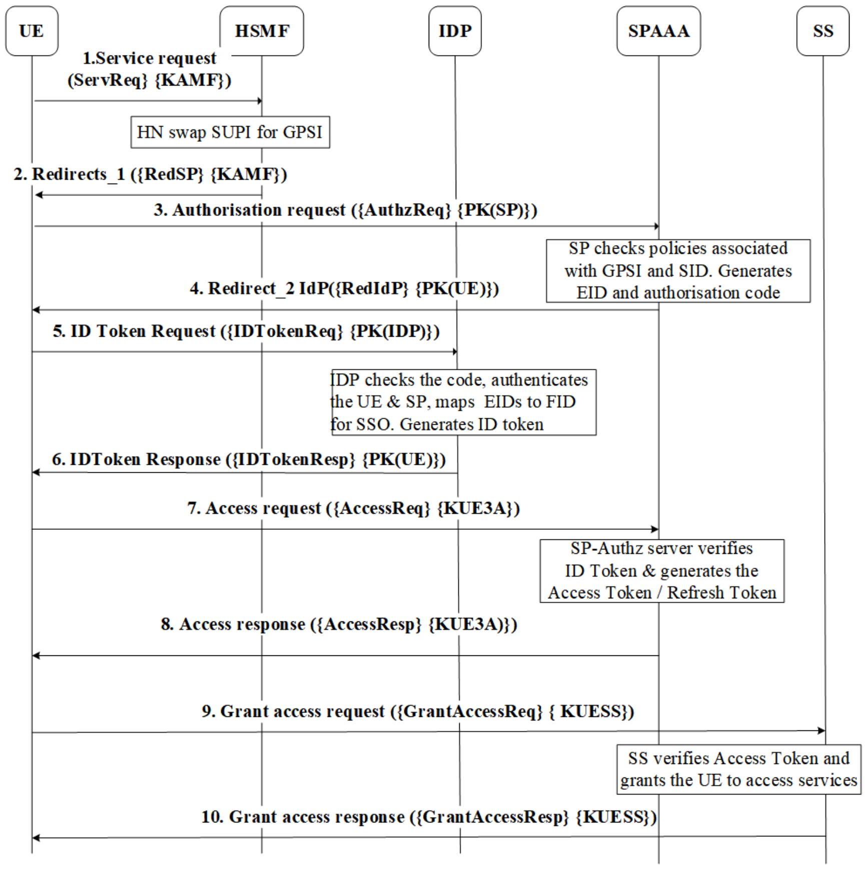

UE
HSMF
IDP
SPAAA
SS
1.Service request (ServReq} {KAMF})
HN swap SUPI for GPSI
2. Redirects_1 ({RedSP} {KAMF})
3. Authorisation request ({AuthzReq} {PK(SP)})
SP checks policies associated with GPSI and SID. Generates EID and authorisation code
4. Redirect_2 IdP({RedIdP} {PK(UE)})
5. ID Token Request ({IDTokenReq} {PK(IDP)})
IDP checks the code, authenticates the UE & SP, maps EIDs to FID for SSO. Generates ID token
6. IDToken Response ({IDTokenResp} {PK(UE)})
7. Access request ({AccessReq} {KUE3A})
SP-Authz server verifies ID Token & generates the Access Token / Refresh Token
8. Access response ({AccessResp} {KUE3A)})
9. Grant access request ({GrantAccessReq} { KUESS})
SS verifies Access Token and grants the UE to access services
10. Grant access response ({GrantAccessResp} {KUESS})

Figure 7.15: Improved NS-FId Protocol Message Exchange Flow

7.11.3 Improved Version of NS-FId Protocol

When the protocol was modelled again, no attack was found due to the changes made in the specification of the new version of the NS-FId protocol. To improve the protocol instead of assuming that the exchange between UE and SMF in the HN is secure, key K_{AMF} is used in their exchange, which is derived during primary authentication discussed in chapter 6. The improved version message exchange is shown in Fig. 7.15 and ProVerif results in Fig. 7.16, shows that no attacker on secrecy is true and authentication holds for both non-injective and injective agreements.

7.11.4 Security Analysis

Protocol Security Analysis

This security analysis for the improved version of NS-FId protocol is based on security requirements of sets 1 and 2 as presented in chapter 3.
Analysis based on security properties of set 1 is as follows:

- Secrecy: Since the EID, FID and K_{UE3A} are not known to the attacker after the run of the protocol, it achieves this requirement. Hence achieving this property would also cover protocol data confidentiality and privacy with the *Secrecy* property in ProVerif.

- Aliveness: When UE sends an authorization request to SP including $SPID$, both the SP and HN get non-injective agreement on FID with the IdP, then aliveness of UE is obtained by the SP conversely.

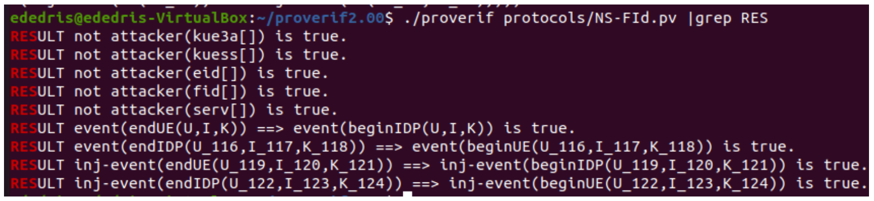
- Weak Agreement: When HN gets non-injective agreement on FID with IdP, it addresses this requirement. In addition, HN and SP achieve a weak agreement after the confirmation of the session key.

- Non-injective Agreement: The UE gets non-injective agreement on FID with the IdP, while the SP gets it on $GPSI$ with HN. The non-injective agreement on EID is obtained by HN with the SP after FID is generated by IdP. Moreover, HN gets non-injective agreement on IDT and AcT with IdP and SP, respectively, as they both include FID, fundamental to the protocol's objective. Since the IDT includes a nonce, HN gets an assurance from the token IDT given to the UE by the IdP.

- Injective Agreement: The UE is assured by the SP that IdP can be trusted with injective agreement on EID. The UE obtain injective agreement on IDT and *AcT* with IdP and SP, respectively, assuring the UE that its sessions with SP were authorized by the HN. Also, SS is assured that its sessions with the UE were authorized by SP.

Analysis based on security properties of set 2 is as follows:

- Mutual Entity Authentication: This is achieved when FID is generated, the UE is authenticated to the IdP and the access token request is sent to SP. Since the IDT is computed using grant code from SP, this enforces weak agreement between SP and UE after a successful authentication and generation of IDT. In addition, when UE sends $GPSI$ via SMF to SP, it is used to verify UE and the computation of grant codes, FID, and IDT, enforces this requirement.



```
ededris@ededris-VirtualBox:~/proverif2.00$ ./proverif protocols/NS-FId.pv |grep RES
RESULT not attacker(kue3a[]) is true.
RESULT not attacker(kuess[]) is true.
RESULT not attacker(eid[]) is true.
RESULT not attacker(fid[]) is true.
RESULT not attacker(serv[]) is true.
RESULT event(endUE(U,I,K)) ==> event(beginIDP(U,I,K)) is true.
RESULT event(endIDP(U_116,I_117,K_118)) ==> event(beginUE(U_116,I_117,K_118)) is true.
RESULT inj-event(endUE(U_119,I_120,K_121)) ==> inj-event(beginIDP(U_119,I_120,K_121)) is true.
RESULT inj-event(endIDP(U_122,I_123,K_124)) ==> inj-event(beginUE(U_122,I_123,K_124)) is true.
```

Figure 7.16: NS-FId Safe ProVerif Results

- Mutual Key Authentication: The session keys K_{UE3A} and K_{UESS} are authenticated after the successful authentication of the UE to the SP and generation of IDT. This is also enforced as the UE and SP credentials are used in the generation of the session keys.
- Mutual Key Confirmation: The successful authentication of UE by IdP and security context agreement between the HN, SP, and IdP that facilitates the generation of grant codes and security tokens enforces this requirement.
- Key Freshness: There is no function in ProVerif to check key freshness, however, the SP checks if $GPSI$ is valid and freshness of K_{UE3A} and while the IdP checks the validity of the grant codes. Then SP checks whether IDT includes a nonce and time stamps hence checking the freshness of K_{UESS}. It also checks if session keys are not from the previous session, with K_{UE3A} linked to a SID, while K_{UESS} is linked to $AuthzGrant$ and IDT. Since the secrecy of these keys is not violated, hence the keys are fresh.
- Unknown-Key Share: Since the reachability property in ProVerif checks the entities' aliveness with their IDs and key binding, this attack is prevented. The protocol achieves this requirement by including $GPSI$, EID, and $SPID$ in the authentication message exchange and session keys derivation. Also, the K_{UE3A} is only sent to UE after validation of $GPSI$, $SPID$ is encrypted with public key PK_{SP}, while K_{UESS} is only sent to UE after the verification of IDT. Moreover, the public keys used are based on ECIES as per the 5G standard.
- Key Compromise Impersonation Resilience: This requirement is achieved since the K_{UE3A} is derived after the SP verifies $GPSI$ and SID, while K_{UESS} is derived after IDT verification. As stated earlier that knowing a key in one session is not enough to deduce a key in another session. So no entity or adversary is capable of computing keys in past sessions or can predict future keys to compromise any keys, the ECIES together with IdP, and SPAAA will have to be compromised at the same time. It should also be noted that if the HN primary authentication protocol is compromised, it does not mean that NS-FId or another protocol in this study is compromised.

Security Consideration

The tokens used in this protocol can be cached for re-use and renewed if they are expired based on the security policies, type of service, security parameters validity, suspicious requests, or faulty process. To facilitate the requirements of the 5G standard, $GPSI$ and EID are used to prevent the exposure SUPI outside the HN. The UE's $GPSI$ is translated to a corresponding $SUPI$ by SMF the HN, to EID by SP and FID by IdP in DN. Hence, a universal recognition of the UE, multiple IDs mapped together into a federated single ID, matching its user profile in the HN and DN.

With 5G is introducing network slices, infrastructure multi-tenancy and restriction of sharing security context with a third-party, secure communication can be facilitated by federated protocols. In terms of network and service access, successful end user authentication should provide SSO in 5G. NS-FId protocol provides mutual authentication, authorization, ID protection, secure access, interoperability, and SSO. The implementation of the NS-FId protocol secures the NS-Fid Model and enables massive communication, seamless connectivity, and secure access to services taking advantage of the single digital ID of the UE.

7.11.5 Summary

This subsection explored how FId can be used to provide the UE with a universal ID in 5G. It proposed a federated protocol NS-FId that is based on the NS-FId model in section 7.6 that can secure the service access in 5G and DN as well as providing the UE with SSO. The protocol was verified with ProVerif and proved to be secured. This NS-FId protocol could be applied by users to access services and by MNOs to share infrastructure resources in HetNets. In the next section, a protocol that addresses the caching and sharing problem in 5G is introduced.

7.12 Data Caching and Data Sharing Security in 5G Network

This section introduces data caching and data sharing security for authorizing the UE to cache and share the data accessed in section 7.8. It proposes a DCSS protocol to address data caching and sharing security. After being granted access to the service, the UE can access data, request further authorization, and performing other activities such as data caching, data sharing, and delegating its access rights to other UEs. This could be achieved with the assistance of the SP or independently depending on the policies and network coverage conditions. The ACL mechanisms such as ABAC and CBAC with security context from the NS-FId protocol are used to enable the UE to request caching and sharing permissions from SP.

7.12.1 Capabilities and Attributes

The capabilities mechanism used in this section is based on the abilities defined in (Dennis & Horn 1983) and labels defined in (Aiash & Loo 2015) as token claims. After being granted access to the services and the UE retrieves the data and then sends another request to SPAAA for caching and sharing data. The SPAAA generates security tokens that define the object and subject abilities, the data is the object and UE is the subject. The generated tokens are cache and share tokens, consisting of capabilities and attributes parameters, similar in structure but with different parameters. They are linked with the data object name and the UE ID, which are used to define objects and subjects' abilities. As in (Aiash & Loo 2015), dot-separated sequence of numbers is used for an ability in the form of a label whereby an ability is represented by a string $.i_1.i_2.i_3...i_n$ for value n and $i_1, i_2, i_3..., i_n$ are integers. In this case, the UE or NDO abilities are .1.2.3.4, or 01.02.03.04.

After tokens are created by the SPAAA's authentication server, they are passed to the authorization and service servers. After authentication, both data and UE will be given labels as part of their security tokens. Access to a data object is granted if the data object's label is a prefix of the UE's label. Whereby, a data object with a label "2.0.0" with abilities "0.1.0" for caching and

"0.0.2" for sharing could only be accessed by UE with abilities like ".2.0.0", ".2.1.0", ".2.1.2"...etc. That is, whenever an authenticated UE requests content data, it must present the right label to SS that confirms its access rights to cache or share the data. The SPAAA on behalf of the SP generate labels for the subject and object, the UE cannot promote themselves to access other data objects, but the permission can be delegated to other UEs if the delegation permission was part of the initial request of the access token in section 7.8. The labels are integrated into the security tokens as access rights. Any other subject to access the data will require the nonce key to decrypt that data object as the data can only be accessed with the access token or nonce key both generated by SP.

7.13 Proposed DCSS Protocol

The proposed DCSS protocol is divided into caching and sharing authorization stages that use cache and share token to authorize users. These tokens have a similar structure as that of the access token, however, they have some additional claims and scope to achieve their objectives. These include labels and access rights, delegation, i.e., UE and data object capabilities, delegation ability in form of label bitstring, and the security context such as nonce key. The labels enable caching and sharing of the data in situations where the SP involvement is limited or non-existent, the UE is out-of-coverage, and when transferring attributes and capabilities from one UE to another.

7.13.1 Data Cache and Share Authorization

The UE uses the access token and security context from NS-FId protocol such as K_{UE3A} and K_{UESS} to request caching authorization of the accessed data. The data caching stage provides authorization to UE to cache the accessed data. After caching the data, the UE uses the cache token and security context such as acknowledgement strings to request sharing authorization of the accessed or cached data. The data sharing stage provides authorization to the UE to share the accessed data. The UE in HN or VN might want to share the cached data with another UE, in order to do so, the UE must request permission from the SP to grant it sharing permission. The cache/share tokens have the same security attributes as the access token discussed in section 7.8 such as a timestamp, expiry date, entity's ID, but they have an additional attribute in form of a capability label. Similarly, the security token and verifiable data are distributed securely to the entities and digitally signed by the SPAAA to guarantee the token's integrity and authenticity.

7.14 Modelling of DCSS Protocol

This section models the proposed DCSS protocol using `UE, SPAAA and SS` entities based on the model in Fig. 7.9 in section 7.6. It also uses similar cryptographic primitives used for NS-FId protocol in section 7.8 with the labels, cache, and share tokens as the new additions. After the successful run of the protocol, it should achieve service authorization by allowing the UE to cache and share the data accessed from SS. It also achieves entity and message authentication. This protocol is an optional protocol that might be used if the UE needs to cache or share data. Moreover, it also depends on data being configured with abilities to be cached or shared and UE being given abilities to cache or share the data. In a case where neither ability of the subject or the object match then the caching/sharing authorization will be denied. This is achieved using capability labels and

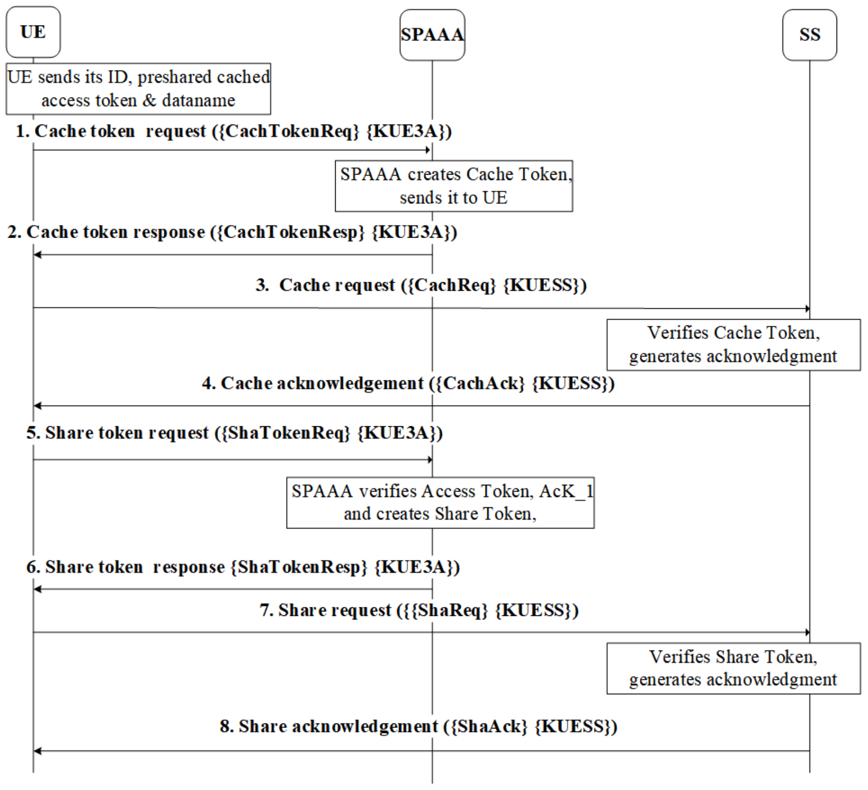

UE
SPAAA
SS
UE sends its ID, preshared cached access token & dataname
1. Cache token request ({CachTokenReq} {KUE3A})
SPAAA creates Cache Token, sends it to UE
2. Cache token response ({CachTokenResp} {KUE3A})
3. Cache request ({CachReq} {KUESS})
Verifies Cache Token, generates acknowledgment
4. Cache acknowledgement ({CachAck} {KUESS})
5. Share token request ({ShaTokenReq} {KUE3A})
SPAAA verifies Access Token, AcK_1 and creates Share Token,
6. Share token response {ShaTokenResp} {KUE3A})
7. Share request ({{ShaReq} {KUESS})
Verifies Share Token, generates acknowledgment
8. Share acknowledgement ({ShaAck} {KUESS})

Figure 7.17: DCSS Protocol Message Exchange Flow

delegation of access permissions. The messages between entities are encrypted and signed by the sender while the tokens are signed by the SPAAA.

7.14.1 Protocol Message Exchange and Execution

This subsection explains the DCSS protocol execution and message exchange, it consists of data caching and sharing authorization phases. The UE is granted access to the SP services using federated authorization protocol presented in section 7.8, that is when the access token AcT, K_{UE3A} and K_{UESS} are generated before this proposed protocol is run. In addition, the UE and SS would also know each other's ID, public key, and shared session key from the previous authentication and authorization protocol that granted the UE access to the service. The protocol messages exchange is illustrated in Fig. 7.17, with reference to notation in Table 7.5 and described as follows:

Caching Authorization

`Msg1.UE→SPAAA: ({CachTokenReq},{KUE3A})`

After the UE is granted access to the data, it sends a cache request $CachTokenReq$ to SPAAA for a cache token which includes the UE ID (EID/FID), data name $DataName$, AcT, the hash of the access token $hAct$ signed by the SPAAA private key SK_{AAA} and key K_{UESS} encrypted with preshared key K_{UE3A}. The Act, $hAcT$, SK_{AAA}, K_{UE3A} and K_{UESS} are generated during the

Table 7.5: DCSS Protocol Notation and Description

Notation	Description
SPID	SP identifier
ASID	authentication server ID
SSID	service server ID
DNN	service code:SPID
R1	nonce
EID	UE permanent identifier
KUE3A	KDF(nonce, eid,spid)
KUESS	KDF(nonce,eid,ss)
Ack_1	acknowledgement 1
Hack_1	hash for Ack 1
Ack_2	acknowledgement 2
Hack_2	Hash for Ack_2
Exp	expiry date
D1	dataname
Ts	timestamp
label	capabilities
Kd	nonce key
AcT	access token
ChT	cache token
ShT	share token
h(x)	hash value of message x
{x}{k}	message encrypted with key K

security procedure between UE and SPAAA to grant the UE access to the service. Key K_{UESS} is included if it was not sent in the previous authentication and authorization procedure and the *AcT* is used as an additional ID as it has all the UE subscription details.

`Msg2.SPAAA→UE: ({CachTokenResp},{KUE3A})`

When the SPAAA receives message 1, the authorization server verifies the *AcT*, if the subscription policy includes caching rights, it creates and sends a cache token *ChT*, with its hash *hChT* signed with its private key SK_{AAA} in cache response *CachResp* message to the UE encrypted with key K_{UE3A}.

`Msg3.UE→SS: ({CachReq},{KUESS})`

The UE receives a cache token in message 2, sends a cache request message *CachReq* encrypted with key K_{UESS} to the service server requesting permission to cache the data, the message includes a cache token *ChT* and a signed hash of the cache token *hChT* with SPAAA private key SK_{AAA}.

`Msg4.SS→UE:({CachAck},{KUESS})`

The SS receives *CachReq* in message 3, verifies the cache token *ChT*, if it is valid then it authorizes the UE to cache the data by sending an acknowledgement Ack_1 in *CachAck* message encrypted with K_{UESS}, which includes the hash of the Acknowledgement $hAcK_1$ signed with the SS private key SK_{SS}.

Sharing Authorization

`Msg5:UE→SPAAA:({ShaTokenReq},{KUE3A})`

Now that the UE is authorized to cache the data, it sends a share request message $ShaTokenReq$ to SPAAA for a share token to get share authorization of the cached data. The message includes AcT and cache acknowledgement Ack_1, for validation and to notify the authorization server that it was granted permission to cache the data it is requesting to share by the SS.

`Msg6:SPAAA→UE:({ShaTokenResp},{KUE3A}})`

When SPAAA receives message 5, it verifies the AcT, checks if the subscription policy includes sharing rights then sends a share response message $ShaTokenResp$ to the UE in message 6. The message includes a share token ShT, hash of the share token $hShT$ signed with SK_{AAA}, encrypted with the K_{UE3A} that authorizes the UE to share the content.

`Msg7:UE→SS: ({ShaReq},{KUESS})`

When the UE receives message 6, it sends a share request message $ShaReq$ to SS encrypted with K_{UESS}, that includes a share token ShT and its hash $hShT$ that were sent to UE in message 6 by SPAAA.

`Msg8:SS→UE:({ShaAck},{KUESS})`

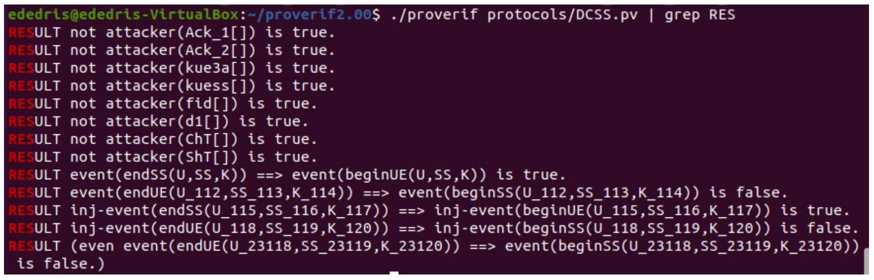
The SS verifies the received share token parameters in message 7 and sends an acknowledgement Ack_2 with its hash $hAck_2$ signed with SS's private key SK_{SS} in a share acknowledgement message $ShaAck$ back to UE authorizing it to share the data.

7.15 Formal Verification of DCSS Protocol

This section follows the same ProVerif modelling process as defined in chapter 3. The protocol modelling used is like that illustrated in section 7.8 with some changes, full ProVerif protocol syntax in Appendix E.1.

7.15.1 Formal Analysis of DCSS Protocol

The DCSS protocol was simulated using secure, insecure channels and processes `procUE(hostU)` as UE, `procAAA(hostA)` as SPAAA, and `procSS (hostSS)` as SS. When the protocol was modelled, there was an attack found on the public channel between UE, SPAAA and SS as shown in Fig. 7.18. The security properties the study was interested in are authentication, authorization, privacy of communication data specifically the tokens (ChT/ShT) and acknowledgements (Ack_1/Ack_2). Using formal analysis, in consideration with adversary vector, there were attacks on the protocol. The above ProVerif results indicate that the secrecy of `ChT, ShT, D1, FID, Ack_1, Ack_2` holds. The authentication event between UE and SS does not hold, in the form of non-injective and



```
ededris@ededris-VirtualBox:~/proverif2.00$ ./proverif protocols/DCSS.pv | grep RES
RESULT not attacker(Ack_1[]) is true.
RESULT not attacker(Ack_2[]) is true.
RESULT not attacker(kue3a[]) is true.
RESULT not attacker(kuess[]) is true.
RESULT not attacker(fid[]) is true.
RESULT not attacker(d1[]) is true.
RESULT not attacker(ChT[]) is true.
RESULT not attacker(ShT[]) is true.
RESULT event(endSS(U,SS,K)) ==> event(beginUE(U,SS,K)) is true.
RESULT event(endUE(U_112,SS_113,K_114)) ==> event(beginSS(U_112,SS_113,K_114)) is false.
RESULT inj-event(endSS(U_115,SS_116,K_117)) ==> inj-event(beginUE(U_115,SS_116,K_117)) is true.
RESULT inj-event(endUE(U_118,SS_119,K_120)) ==> inj-event(beginSS(U_118,SS_119,K_120)) is false.
RESULT (even event(endUE(U_23118,SS_23119,K_23120)) ==> event(beginSS(U_23118,SS_23119,K_23120))
 is false.)
```

Figure 7.18: DCSS Attack ProVerif Results

injective agreements that is SS may end the protocol thinking it is talking to UE while UE never run the protocol with SS.

The `event beginSS` means that the SS has completed the protocol, that the UE received message 8 and sent message 7, `event beginUE` means that the SS received message 3. These events take as arguments all parameters of the protocol: K_{UESS} and ChT and its claims, SS which must verify the ChT and digital signature. If the arguments are true, then Acknowledgement is sent otherwise it sends `invalid token` message. This research tries to prove the following correspondence between UE, SPAAA, and SS events.

```
(*Check authentication between UE and SS*)
query U: host, SS: host, K: key;
event(endSS(U, SS, K)) ==> event(beginUE(U,SS,K)).
inj-event(endSS(U, SS, K)) ==> inj-event(beginUE(U, SS, K)).
```

The direct proof of this correspondence in ProVerif does not hold because message 7 and message 8 sent after `query []; event (endSS()) event(beginUE())` fails to hold due attacker's knowledge of M, `query attacker (M) ==> event ()` as explained in attacker trace below. We also try to prove the desired correspondence by concluding that event which has K_{UESS} as an argument cannot be executed before *Ack_2* has been sent. That is before the *CacheReq* message has been executed with *CachAck* to generate the *Ack_2*. One part of correspondence holds with true while the other does not hold in ProVerif.

7.15.2 The Attack Against DCSS Protocol

After the simulation ProVerif produced results indicating there was an attack on the protocol as shown in Fig. 7.18. The attack derivation and trace are explained in this subsection.

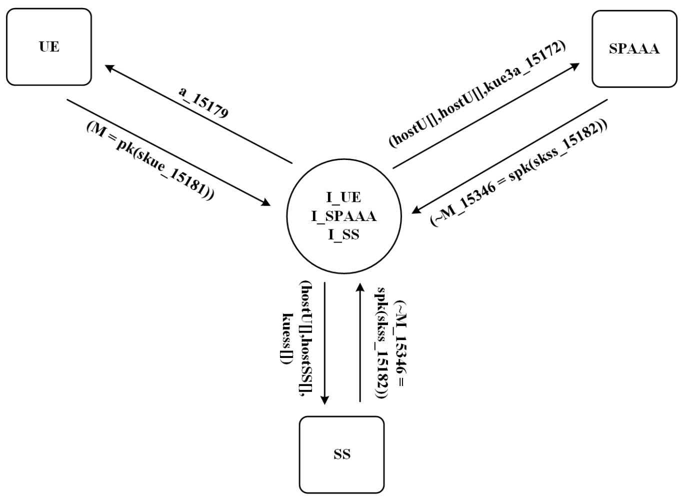


Figure 7.19: DCSS Attack Trace

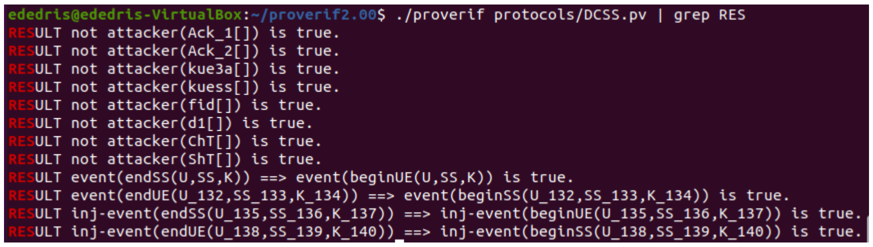
Attack Derivation and Trace

The attacker I starts by eavesdropping on the communication between entities, impersonates the SS, continuing the protocol with UE, which completes the protocol with the attacker instead of SS. The attacker's actions are illustrated in Fig. 7.19 and explained concisely in derivation and trace steps below, some text is omitted for simplicity, full trace output in Appendix J:

- Steps 1-3 indicate that the attacker may know the UE, therefore, may also know key `kue3a` by using the 3-tuple function. The attacker may have received message A'_15173 by eavesdropping on the public channel at input {12}, uses the knowledge to obtain $kue3a\_19424$. Output {6}-{8} corresponds with the creation of public keys and their insertion on the public channel, which the attacker saves in the new variables `~M = pk(skue_15181), ~M_15346 = spk(skss_15182)` and `~M_15427 = spk(skaaa_15183)`for reuse later. The attacker gets key `kuess` at input {56} in session `copy a_15180`.
- Trace 1 shows there was an attack against the `event endUE(hostU,hostSS,kuess)` at event {25} where the attacker achieves his goal in session `copy a_15180 (goal)` with UE, when the `event endUE(hostU,hostSS,kuess)` is executed. In trace 2, the attacker was able to achieve another goal against one-to-one relationship correspondence in session `copy a_19431`, that is the UE end the protocol thinking it was talking to SS while UE never run the protocol with the SS.

7.15.3 Improved Version of DCSS protocol

This subsection discusses the changes made in the modelling of the protocol to address the security threats discovered in the previous version and presents an improved version of the DCSS protocol. To improve this protocol the UE sends its ID (FID) to SPAAA in message 1 and to SS in message 3, SPAAA sends its ID ($SPID$) to UE in message 2, and the service server sends its ID ($SSID$) in message 4. After the changes were made, the protocol was modelled and run again in ProVerif as shown in Fig. 7.20, there was no attack found on the protocol, hence the protocol is secure. The above ProVerif results indicate that the secrecy of `AcK_1, Ack_2, Fid, d1, ChT, ShT` holds. The authentication event in the form of non-injective and injective agreements is true.



```
ededris@ededris-VirtualBox:~/proverif2.00$ ./proverif protocols/DCSS.pv | grep RES
RESULT not attacker(Ack_1[]) is true.
RESULT not attacker(Ack_2[]) is true.
RESULT not attacker(kue3a[]) is true.
RESULT not attacker(kuess[]) is true.
RESULT not attacker(fid[]) is true.
RESULT not attacker(d1[]) is true.
RESULT not attacker(ChT[]) is true.
RESULT not attacker(ShT[]) is true.
RESULT event(endSS(U,SS,K)) ==> event(beginUE(U,SS,K)) is true.
RESULT event(endUE(U_132,SS_133,K_134)) ==> event(beginSS(U_132,SS_133,K_134)) is true.
RESULT inj-event(endSS(U_135,SS_136,K_137)) ==> inj-event(beginUE(U_135,SS_136,K_137)) is true.
RESULT inj-event(endUE(U_138,SS_139,K_140)) ==> inj-event(beginSS(U_138,SS_139,K_140)) is true.
```

Figure 7.20: DCSS Safe ProVerif Results

7.15.4 Security Analysis

Protocol Security Analysis

This analysis is for the improved version of DCSS protocol based on security requirements sets 1 and 2 as presented in chapter 3.
Analysis based on security properties of set 1 is as follows:

- Secrecy: This is achieved since the FID and $D1$ are never revealed to the attacker. By achieving this property also confidentiality and privacy of the protocol data are addressed.
- Aliveness: The SP obtain the aliveness of UE when UE sends a cache authorization request to SP with AcT, while SS does when UE sends a cache or share message to SS and in return, it receives tokens (ChT, ShT) and Acknowledgement strings $(Ack\_1, andAck\_2)$, respectively.
- Weak Agreement: This is achieved when SP achieves non-injective agreement on AcT with UE while SS achieves this when Acknowledgement strings are sent to the UE.
- Non-injective Agreement: The UE obtains non-injective agreement on AcT with the SP. The SS achieves non-injective agreement on ChT/ShT with UE after it is generated by SP. The token ChT/ShT includes labels; therefore, SS obtains assurance as a non-injective agreement on ChT/ShT from the SP with UE.
- Injective Agreement: The injective agreement on tokens between the SP and SS is central to the protocol's purpose. The injective agreement on labels with the SP assures the UE that SS is known and trusted. The UE obtain the injective agreement on ChT/ShT and AcT with the SP and SS, respectively to assure that the sessions with SS were authorized by the SP. At the same time, SS is assured that its sessions with the UE were authorized by SP.

Analysis based on security properties of set 2 is as follows:

- Mutual Entity Authentication: Since the UE is already authenticated with SP and uses SSO for further authorization, it uses AcT to acquire both cache/share tokens from SP, it enforces weak agreement and implicit authentication between SS and UE. In addition, reverifies the AcT, the SS verifies the ChT/ShT and in return, it accepts the UE's caching/sharing requests by sending Acknowledgements. The FID and $Ack1/Ack2$ proved to hold which also enforces this requirement.
- Mutual Key Authentication: This property is not required as the involved parties are in a possession of session keys K_{UE3A} and K_{UESS}.
- Mutual Key Confirmation: The successful run of the protocol between the UE, SP, and SS enforces this requirement.
- Key Freshness: ProVerif has no function to check key freshness, however, the SP checks if token expiry date and timestamp are included in the message, enforcing the freshness of the session, but this does not check key freshness. However, since the secrecy of these keys is not violated, it implies keys are fresh.
- Unknown-Key Share: The reachability property in ProVerif is used to check aliveness. The entity's ID and key binding prevent this attack. The inclusion of FID, $SPID$, and AcT parameters; EID, $ASID$ and labels in the authorization process proves this requirement.

- Key Compromise Impersonation Resilience: The UE, SPAAA, and SS are in possession of K_{UE3A} and K_{UESS} when the protocol starts to run which are pre-shared keys. Since the secrecy property of all data exchanged holds hence, they enforce this requirement. Further enforcement of this requirement is achieved when digital signatures are used in the message exchange.

Security consideration

The UE will be able to refresh the access token and request cache and share tokens for other services as defined by the SP with ACL claims and security parameters. The use of FID and security tokens enables the UE and SP to start a new authorization process that might require new parameters. With SSO the UE can request other services which reduce exposure to adversaries, while DCSS protocol enables the UE to request further authorization from SP to cache and share data. It will also be able to share data and security tokens with other UEs, enabling access right delegation to other UEs. This protocol will provide authorization, implicit authentication to the UE and SP which is another security assurance in the integrated security solution proposed in this study, ensuring secure communication, user experience, and utilization of network resources. Therefore, DCSS protocol addresses data caching and sharing issues at the SLS as discussed in chapter 5. As the UE is granted access to a service, it must request further authorization if it intends to cache and share the data. Whereby during the generation of the security token, the right access permissions and labels can be included in tokens and data security.

7.16 Summary

This chapter presented and formally analysed the proposed secondary authentication and service authorization protocols that will enable the UE to access services from any SP in HN or DN using FID and SSO. These protocols allow the UE to access services, cache data, get permission to share data, and delegate access rights to others. The next chapter presents the DDS protocols that enable the UEs in proximity to authenticate each other and share data as well as delegate their authorization permissions to other UEs in different scenarios.

Chapter 8

Device-to-Device Security

8.1 Introduction

Primary authentication methods such as 5G-AKA and EAP-AKA' presented in chapter 6 are used to authenticate the UE before being granted access to the HN. After a successful authentication, usually, the UE gets static service authorization but due to 5G's characteristics and objectives, multi-level service authorization mechanisms like those presented in chapter 7 will have to be applied to allow the UE access to services in different security domains and interoperability between networks. With NAC and SLS protocols presented and proved to be unflawed, they can provide UE security at different levels of the network and services. The next step is to define security mechanisms that can secure data access, caching, and sharing between two UEs. Therefore, this chapter proposes a DDS solution and the underlying security protocols. It discusses the need for network and non-network assisted security methods.

After gaining access to the network and services, two UEs in proximity of each other can engage in D2D communications to share data. The D2D communication process can be initiated by performing device discovery, link setup, and communication steps. Each step needs to be protected from any possible attacks as discussed in chapter 4, using security measures proposed in this chapter. Some of the work in this chapter is also presented in (Edris et al. 2021*b*).

The rest of this chapter is structured as follows. Section 8.2 presents the proposed D2D security solution. The DDSec protocol modelling is presented in section 8.3. Section 8.4 presents the verification of DDSec protocol. The DDACap protocol modelling is presented in section 8.5. Section 8.6 presents the verification of DDACap protocol. This chapter is summarised in section 8.7.

8.2 The Proposed D2D Service Security Solution

The proposed solution is composed of two protocols for D2D security that can be applied to two different scenarios. One is based on network-assisted D2D communication relying on gNB for discovery and initiation of communication and the other is based on direct D2D communication whereby the communication and discovery are controlled and managed by the D2D UEs without a need of a central authority for connectivity and data distribution. In addition, the UEs can choose any publicly known information as its ID, the UE's public key, or IP address based on IBE scheme (Boneh & Franklin 2003), it can be used with another form of ID such as FId or EID. If the UE does not have either IDs assigned from previous security procedures like primary or secondary authentication in chapters 6, 7, respectively, GPSI is used. Hence, no need to distribute the keys as in the traditional AKA or PKI or using SUPI outside the HN. For integrity, privacy and uniqueness, the UE uses a pseudonym ID ($UEID$) in D2D communications, calculated from the UE's preferred choice of ID, $hash(x(ID))$, where x is the public key of UE, randomly generated using hash function $(UEID) = hash(PK_{UE_i}(ID)$ (Altmann et al. 2014).

As mentioned earlier, the original NDO is published by the content provider/data owner using a URI of the data as the data ID or data name. For service discovery, a UE searches for specific data by the data ID, and some contacts from the owner's routing table are requested, which are in the search tolerance of the data ID (Altmann et al. 2014). The searching UE knows only the desired data object name and thus can generate only one hash value. Additionally, the UE must learn the ID mapping, so when the UE with the data wants to publish, first it calculates the logical/node address *nAdd*. The relation between UE and data is determined by pseudonym IDs of the UE and the data which becomes the logical address *nAdd* of the UE with the data. The pseudonym ID of the data is $D1$, generated by hashing the data name $hash(DataName)$. The UE_i would first hash both the $UEID$ and $D1$ as $hash(UEID, D1)$ then publish it as its logical address. The keys and IDs are generated using ECIES (SECG 2009) according to 5G standard (3GPP 2020*f*). Whereby *nAdd* $=hash(UEID, D1)$, the offered service types, the NDO, and a UEID are mapped together. The UE interested in the data uses the logical address to send a query to the UE in possession of the data.

Our proposed solution intends to provide a secure exchange of data in coverage or out-of-coverage scenarios. The D2D discovery, communication, and content dissemination processes are discussed in chapter 4. The content discovery is based on CCN interest and data messages, any UE can take part and retrieve data from the UE in possession of the data. The UE with the data verifies the requesting UE before authorizing the request. This mode provides autonomy, decentralization, authorization, scalability, fault tolerance, data integrity, and authentication during D2D data sharing.

Keys Derivation

Based on ECIES, each UE generates its own self-certifying public key, PK_{UE_i} and its corresponding private key SK_{UE_i} (Girault 1991). In this study, the private/public key generation is done at each of the two communicating parties UE_i. Whereby UE_i must generate its respective secret key SK_{UE_i} by selecting a random number using chosen binary of ECC then public keys PK_{UE_i} (SECG 2009). Each UE_i sends its public key and pseudonym ID to the gNB in terms of network-assisted application and to each other in a direct communication application. Traditionally certificates would be used to bind the UE_i to its public key, however in this case self-certifying public keys are used. Each UE_i also can estimate the shared channel between them and extract hashed random number generated with a hash function. Furthermore, the UE_i can generate hash functions, symmetric/asymmetric keys, and digital signatures achieved using ECIES. ECIES is suitable for systems with low computational capabilities and resources, such as mobile devices and smart cards due to the point multiplication operation. The proposed protocols use ECIES as applied by 5G to conceal SUPI and generation of the anchor key (3GPP 2020*f*).

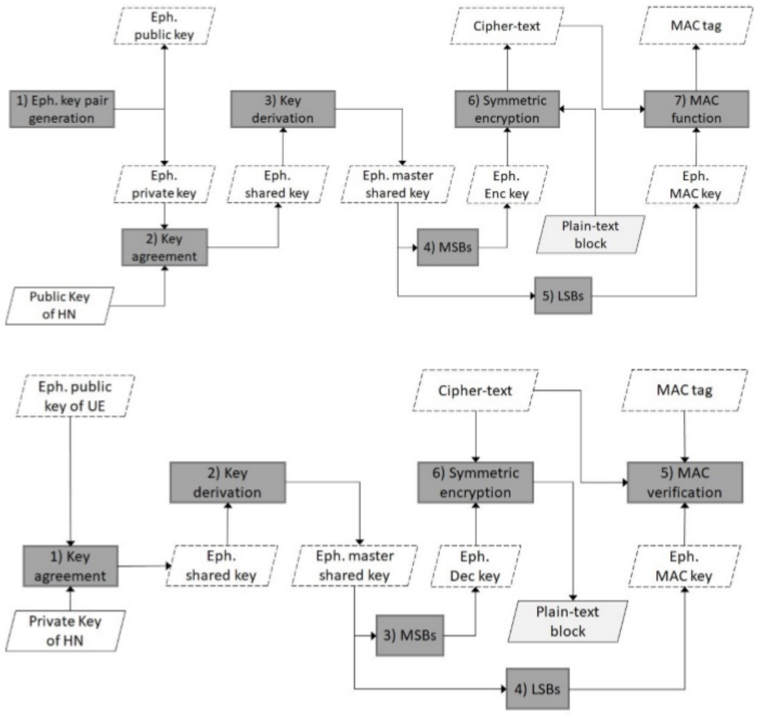


Figure 8.1: ECIES Encryption and Decryption at UE side (3GPP, 2020)

Security Assumptions

Each UE_i can generate its own public/private key, stores its public key PK_{UEi}, SK_{UE_i} and its hash $hash(PK_{UE_i})$ while gNB stores the public keys, pseudonym ID and other relevant information of all users in the cell. Information about a specific UE can be obtained by any interested UE. The UEs involved in D2D communications can authenticate and establish a secure communication channel, generate hash functions, public/private keys through asymmetric cryptography instead of sending authentication requests to HN, achieved by using ECIES. The proposed protocols use ECIES as used by 5G to conceal SUCI and generate the anchor key as shown in Fig. 8.1 and provide perfect forward secrecy (Pardo 2013). In this proposed solution, the UEs and gNB have a pair of public and private keys. The public key is known by all entities including the gNB which acts as a central authority, it can also issue certificates that bind users' IDs to their public keys in case of the network-assisted D2D communications, the private key is secret to all but the owner. The security properties for DDS protocols are also analysed using the taxonomies (Menezes et al. 2018) and (Lowe 1997) presented in chapter 3.

8.2.1 Authentication

In this phase, the D2D communication the parameters such as PK_{UEi}, $hash(PK_{UE_i})$ and nonce are used for authentication. As explained earlier, for instance, communication between UE_A and UE_B after generating their self-certifying key pairs, then they generate their unique IDs (UA) and (UB).

The UE_A must learn the UEID/Dataname mapping $hash(UEID, D1)$ and the same for UE_B. If UE_A has data $D1$ to publish, it uses its logical address $nAdd$ to publish. Later, when node UE_B wants to access $D1$, it uses $nAdd$ to send a query to UE_A for obtaining more details and initiating authentication procedure. It sends a service request with $UEID$ to gNB for D2D services, where the gNB maintains a registration table to associate the UE_i with PK_{UE_i} and service information.

Then authentication process begins between the two UEs after authentication UE_B proceeds with requesting the data, fetching the data $D1$ via conventional routing mechanisms of D2D communications and CCN processes. For routing, any UE can access the UE responsible for the D1 and to each UE along the routing path, providing access to UE_A and D1 without exposing any other interface. The assumption is that users have not communicated before and there are no shared keys between them. In a case where they have, there is no need for authentication but can use cached security context to re-authenticate if required and context is still valid. After the discovery and communication process, the two D2D perform mutual authentication of each other before exchange of data securely.

8.2.2 Authorization

In this phase, in application 1 when a UE requested content and cannot be resolved by BBU pool then it can be resolved by UE in proximity. The two proximity users discover each other, establish a secure D2D communication link without involving their HN or gNB. However, in application 2 the D2D devices can communicate in coverage and when out of coverage. In both applications, the D2D device can share the data securely without involving the BBU pool in the authentication and authorization of content even though the SP still has control of who accesses the data and for how long. There are two types of sharing applications and are as follows:

1. When the D2D UEs are in proximity both their HNs have full service agreements including FIdM with VN in coverage and out-of-coverage, then cache and share tokens can be used.

2. When the D2D UEs are in proximity but one of the UE's HN may have partial service agreement hence permission delegation can be used.

In application 1, the authorization of data can be achieved through tokens, while in application 2 this can be achieved with the delegation of access rights to the user, capability, and attributes of the user and data itself. The UE uses FId when communicating to other UE with FId. In application 2, the UE uses GPSI/public key as its global profile to communicate with FId authorized UE. The UE generates the hash value of global ID to communicate to other UEs in proximity. UEs can get authorized to access, cache, and share the data without involving a central authority even though the security context such as data encryption keys might be set by a third-party. Therefore, the authorization procedure might depend on the MNO set policies and service agreements between UE and HN or SP.

As discussed in chapter 7, the SP generates capabilities in form of labels for the UE and data object and each object specifies a set of access rights that can be granted to a UE, these capabilities are transferable through the delegation process. The object's data includes data name, metadata, payload, and other security configuration. The metadata consists of the publication date, expiry date, and created time while the payload consists of ACLs configured in the content. The access attributes and capabilities are specified in the ACL profile within the payload of the data object which includes the name (key id) of the symmetric key (nonce key). The ACap token is created by the UE possessing the data object and the ACap = (D1, DA, AR, L, hKd, exp) defined in Table

Table 8.1: ACap Token Attributes

Attributes	**Description**
D1	hash of the data name of the data object
DA	delegated access: dataset delegation (true)
label	ucap 2.1.2/dcap 0.1.2
hSPid	data owner id
UE_I	ACap owner
Scope	(offline access/cache/share)
hKd	nonce key id signed by SK_{SP}
Exp	expiry date of ACap

8.1. The ACap token is made up of DA, which includes the dataset delegation set to true that permits the delegation of attributes and capabilities to other users. The scope is the access rights of the user, it includes the SPID and user ID. The nonce key is the symmetric key for decrypting the data object. The data object capabilities include whether it should be cached or shared by entities and for how long.

The next sections present the DDSec and DDACap Protocols which intend to provide authentication and authorization to cache, share data and allow the transfer of access capabilities to other UEs. The DDSec protocol intends to provide security for D2D devices in a network-assisted communication scenario while the DDACap protocol intends to provide security for D2D devices in direct communication scenarios.

8.3 Modelling of DDSec Protocol

This section models the proposed DDSec protocol using UE_A, UE_B and gNB entities based on architecture shown in Fig. 4.1 and scenarios A, B and C presented in chapter 4. This protocol applies cryptographic primitives like those used in 5G-AKA, NS-FId, and DCSS protocols in chapters 6 and 7, respectively to achieve security requirements of 5G enabled D2D communications. The cryptographic primitives used include symmetric and asymmetric encryption, one-way hash function, digital signature, and MAC.

The proposed solution intends to provide mutual authentication and authorization between two UEs to cache and share data assisted by gNB. The UEs would be allowed to access services even in the cases where certain roaming agreements do not apply between different networks or D2D. At this security level, the UE uses the federated security context, however, it is still capable of generating its own cryptographic primitives and security tokens such as asymmetric pair of keys using ECIES. It leverages the security parameters used in NAC and SLS mechanisms proposed so far in this research.

After UE_A is granted access to the service in the form data object, it also gets authorized to cache and share data. The UE_A can share the data and delegate its access right and capabilities to UE_B, the UE_B must be on the same NAC-level, that is primary authentication. The UE_A and UE_B can use the proposed DDSec protocol to authenticate and share data. The UE_A and UE_B can use the proposed DDSec protocol to mutually authenticate and share data assisted by the gNB.

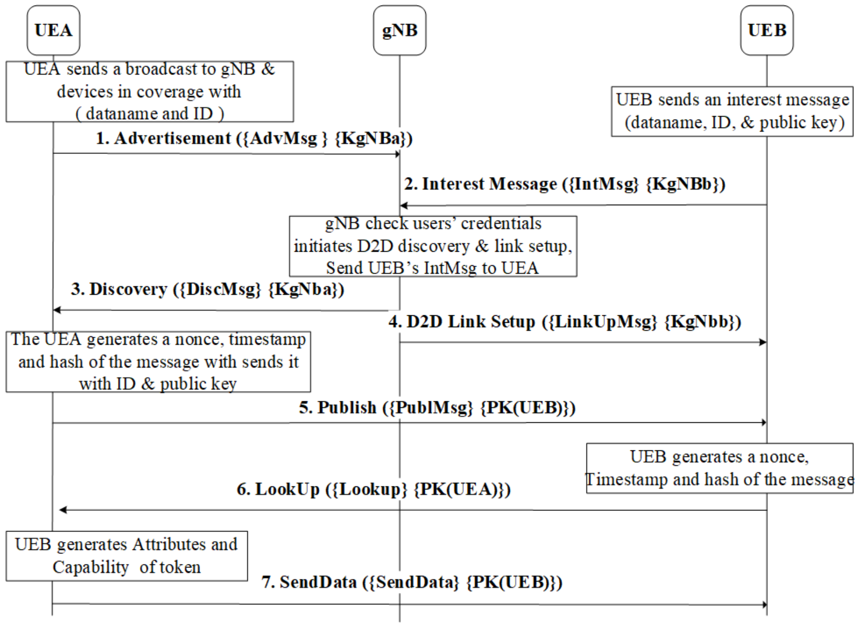

UEA
gNB
UEB
UEA sends a broadcast to gNB & devices in coverage with (dataname and ID)
UEB sends an interest message (dataname, ID, & public key)
1. Advertisement ({AdvMsg } {KgNBa})
2. Interest Message ({IntMsg} {KgNBb})
gNB check users' credentials initiates D2D discovery & link setup, Send UEB's IntMsg to UEA
3. Discovery ({DiscMsg} {KgNba})
4. D2D Link Setup ({LinkUpMsg} {KgNbb})
The UEA generates a nonce, timestamp and hash of the message with sends it with ID & public key
5. Publish ({PublMsg} {PK(UEB)})
UEB generates a nonce, Timestamp and hash of the message
6. LookUp ({Lookup} {PK(UEA)})
UEB generates Attributes and Capability of token
7. SendData ({SendData} {PK(UEB)})

Figure 8.2: DDSec Protocol Message Exchange Flow

8.3.1 Protocol Message Exchange and Execution

This subsection explains the DDSec protocol message and execution, it involves UE_A and UE_A and gNB entities. Before the run of the protocol, the UEs would have achieved a successful network access authentication through primary authentication using 5G-AKA/5G EAP-AKA' protocols presented in chapter 6. The communication between the UEs and gNB is secured with respective keys K_{gNBa} and K_{gNBb}, derived after a successful primary authentication. The UE_A is also in possession of a symmetric key K_{d1} for decrypting the data, received as part of the service authorization procedure that grants the UE access to the service, discussed in chapter 7. The UE_A can advertise to devices in proximity including the gNB about the data in its possession, also the gNB would have information on IDs, previous data transmissions, data names, and data owners. Additionally, when another UE sends an interest message about that data, the gNB assists in the D2D device discovery and link setup between the two UEs. The protocol messages exchange is between UE_A, UE_B and gNB entities for scenarios A, B and C discussed in chapter 4. In the case of B and C, two gNBs would be used, Fig. 8.2 illustrates messages execution for scenario A with reference to notation in Table 8.2 and described as follows:

`Msg1 UEA` $\rightarrow$ `gNB:({AdvMsg},{KgNBa})`

The UE_A sends an advertisement to the gNB that it is in possession of data. The advertisement message $AdvMsg$ includes the data name $DataName$, its $UEID$ (UA), logical address $nAdd$ and public key PK_{UE_A} encrypted with preshared key K_{gNBa} with gNB. Then gNB updates its registration table and BBU Pool updates the intra cell content database with the received message.

`Msg2 UEB` $\rightarrow$ `gNB:({IntMsg},{KgNBb})`

The UE_B sends an interest message $IntMsg$ to the affiliated BBU pool via gNB, it includes its $UEID$ (UB), data name $DataName$ of the data it is interested in and its public key PK_{UE_B}

Table 8.2: DDSec/DDACap Protocol Notation and Description

Notation	**Description**
UEID	UE identifier
Rand_1, 2, ra, rc	nonce
PK(X)	public key
SK(X)	private key
Ack_1/Ack_2	acknowledgement
Hack_1/Hack_2	hash(Ack_1/Ack_2)
Exp	expiry date
D1	hash(Dataname)
Ts	timestamp
ChT_2	cache token
ShT_2	share token
KD1	nonce key
hKD1	hash(KD1) key id
KgNBa/KgNBb	session key UE and gNB
UA/UB/UC	hash(X)UEID))
hUA,hUB/hUC	hash(UA/UB/UC)
ACap	attributes and capabilities token
h(x)	hash value of message x
{x}{k}	message encrypted with key K

encrypted with preshared K_{gNBb} with gNB.

`Msg3.gNB→UEA:({DiscMsg},{KgNBa})`
The gNB with BBU pool check intra cell content database for an interest match within the local cell coverage or the D2D users in proximity via AMF which may have information about other UEs from other cells in intra or inter-operator scenarios. If there is a match, the gNB initiates the D2D communications process of discovery and link setup between UE_A and UE_B via D2D communications supported by SMF. In this case the BBU pool finds a match from UE_A, gNB sets up a link with UE_B and forwards (UB), $DataName$ and PK_{UE_B} to UE_A in $DiscMsg$ message encrypted with key K_{gNBa}.

`Msg4.gNB→UEB:({LinkUpMsg},{KgNBb})`
The gNB forwards UE_A's (UA), data name $DataName$, logical address *nAdd* and public key PK_{UE_A} to UE_B in *LinkUPMsg* message encrypted with key K_{gNBb}.

`Msg5.UEA→UEB:({PublMsg},{PKUEB})`
After the confirmation of link quality of the allocated channel, UE_A initiates the authentication request by sending a *PublMsg* message that includes its $UEID$ (UA), data name $DataName$, pseudonym ID of the $DataName$ ($D1$), a nonce *Rand_a* as authentication challenge, a timestamp Ts_1 and hash of the whole message $hash(UA, DataName, D1, Rand\_a, Ts_1)$ signed with its private key SK_{UE_A} and *PublMsg* is encrypted with UE_B's public key PK_{UE_B} the hashing is for integrity and message authentication, while the timestamp is for replay protection..

`Msg6.UEB→UEA:({LookUp},{PKUEA})`
When UE_B receives message 5, responds with a *LookUp* message that includes its pseudonym ID (UB) and dataname pseudonym ID $D1$, confirming the dataname and its ID. The UE_B also returns

the nonce *Rand_a*, together with new nonce *Rand_b*, a timestamp Ts_2, hash of the whole message $(UB, D1, Rand\_a, Rand\_b, Ts_2)$ signed with its private key SK_{UE_B} and encrypts the *LookUp* message with UE_A's public key PK_{UE_A}. The UE_B authenticates UE_A by sending back *Rand_a* together with *Rand_b*.

`Msg7.UEA→UEB:({SendData},{PKUEB})`

When UE_A receives message 6, it generates attributes and capabilities token *Acap* using the permission delegation authorization granted by the original data owner. Then UE_A returns *Rand_b* to confirm a successful authentication of UE_B together with the requested $DATA$, $Kd1$ symmetric key to decrypt the data, *ACap* token authorizing the UE_B to cache, share data and delegate its access rights to another UE. It also includes the hash of the symmetric key with the token $hash(Kd1, ACap)$ signed with SK_{UE_A} in *SendData* message encrypted with PK_{UE_B}.

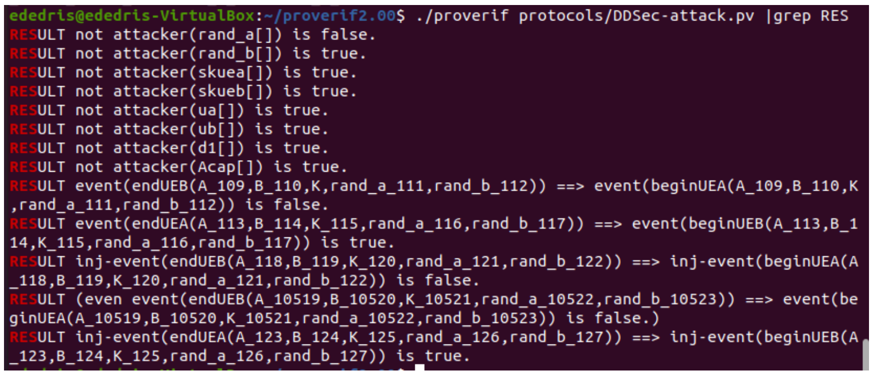
8.4 Formal Verification of DDSec Protocol

This section follows the same ProVerif modelling process as that defined in chapter 3. The protocol modelling used is like that illustrated in chapter 7 with some changes and full ProVerif protocol syntax in Appendix F.1.

8.4.1 Formal Analysis of DDSec Protocol

The protocol was simulated using insecure public channel and processes `processUEA` as UEA, `processUEB` as UEB, and `processgNB` as gNB. When the protocol was modelled, there was an attack found on the public channel between UE_A and UE_B as shown in Fig. 8.3. The security properties the study was interested in are secrecy, mutual authentication, authorization, and secrecy on communication. Using formal analysis, in consideration with adversary vector, there were attacks on the protocol hence the protocol is not secure. ProVerif results show that the secrecy of $rand\_b, skuea, skueb, ua, ub, d1, ACap$ holds but the secrecy of $rand\_a$ and mutual authentication of UE_A to UE_B do not hold that is non-injective and injective agreements. Whereby UE_B may end the protocol thinking it is talking to UE_A while UE_A never run the protocol with UE_B.

The `event beginUEB` means that the UE_B has completed the protocol, that the UE_B received message 2 and sent message 3, `event beginUEB` means that the UE_A sent message 4. These events



```
ededris@ededris-VirtualBox:~/proverif2.00$ ./proverif protocols/DDSec-attack.pv |grep RES
RESULT not attacker(rand_a[]) is false.
RESULT not attacker(rand_b[]) is true.
RESULT not attacker(skuea[]) is true.
RESULT not attacker(skueb[]) is true.
RESULT not attacker(ua[]) is true.
RESULT not attacker(ub[]) is true.
RESULT not attacker(d1[]) is true.
RESULT not attacker(Acap[]) is true.
RESULT event(endUEB(A_109,B_110,K,rand_a_111,rand_b_112)) ==> event(beginUEA(A_109,B_110,K
,rand_a_111,rand_b_112)) is false.
RESULT event(endUEA(A_113,B_114,K_115,rand_a_116,rand_b_117)) ==> event(beginUEB(A_113,B_1
14,K_115,rand_a_116,rand_b_117)) is true.
RESULT inj-event(endUEB(A_118,B_119,K_120,rand_a_121,rand_b_122)) ==> inj-event(beginUEA(A
_118,B_119,K_120,rand_a_121,rand_b_122)) is false.
RESULT (even event(endUEB(A_10519,B_10520,K_10521,rand_a_10522,rand_b_10523)) ==> event(be
ginUEA(A_10519,B_10520,K_10521,rand_a_10522,rand_b_10523)) is false.)
RESULT inj-event(endUEA(A_123,B_124,K_125,rand_a_126,rand_b_127)) ==> inj-event(beginUEB(A
_123,B_124,K_125,rand_a_126,rand_b_127)) is true.
```

Figure 8.3: DDSec Attack ProVerif Results

take parameters of the as arguments; the $A, B, pkueb, rand\_a, rand\_b$ which must verify the ID and respond to a nonce with a nonce. If the arguments are true, then DATA is sent otherwise it sends authentication failure and terminates the communication. This study would like to prove the following correspondence.

```
(*Check authentication between UEA and UEB*)
query A: host, B: host, K: pkey, rand_a:nonce, rand_b:nonce;
event(endUEB(A,B,K,rand_a,rand_b))==> event(beginUEA(A,B,K,rand_a,rand_b)).
inj-event(endUEB(A,B,K,rand_a,rand_b))==>inj-event(beginUEA(A,B,K,rand_a,rand_b)).
```

The direct proof of this correspondence in ProVerif does not hold because message 3 is sent before message 4. This study also tries to prove and conclude the desired correspondence by noticing that event which has $ub, d1, rand\_a, rand\_b, Ts\_1$ as arguments cannot be executed before $ua, d1, rand_a$ has been sent, that is, before $rand\_a$ request has been executed with $rand\_a, rand\_b$ response. Which does not hold in ProVerif.

8.4.2 The Attack Against DDSec Protocol

The ProVerif results show an attack on the protocol as shown in Fig. 8.3. As attack derivation represents the attacker's action while trace represents the real attack as an executable trace of the considered process as explained in chapter 3. The attack derivation and trace sequence of steps, input, and outputs are explained in this subsection.

Attack Derivation and Trace

The attacker I starts by eavesdropping on the communication between entities, impersonates the UE_A, continuing the protocol with UE_B, which completes the protocol with the attacker instead of UE_A. The attacker's actions are illustrated in Fig. 8.4 and explained concisely in derivation and trace steps below, some text is omitted for simplicity, full trace output in Appendix J:

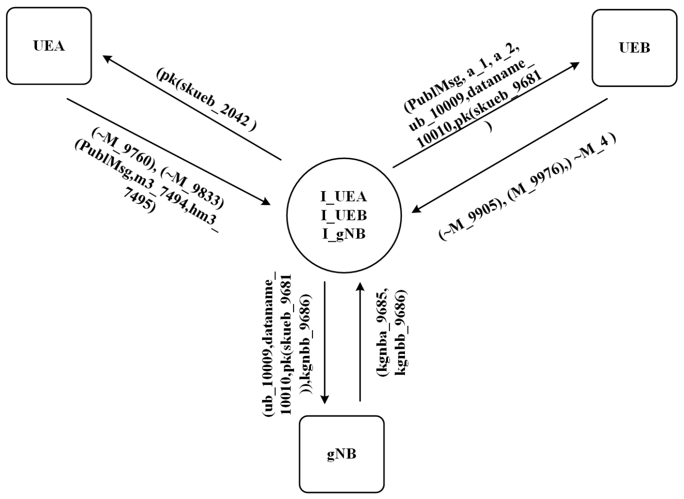

UEA
UEB
(pk(skueb_2042)
(PublMsg, a_1, a_2, ub_10009,dataname_ 10010,pk(skueb_9681)
(~M_9760), (~M_9833) (PublMsg,m3_7494,hm3_ 7495)
(~M_9905), (M_9976),) ~M_4)
I_UEA
I_UEB
I_gNB
(ub_10009,dataname_ 10010,pk(skueb_9681)),kgnbb_9686)
(kgnba_9685, kgnbb_9686)
gNB

Figure 8.4: DDSec Attack Trace

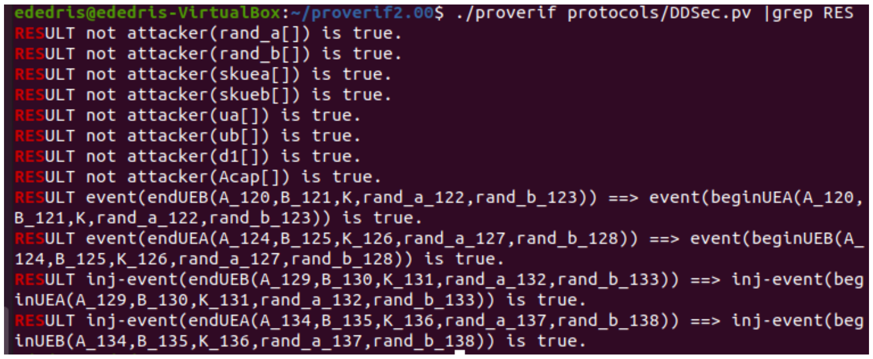
- In the first trace, the attacker knows rand_a, {1} - {6} corresponds to the creation of secret and shared keys, the output of public keys of UE_A and UE_B in the main process. The output {8}, {10}, {12}-{14} corresponds to the public keys, they are stored in fresh variables `~M, ~M_2190`, `~M_2262, ~M_2333` respectively, by the attacker for later reuse. UE_A starts to input the public key of the attacker. In {15} and {16}, the public keys are inserted in the key table in the main process for UE_A and UE_B. Then the attacker gets `rand_a[]`.

- In the second trace, steps 1-3 indicate what the attacker has terms `hm3_7794` and `m3_7494`, using `PublMsg` function 3-tuple the attacker may also obtain `PublMsg`, `attacker(PublMsg)`. In step 5, the attacker may have received message `(PublMsg,m3_7793,hm3_7794)` at input {63}. Inputs {57} and {58} correspond to creations of `ua, ub`, also the attacker creates `ua_8130, ub_8131` and stores them in copy `a_7802`. So, output {62} corresponds to a session `copy a_7802` that UE_B has with the attacker, who sends ID `ub_8132` and dataname `dataname_8132` stored in a new variable `~M_8189` for later use. In the same session the attacker receives `(PublMsg, a_7800, a_7801))` at input {63}. The attacker achieves her goal at {64} when `event endUEB(A, B, pk(skueb_7803), rand_a, rand_b)` is executed in session `a_7802`. That is the entity authentication, key confirmation, unknown key share fail. The trace shows there was an attack against the correspondence on injective agreement.

- In the third trace, the attacker was able to achieve another goal against one-to-one relationship correspondence, when `event endUEB(A, B, pk(skueb_9980), rand_a, rand_b)` is executed in session `a_9977` with UE_B. The trace shows false results on the injective agreement.

8.4.3 Improved Version of DDSec Protocol

To address the security threat discovered in the previous version of the DDSec protocol, this subsection discusses the changes made in modelling of the protocol and presents the improved version of DDSec protocol. To improve this protocol the UE_A send nonce *rand_na* with hash of its public key PK_{UE_A} to gNB in message 1 and while UE_B sends *rand_nb* with hash of its public key PK_{UE_B} to gNB in message 2. The gNB returns *rand_na* along with *rand_nb* in message 3 and which UE_A returns to UE_B in message 4. That prevents the attacker from starting a conversation with UE_B as it will need *rand_nb*, UB and the right hash for the public key to be trusted by UE_B.



```
ededris@ededris-VirtualBox:~/proverif2.00$ ./proverif protocols/DDSec.pv |grep RES
RESULT not attacker(rand_a[]) is true.
RESULT not attacker(rand_b[]) is true.
RESULT not attacker(skuea[]) is true.
RESULT not attacker(skueb[]) is true.
RESULT not attacker(ua[]) is true.
RESULT not attacker(ub[]) is true.
RESULT not attacker(d1[]) is true.
RESULT not attacker(Acap[]) is true.
RESULT event(endUEB(A_120,B_121,K,rand_a_122,rand_b_123)) ==> event(beginUEA(A_120,
B_121,K,rand_a_122,rand_b_123)) is true.
RESULT event(endUEA(A_124,B_125,K_126,rand_a_127,rand_b_128)) ==> event(beginUEB(A_
124,B_125,K_126,rand_a_127,rand_b_128)) is true.
RESULT inj-event(endUEB(A_129,B_130,K_131,rand_a_132,rand_b_133)) ==> inj-event(beg
inUEA(A_129,B_130,K_131,rand_a_132,rand_b_133)) is true.
RESULT inj-event(endUEA(A_134,B_135,K_136,rand_a_137,rand_b_138)) ==> inj-event(beg
inUEB(A_134,B_135,K_136,rand_a_137,rand_b_138)) is true.
```

Figure 8.5: DDSec Safe ProVerif Results

After the changes were made, the protocol was run again as shown in Fig. 8.5, ProVerif found no attack this time hence the protocol is secure. ProVerif full description of the protocol is illustrated in Appendix F.1. ProVerif results indicate that the secrecy of protocol messages holds and mutual authentication of UE_A and UE_B holds.

8.4.4 Security Analysis

Protocol Security Analysis

This analysis is for the improved version of the proposed protocol based on security requirements of sets 1 and 2 as presented in chapter 3.
Analysis based on security properties of set 1 is as follows:

- Secrecy: This is achieved since the `rand_a` and `rand_b` are never revealed to the attacker. By achieving this property also covers confidentiality and privacy of the protocol data.
- Aliveness: The UE_A obtain the aliveness of UE_B when UE_B sends *LookUp* request to UE_A with `ub, hd1, rand_a,rand_b, pkueb, Ts_6`, then UE_A gets non-injective agreement on `rand_a` with UE_B.
- Weak Agreement: This is achieved when UE_A achieves non-injective agreement on `rand_b` with UE_B. Also, the UE_B achieves weak agreement with UE_A on `rand_b`.
- Non-injective Agreement: The UE_A obtains non-injective agreement on `rand_a` with the UE_B. Also, UE_B get non-injective agreement on `rand_b` with UE_A. The non-injective agreement UE_A to UE_B and UE_B to UE_A holds.
- Injective Agreement: The injective agreement on `rand_a`, `rand_b` between the UE_A and UE_B is central to the protocol's purpose. The injective agreement between UE_A and UE_B on PK_{UE_A} and PK_{UE_B} assures each other that they are known and it holds in ProVerif.

Analysis based on security properties of set 2 is as follows:

- Mutual Entity Authentication: The UE_A and UE_B authenticated each other with `A,B, pkuea`, `rand_a`, `rand_b`. When they proved to hold, they enforced this requirement.
- Mutual Key Authentication: The public keys PK_{UE_A} PK_{UE_B} are known to the public and private keys are not, they are verified with a digital signature using their private keys, which implicitly authenticates the self-certifying public keys.
- Mutual Key Confirmation: This requirement does not apply to this protocol as the UEs generate their own public and private keys. However, since they both use ECIES that partly enforces this agreement. The successful authentication of UE_A to UE_B and UE_B to UE_A implicitly enforces this requirement.
- Key Freshness: ProVerif has no function to check key freshness, however, the messages exchange includes random numbers and timestamps hence checking the freshness of messages which include the keys. And since the secrecy of these keys is not violated, it implies the keys are fresh.

- Unknown-Key Share: Since the entities use their own unique public keys/private keys, it enforces this requirement. Also, the reachability property in ProVerif is used to check aliveness. The entities' ID and digital signature prevent this attack. The inclusion of UE_A, UE_B, UA and UB in the authentication process proves this requirement.

- Key Compromise Impersonation Resilience: Since the entities use their own unique public keys/private keys using ECIES, it enforces this requirement. Furthermore, knowing one key in a session is not enough to deduce another. Backward secrecy and forward secrecy of keys are possible, no entity or adversary is capable of computing keys in past sessions or predict future keys, this is due to ECC. Obviously, the public keys are globally known but the private keys are only known to the owners. However, to compromise the keys, the ECIES, UE_A and UE_B will have to be compromised at the same time. Also compromising the HN during the primary authentication does not mean that D2D security will be compromised.

Security consideration

With DDSec the UEs will be able to authenticate each other, share data and delegate their access rights via security tokens. Both UEs use ECC whereby solving ECDLP is not feasible. Therefore, attackers cannot break the proposed protocol's KDF and find the secret key if a big size of the binary elliptic curve is used as it requires very high computational time. The larger the curve size the better the security levels of a protocol. Also, the use of randomness, hash function, tokens, delegation, ID, and certificate digital signature makes the comprise unlikely. Hence, the DDSec protocol will address security issues for D2D communications discussed in chapter 4.

8.5 Modelling of DDACap Protocol

This section models the proposed DDACap protocol using two devices UE_B and UE_C based on architecture shown in Fig. 4.1, scenario D presented in chapter 4 and the improved version of DDSec protocol. This protocol applies similar cryptographic primitives like those used in the DDSec protocol in section 8.3 to achieve security requirements of 5G enabled D2D communications. The proposed DDACap will provide mutual authentication and authorization between two UEs to cache and share data, like the DDSec protocol in a direct communication scenario. The UEs would be able to access and share services with each other without roaming and communication restrictions. For instance, during disaster events or where there is no network coverage. At this security level, the UE uses some of the security contexts from the DDSec protocol if it applies to its scenario, however, it is still capable of generating its own cryptographic primitives and security tokens. It also leverages the security parameters used in NAC and SLS mechanisms.

After UE_A shares the data and delegates its capabilities to UE_B then UE_B can use the same attributes and capabilities to cache the data object and later share it with UE_C in a non-network assisted D2D communications. For UE_B to share data with UE_C, it does not have to be on the same NAC-level as in the case of DDSec protocol. The UE_B and UE_C can use the proposed DDACap protocol to authenticate and share data relying on the subject and object security attributes and capabilities.

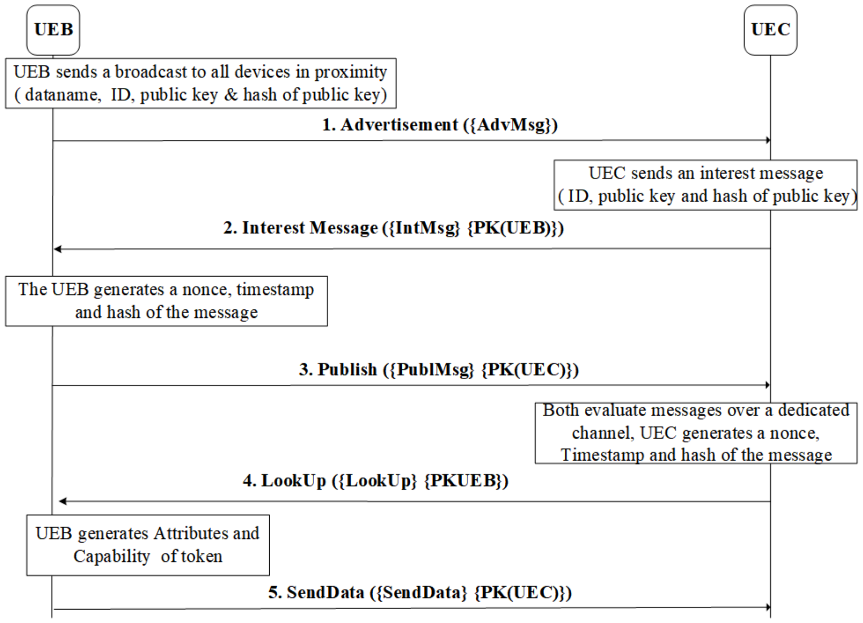

UEB
UEC
UEB sends a broadcast to all devices in proximity (dataname, ID, public key & hash of public key)
1. Advertisement ({AdvMsg})
UEC sends an interest message (ID, public key and hash of public key)
2. Interest Message ({IntMsg} {PK(UEB)})
The UEB generates a nonce, timestamp and hash of the message
3. Publish ({PublMsg} {PK(UEC)})
Both evaluate messages over a dedicated channel, UEC generates a nonce, Timestamp and hash of the message
4. LookUp ({LookUp} {PKUEB})
UEB generates Attributes and Capability of token
5. SendData ({SendData} {PK(UEC)})

Figure 8.6: DDACap Protocol Message Exchange Flow

8.5.1 Protocol Message Exchange and Execution

This subsection explains the DDACap protocol message and execution, involving UE_B and UE_C. The protocol initiation can be performed by the interested UE or the UE with data. In this case, UE_B with data initiates the direct communication by sending beacon signals with advertisement message and UE_C in proximity responds with interest messages. When the messages are evaluated, and the signal has exceeded the predetermined metric threshold then both UEs start to communicate directly. The control signal and the data are exchanged over dedicated control and shared data channels, respectively. The protocol messages exchange between UE_B and UE_C is illustrated in Fig. 8.6, similarly with reference to notations in Table 8.2 and described as follows:

`Msg1 UEB` $\rightarrow$ `UEC:({AdvMsg})`

The UE_B sends beacons signals with an advertisement message $AdvMsg$ by broadcasting to the D2D users in proximity which includes UE_C. It includes its ID (UB), data name $DataName$, logical address $nAdd$, its public key PK_{UE_B} and hash of its public key $hash(PK_{UE_B})$ signed with its private key SK_{UE_B}, the hash is used for integrity and message authentication.

`Msg2 UEC` $\rightarrow$`UEB:({IntMsg},{PKUEB})`

The UE_C sends an interest message $IntMsg$ to UE_B by replying $AdvMsg$. It includes its ID (UC), data name $DataName$ of its interest, its public key PK_{UE_C} and its hash with ($hash(PK_{UE_C})$ signed with its private key SK_{UE_C} and encrypted with UE_B's public key PK_{UE_B} hoping to make a direct communication.

`Msg3.UEB`$\rightarrow$`UEC:({PublMsg},{PKUEC})`

When UE_B receives message 2, responds with a publish message $PublMsg$. The UE_B generates a nonce $Rand\_rb$ as an authentication challenge, a timestamp Ts_3 for freshness and replay protection, pseudonym ID of $DataName$ ($D1$) and hash of the whole message $hPublMsg$

Table 8.3: Secrecy Properties

Properties	UE_B	UE_C
UB	H	H
UC	H	H
Rand_rb	H	H
Rand_rc	H	H

$=hash(UB, D1, Rand\_rb, Ts_3)$ signed with its private key SK_{UE_B}. The UE_B sends a $PublMsg$ message that includes (UB), $D1$, $Rand\_rb$, Ts_3 and $hPublMsg$ as an authentication request encrypted with UE_C's public key PK_{UE_C}.

`Msg4.UEC→UEB:({LookUp},{PKUEB})`

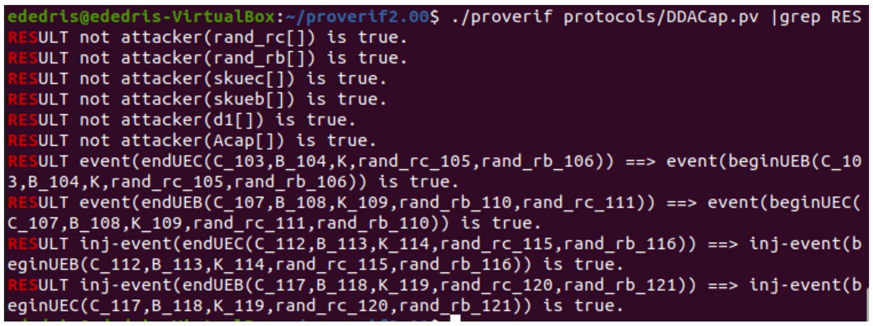
Both UEs evaluate the received messages to establish direct communication over dedicate channels. Then UE_C responds with $LookUp$ message that includes its ID (UC), data name $D1$, new nonce $rand\_rc$, nonce $Rand\_rb$, a timestamp Ts_4 and hash of the whole message $hash(UC, D1, Rand\_rb, Rand\_rc, Ts_4)$ signed by its private key SK_{UE_C}. The $LookUp$ message is encrypted with PK_{UE_B}. The UE_C authenticates UE_B by sending back $Rand\_rb$ together with $Rand\_rc$.

`Msg5.UEB→UEC:({SendData},{PKUEC})`

When UE_B receives message 4, checks if there are any delegated permissions, if there are then generates attributes and capabilities token $ACap1$ including cache and share permissions for the requested data by UE_C. It returns $Rand\_rc$ to confirm a successful authentication of UE_C, together with requested $DATA$, $ACap1$ token with abilities and capabilities string, $Kd1$ symmetric key to decrypt the data and hash of the symmetric key and token $hash(ACap, Kd1)$ signed with SK_{UE_B} to UE_C in $SendData$ message encrypted with PK_{UE_C}.

8.6 Formal Verification of DDACap Protocol

This section follows the same ProVerif modelling process as that defined in chapter 3. The protocol modelling used is like that illustrated in section 8.3 with some changes and full ProVerif protocol syntax in Appendix G.1.



```
ededris@ededris-VirtualBox:~/proverif2.00$ ./proverif protocols/DDACap.pv |grep RES
RESULT not attacker(rand_rc[]) is true.
RESULT not attacker(rand_rb[]) is true.
RESULT not attacker(skuec[]) is true.
RESULT not attacker(skueb[]) is true.
RESULT not attacker(d1[]) is true.
RESULT not attacker(Acap[]) is true.
RESULT event(endUEC(C_103,B_104,K,rand_rc_105,rand_rb_106)) ==> event(beginUEB(C_10
3,B_104,K,rand_rc_105,rand_rb_106)) is true.
RESULT event(endUEB(C_107,B_108,K_109,rand_rb_110,rand_rc_111)) ==> event(beginUEC(
C_107,B_108,K_109,rand_rc_111,rand_rb_110)) is true.
RESULT inj-event(endUEC(C_112,B_113,K_114,rand_rc_115,rand_rb_116)) ==> inj-event(b
eginUEB(C_112,B_113,K_114,rand_rc_115,rand_rb_116)) is true.
RESULT inj-event(endUEB(C_117,B_118,K_119,rand_rc_120,rand_rb_121)) ==> inj-event(b
eginUEC(C_117,B_118,K_119,rand_rc_120,rand_rb_121)) is true.
```

Figure 8.7: DDACap Safe ProVerif Results

8.6.1 Formal Analysis of DDACap Protocol

The protocol was simulated using insecure public channel and processes `ProcessUEB` as UEB and `ProcessUEC` as UEC. When the protocol was modelled, no attack was found on the public channel between UE_B and UE_C as shown in Fig. 8.7 because it was based on the improved version of DDSec protocol in section 8.3. However, to harden the protocol and compensate for not having a central entity, each UE includes a hash of its public key at the beginning of communication with another UE. This increases integrity and reduces replay attacks. The security properties this study is interested in are like that of the DDSec protocol. Using formal analysis, in consideration with adversary vector, there were no attacks on the protocol hence the protocol is secure. ProVerif results above indicate that the secrecy of $rand\_rb, rand\_rc$, SK_{UEB}, SK_{UEC}, $d1$, $ACap$ holds and mutual authentication of UE_B to UE_C holds in form of non-injective and injective agreements as shown in Tables 8.3, 8.4 and 8.5.

The events that prove the correspondence are like that of DDSec protocol and take similar arguments, ID, PK_{UE_I} and nonces. If the arguments are true, then data is sent otherwise it sends authentication failure and terminates the communication. This study would like to prove the following correspondence.

```
(*Check authentication between UEB and UEC*)
query C: host, B: host, K: pkey, rand_rc:nonce, rand_rb:nonce;
event(endUEC(C,B,K,rand_rc,rand_rb))==> event(beginUEB(C,B,K,rand_rc,rand_rb)).
inj-event(endUEC(C,B,K,rand_rc,rand_rb))==> inj-event(beginUEB(C,B,K,rand_rc,
rand_rb)).
```

The direct proof of this correspondence in ProVerif holds because message 3 is sent before message 4. This study also tries to prove and conclude the desired correspondence by noticing that event which has $uc, hd1, rand\_rb, rand\_rc, Ts_3$ as argument cannot be executed before $ub, hd1, rand\_rb$ has been sent, that is before $rand\_rb$ request is has been executed with $rand\_rb, rand\_rc$ response. Which holds in ProVerif with true.

8.6.2 Security Analysis

Protocol Security Analysis

The security analysis of the DDACap protocol is based on security requirements of sets 1 and 2 shown in Tables 8.4 and 8.5 respectively as presented in chapter 3 and like that of DDSec protocol in section 8.3. The query attacker on secrecy, aliveness, weak, non-injective, injective agreement holds. Also, mutual authentication, mutual key authentication, mutual key confirmation, key freshness, unknown key share, and key compromise properties are achieved.

Table 8.4: Security Properties for Set 1

Properties	UE_B - UE_C	UE_C - UE_B
Aliveness	H	H
Weak Agreement	H	H
Non-Injective Agreement	H	H
Injective Agreement	H	H

Table 8.5: Security Properties for Set 2

Properties	UE_B - UE_C	UE_C - UE_B
Mutual Entity Authentication	H	H
Mutual Key Authentication	H	H
Mutual Key Confirmation	H	H
Key Freshness	H	H
Unknown Key Share	H	H
Key Compromise Impersonation Resilience	H	H

Security consideration

Similarly, with DDACap protocol the UEs will be able to authenticate each other and share data as well as delegate their access rights using security tokens via direct communication without network assistance. The security encryption schemes and parameters used are the same as those of the DDSec protocol, hence the same security properties are achieved and the protocol will be able to address D2D communications security issues discussed in chapter 4. The main goal of the proposed two protocols is to provide entity authentication and data authorization between two devices in 5G enabled D2D communications, in network-assisted and non-network assisted scenarios.

8.7 Summary

This chapter proposed two DDS protocols that can be used by two UEs in proximity to share and cache data. The DDSec protocol can be used with network-assisted communication and in coverage scenarios whereas the DDACap protocol could be used in non-network assisted communication and out-of-coverage scenarios. Now that the proposed security protocols are verified and security properties analysed, the next chapter evaluates the performance of all proposed security Protocols in this study. It also explores the integration and possible implementation of the proposed protocols.

Chapter 9

Performance Evaluation and Integration of the Proposed Security Protocols

9.1 Introduction

Now that the proposed security protocols have been specified, verified and analysed, this chapter evaluates security protocols performance using analytical and simulation modelling (Bodei et al. 2005*b*), (Nsnam 2021), respectively for quantification of cryptographic properties of the protocols. It also presents the integration and encapsulation of the proposed protocols to create a comprehensive security solution as well as possible implementation of the proposed protocols. Some of the work in this chapter is also presented in (Edris et al. Submitted for Publication*a*,S).

The rest of this chapter is structured as follows. Section 9.2 presents security protocols performance evaluation based on analytical modelling. In section 9.3 security protocols performance evaluation based on simulation modelling is presented. The integration of the proposed security protocols is discussed in section 9.4. In section 9.5, possible implementation of the proposed protocols is explored. This chapter is summarised in section 9.6.

9.2 Performance Evaluation based on an Analytical Model Approach

Our analytical modelling differs from (Bodei et al. 2005*b*) as ProVerif and applied *pi* calculus are used in this study for protocols specifications as discussed in chapter 3. The evaluation procedure uses special operational semantics of applied *pi* calculus (Abadi et al. 2017), enabling the use of quantitative measures on processes describing cryptographic protocols by deriving Markov chains. Every cryptographic operation and message exchange have a cost on the system, which can be estimated through quantitative measures specified and evaluated based on quantitative properties such as speed and availability (Bodei et al. 2005*b*). The protocols are measured by describing them through a ProVerif process and using labelled semantics (Abadi et al. 2017) to associate a cost with each transition of the system.

The transitions have enhanced labels that associates with cost (Nottegar et al. 2001). This is achieved by assigning rates to transitions of system activities, whereby these rates reflect the distributed architecture of the system model and the use of encryption system such as ECIES and PKI. Then transition systems and Markov chains are mapped together for protocol performance evaluation based on literature in (Bodei et al. 2005*b*). The quantitative measures and qualitative semantics are abstracted symbolically from ProVerif specification of the proposed protocols and the full syntaxes can be found in Appendix H.

Since applied *pi* calculus is used for specifying our protocols, it is easier to estimate and evaluate their cost. The interpretation of the quantities is associated to transitions, includes rates and measures as a cost of using cryptographic primitives. This method supports users input in analysing the actions and performance of the protocols.

9.2.1 Protocol Specifications for Performance Evaluation

This subsection introduces the enhanced operations' semantics, and functions that enhance a system's transitions and costs used in evaluating a protocol's performance. It integrates applied *pi* calculus process with enhanced labels (Bodei et al. 2005*b*), (Blanchet 2016), (Abadi et al. 2017). The system behaviour is illustrated using a graph called transition system defined by operational semantics (Nottegar et al. 2001) as protocol specification. The communication between entities are states and the protocol are *pi* calculus processes, with arcs as translations from one state to another.

Semantic Operations

The applied *pi* calculus grammar used in ProVerif as presented in section 3.3.6 is recalled, the formal specifications of the protocols use channel c as the communication channel, which is associated with input and decryption. ProVerif syntax comprises terms M, N with set of names and processes P, Q with set of variables, and encryptions are tuples of terms $M1, , Mk$. With these main semantics, message M is pushed on channel N by $N(M).Q$ and received by $N(x).P$. The processes Q and

Table 9.1: Enhanced Operation Semantics

$M ::=$	$terms \in \mathcal{M}$
n	name ($n \in \mathcal{N}$)
x	variable ($n \in \mathcal{X}$)
$\{M_1, ..., M_k\}_{E_o}$	symmetric encryption ($k \geq 0$)

$M ::=$	$processes \mathcal{P}$
0	nil
$\langle M_1, ..., M_k \rangle .P$	output
$(M_1, ..., M_j; x_{j+1}, ...x_k).P$	input (with matching)
$P1 \vert P2$	parallel composition
(νn)	restriction
$A(y1, ..., yn)$	constant definition

P stay in parallel with each other, whereby the received message M replaces X. The enhanced semantics are presented in Table 9.1.

The process is prepared to perform reductions by defining semantics using reduction semantic configurations, with E as a finite set of names and P as finite multiset of closed processes. The environment E with at least all free names of processes in P. The configuration $\{a_1, ..., a_n\}, \{P_1, ..., P_n\}$ corresponds intuitively to the process **new** (νa_1)...**new**$(\nu a_n)(P_1|...|P_n)$. The reduction simplifies the computational evaluation of a process in ProVerif (Blanchet 2016). The meta-variables μ_{out} and μ_{in}, resp are used to denote runtime prefixes, whereby the enhanced operational semantics are built on top of a reduction semantics (Bodei et al. 2005*b*). The process $\nu n.P$ makes private name n and acts as P. If $M = N$ then P when Q is 0. With $N(x).P$ ready to input from channel N, then runs P with formal parameter x that replaced the actual message, while $\bar{\text{N}}\langle M\rangle.P$ is ready to output M on channel N, then to run P and omitting P when it is 0 (Abadi et al. 2017). The focus is on normal semantics, new processes and labels that enhance transitions. Using constructors and destructors (Blanchet 2016), the data structure can be represented as tuples and cryptographic operations for encryption and decryption hence modelling perfect symmetric and asymmetric cryptography.

Example 1 *Let us consider one of the protocols SAP-AKA fully modelled in section 7.3 using entities end user (UE), pass through authenticator (SMF) and authentication server (SPAAA), $(UE|SMF|SPAAA)$ as principles to explain this approach.*

The UE, SMF and $SPAAA$ are principals which are running in parallel. ν is binder which binds the key PK_{UEA} and PK_{SP} with the principals, respectively. The processes are extended with active substitutions in Table 9.2, written as $\{^M/_x\}$ to replace the variable x with the term M, $\{^M/_x\}$ can be defined as let $x = M$ in . . ., and which also practical to add a restriction: $\nu x.(M/x|P)$ corresponding to let $x = M$ in P (Abadi et al. 2017). The principals receive and send messages that include terms, associated with input and output. Using parallel composition (|) of the processes, a definite number of activities are executed by each of process before starting again. The new names are created with a restriction operator νnP, that acts as a static binder for n in the process P. The protocols specifications and ProVerif modelling are presented in chapters 6, 7 and 8. The communication between entities using protocol can be shown by using transitions of system from one state to another.

Table 9.2: Processes Extensions with Active Substitutions

$A, B.C ::=$	*extended processes*
P	plain processes
$A \mid B$	parallel composition
$\nu n.A$	name restriction
$\nu x.A$	variable restriction
$\{^M/_x\}$	active substitution
$L ::=$	*labels*
$\nu n.P$	name restriction (”new”)
$N(x).P$	message input
$\bar{\text{N}}\langle M\rangle.P$	message output

9.2.2 Enhanced Labels

After specification, an enhanced label is associated to each transition of the system i.e., each communication and to each decryption (Bodei et al. 2005*b*), by using enhanced operational semantics based on labelled bisimilarity (Abadi et al. 2017) used to prove observation equivalence. The enhanced label (l) records communication output and input elements with their syntactic contexts, facilitated by specific processes creating finite state spaces. With labelled operational semantics, each enhanced label is associated to each transition of ProVerif processes i.e.,`((!procUE()) (!procSMF()) (!procSPAAA())`.

A context label ϑ is associated to each prefix of a specific process. During the building of the labels, parallel compositions and restriction construct are considered by differentiating the type of name restriction created. A $\parallel 0$ (resp.$\parallel 1$) for the left branch of a parallel composition and a ν_a for each restriction of the name a are introduced. Each prefix is associated by a context label (Bodei et al. 2005*b*). The enhanced labels of transitions systems show input and put during communication between entities.

Labelled transitions $\overset{\vartheta}{\to}$ occurs when an output P is the same as an input P'. That is matching the term from an encryption M_o against the decryption pattern $M\prime_o$. The reduction $\overset{\vartheta}{\to}$ ends under parallel composition and restriction and only used for communication labels. With ProVerif processes, the output or input prefix are improved with sequences ϑ of tags such as ν_n, $\|_0$ or $\|_1$. If the prefix appears after a restriction, the tag ν_n occurs in the sequence and tag $\|_0$ (resp$\|_1$) occurs, if the prefix is moved in the left branch of a parallel composition (Bodei et al. 2005*b*).

Example 2 *The state transitions of the protocol are preceded by the sequence of tags which can be reduced as*

.

- $\vartheta_{UE} = \nu_{PKUE}\nu_{PKSP}\nu_{ServName}\nu_{SID} \parallel 0 \parallel 0$ preceding the prefixes of UE.
- $\vartheta_{SMF} = \nu_{PKUE}\nu_{PKSP} \parallel 0 \parallel 1$ (resp. $\vartheta'_{SPAAA} = \nu_{PKUE}\nu_{PKSP}\nu_{GPSI}\nu_{SPID} \parallel 0 \parallel 1$) preceding the first input of SMF.
- $\vartheta_{SPAAA} = \nu_{PKUE}\nu_{PKSP} \parallel 1$ (resp. $\vartheta' SPAAA = \nu_{PKUE}\nu_{PKSP}\nu_{Identity}\nu_{KUESP} \parallel 1$) preceding the first input of $SPAAA$.

In the transition system graph as shown in Fig. 9.1, processes are the entities while arcs are the possible transitions between those entities. Labelled operational semantics enables reason about processes, states, and transitions. The labelled semantics define a relation $P \overset{(\alpha)}{\to} P'$ referring to abstract transitions from states P and P' in the form of $P \xrightarrow{(label,caption)} P'$, the multi states transitions are presented as $P \xrightarrow{(label,caption)} P'^1$, P'^2,P'^n. Where α can be a label of any of the following as presented in Table 9.2:

- a label $N(M)$, where M is a term that may contain names and variables, corresponding to an input of M on N;
- a label $\bar{N}\langle u\rangle$ or $\nu u.\bar{N}\langle u\rangle$, where u refers to a channel name or a variable type, which corresponds to an output of u on N.

This represents the label of the transition and the part of the protocol in transition, such as 1 : UE→SMF, for the communication between UE and SMF.

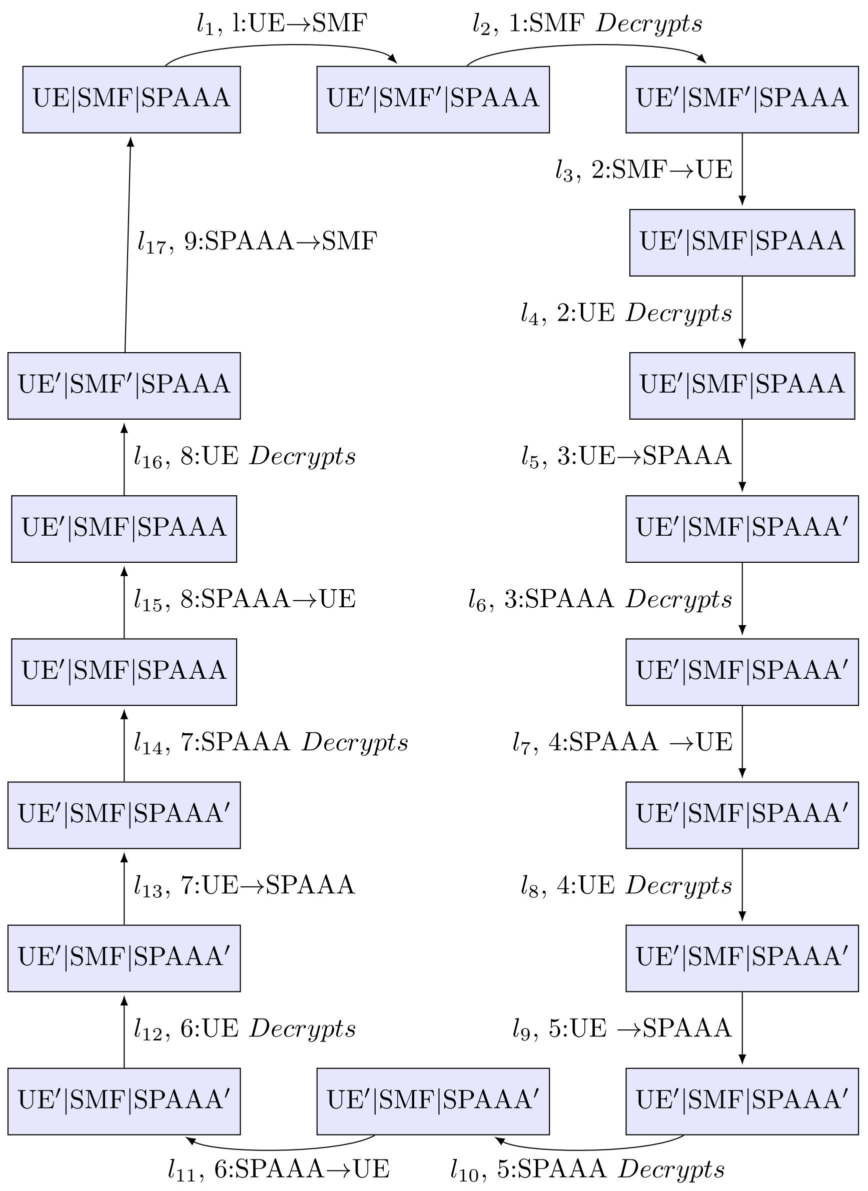


Figure 9.1: SAP-AKA State Transition System

9.2.3 Defining the Cost of a Protocol as a Process

The transitions attained from the enhanced labels are allocated cost by the cost function `$(·)` (Bodei et al. 2005*b*), whereby the cost is any quantitative measure that effects transitions' properties like cryptographic procedure that apply encryption and decryption. The cost of transitions is derived by inspecting enhanced labels, the time that the system is probable to remain within transitions is measured to get the cost. Therefore, the cost of the protocol can be specified based on the time overhead of the activities of the primitives.

Additionally, the cost function associates a cost with each transition labelled by θ, where $\theta = (out, in|dec)$, representing the transition rate, which is the parameter that identifies the exponential distribution of the duration times of θ (Bodei et al. 2005*b*) and action of the originating transition $\alpha = l(\theta)$. Moreover, the cost the label element $\vartheta\mu$ relies on the action μ and on the context ϑ. A scaling factor r is introduced in correlation with each procedure of the transition θ under consideration. The costs are assigned to terms and components of activities μ_{in}, μ_{out} represented by functions as follows:

- $f_u(n)$ and $f_u(n)$ are the cost of unary terms and functions.
- f_{enc} is the cost function for the encryption which computes the cost of the routines that implement the encryption algorithm.
- f_{in} and f_{out} specify the costs of the procedure which applies sending and receiving primitives,
- $f = (j)$ is the cost function for a pattern matching of size j.
- $f_{kind}(crypt)$ indicates an encryption or a decryption costs in line with the encryption scheme.
- $f_{kind}(MO)$ indicates the cost based on the used key.
- $f_{size}(ctxt)$ indicates the cost based on size of the cleartext to encrypt.
- $f_{size}(msg)$ indicates the cost based on the size of message created as the result of the evaluation of the cost of an output or an input.
- $f = (j)$ computes the cost because of pattern matching on j.

$\$_T(n) = f_u(n)$
$\$_T(x) = f_u(x)$
$\$_T(\{M_1, ..., M_n\}_{M_0}) = f_{enc}(f_{kind}(crypt), f_{size}(ctxt), f_{kind}(M_0),\ \$_T(M_1, ..., M_k))$
$\$_T((M_1, ..., M_n) = min\$_T(M_1), ..., \$(M_n))$
$\$_{in}(\mu_{in} = f_{in}(f_{size}(msg), f = (j), \$_T(M_1, ..., M_j))$
$\$_{out}(\mu_{out} = f_{out}(f_{size}(msg), \$_T(M_1, ..., M_k))$

The number of np of processes available determine how the parallel composition is evaluated, $\$_o(\|) = 1$ for an unbound number of ProVerif processes not including the communication cost. In that the number of names $n(P)$ of process P determines cost of restriction as defined in (Bodei et al. 2005*b*), it also depends on the name $f_{kind}(a)$ such as nonce, key, hash function, MAC. Therefore, the label of the transition is $\langle\vartheta \parallel_i \vartheta_{in}\mu_{in}, \vartheta \parallel_{1-i} \vartheta_{out}\mu_{out}\rangle$, which records the actual communication operations. In addition, an exponential distribution with rate r determines the time parameter Δt, that is needed to have a probability near to 1 in relation with Markov chains. By estimating the corresponding duration with a fixed rate $r = min\{\$_{in}(\|_i\ \vartheta_{in}\mu_{in}),\ \$_{out}(\|_{1-i}\ \vartheta_{out}\mu_{out})\}$ as a minimum cost as the communication occurs at the same time. If the label is for decryption $\langle dec\rangle$, f_{dec} is used to compute the cost to derive the decryption algorithm cost. Therefore, cost is defined by induction on θ and by using the functions $\$\mu$ as basis, and then $\$_o$.

Definition 1 *Whereby the cost function is i = 0, 1 as defined by (Bodei et al. 2005*b*)*

$\$(\mu) = \$\mu(\mu)$
$\$(o\theta) = \$o(O) \times \$(\theta)$
$\$(\|_i\ \theta) = \$o(\|\ i) \times \$(\theta)$
$\$(\langle\vartheta\ \|_i\ \vartheta_{in}\mu_{in}, \vartheta\ \|\ 1 - i\vartheta_{out}\mu_{out}\rangle) = \$(\vartheta) \times \min\$(\|_i\ \vartheta_{in}\mu_{in}), \$(\|_{1-i}\ \vartheta_{out}\mu_{out})$
$\$(\langle dec\rangle) = f_{dec}(f_{kind}(crypt), f_{size}(ctxt), f = (j), f_{kind}(MO), \$T(M_1, ..., M_j))$

Cost Metrics

Next is to specify the costs to tune a probabilistic distribution relating to the expected speed of actions as the cost is influenced by the following factors:

- The input, output components and context determine the cost of communication, whereby the cost of an output depends on the message size and the cost of each part of the message.
- The type of algorithm, size of the cleartext and type of key, determine the cost of an encryption together which is not constant.
- The size of the message, the cost of the terms to be matched and the number of checks made before accepting the message determine the cost of an input.
- The size of the ciphertext, type of algorithm and the key used determine the cost of decryption of a ciphertext.
- The number of checks made during the decryption also determine the decryption cost even though it does not depend on its context.

Cost function parameters are used to reflect on the architecture and encryption scheme, taking in count the number processes and cryptographic algorithms (Bodei et al. 2005*b*). Whereby the cost only considered due to parallel composition. To associated a cost to each transition in the system, while computing the performance, the cost due to restrictions is neglected. Also, since it can be assumed that each principal might have a processing unit of its own, cost 1 can be given to each tag $\|\ i$ (i = 0, 1). The context is ignored, and the same cost is given to output and input. In a transition, communication is assigned a cost equal to $n * s + \sum_{i=1}^{l} m_i * e$, decryption is assigned a cost equal to $n * d$ and terms are described in Table 9.3.

Example 3 *The cost of the third transition of SAP-AKA protocol H.3 with label*

Table 9.3: Cost Description

Term	**Description**
n	size of the message
m_i	size of the ith encryption
e	cost of unitary encryption
d	cost of unitary decryption
s	cost of unitary output
l_i	label in relation to the state
c_i	cost in relation to the label

$l_3 = \langle \vartheta_{UE}(\{ServName, SID, SPID\}_{PK_{SP}}), \vartheta_{SPAAA}(t^{UE}_{enc}\rangle$ (msg3), the cost is 3s + 3e = C, whereby the output message is 3S and the size of the encrypted cleartext is 3e. The cost of decryption of this message in the third transition is 3d, that is for a ciphertext decryption back to a cleartext of size 3e. The full list of the protocols' costs c_i in relations to their labels l_i are presented in section 9.2.7. The cost parameters vary this due to the difference system architecture, protocol, encryption scheme, algorithm and cryptographic primitives used. For instance in 5G, ECC, SQN, AKA challenge, XOR must be considered. The cost of communication and encryption are affected by speed operations and the communications link.

9.2.4 Continuous Time Markov Chains

The required quantitative information is extracted to derive the Continuous Time Markov Chains (CTMC) (Stewart 1994) using enhanced operational semantics by mapping transition system to Markov chains. The CTMC compromises of sets of states and labelled transitions between the states with a sequence of random values, the probabilities of these values at a time interval depends on the values of the previous states (Hillston 2005). As explained earlier, a function is used to assign costs to individual transition to enable an application of costs to tune a probabilistic distribution. These costs are interpreted as parameters of exponential distributions (Bodei et al. 2005*b*). After computing the exponential distributions of transitions, it leads to numerical process because of collapsing the arcs which share source and target.

However, the next transition appearance does not depend on when the last transition appeared. The assumption is that all transitions are time homogeneous, so the rate of a transition does not depend on the time at which it happens. The parameter r is associated with a transition to extract some transition probabilities, that is the rate at which a system changes from process P_i to behaving like P_j. Hence, corresponding to the sum of the costs of all the transitions that are performed from P_i to P_j. Moreover, there is only one transition between any two entities and so rates coincide with single costs within a transition system.

Definition 2 *The transition rate between two states P_i and P_j, written $q(P_i, P_j)$, is the rate at which the transitions between P_i and P_j occur (Bodei et al. 2005*b*)*

$$q(P_i, P_j) = \sum_{P_i \xrightarrow{\theta_k} P_j} \$(\theta_k). \tag{9.1}$$

A directed graph is used to represent a CTMC C, in that the entities are the states of C, only the states that can reach each other are connected by the arcs. Whereby the rates at which the process moves from one state to another is illustrated by a square matrix $\mathcal{Q}$, called *generator matrix*, which is the adjacency matrix of the graph representation of the CTMC of process ($CTMC(P)$). The entries of Q are instantaneous transition rates defined as 3.2 (Bodei et al. 2005*b*).

$$q_{ij} = \begin{cases} q(P_i, P_j) = \sum\limits_{P_i \xrightarrow{\theta_k} P_j} \$(\theta_k) & \text{if } i \neq j \\ \\ -\sum\limits_{\substack{j=1 \\ j \neq i}}^{n} q_{ij} & \text{if } i = j \end{cases} \tag{9.2}$$

9.2.5 Quantitative Measure of a Process using CTMC

After long periods executions the performance measures of systems become comprehensible. The derivation of these measures of process P are achieved by exploiting the stationary probability distribution Π for the CTMC, associating it with P, since they have finite and cyclic properties.

Definition 3 *Let $\Pi^t(x_i) = p(X(t) = x_i)$ be the probability that a CTMC is in the state x_i at time t, and let $\Pi^0 = (\Pi^0(x_0), ..., \Pi^0(x_n))$ be the initial distribution of states $x_0, x_1, ..., x_n$. Then a CTMC has a stationary probability distribution $\Pi = (\Pi(x_0), ..., \Pi(x_n))$ if (Bodei et al. 2005*b*)*

$$\Pi Q = 0 \; and \sum_{i=0}^{n} \Pi(x_i) = 1. \tag{9.3}$$

The stationary distribution for each of the system solves the system linear equations 3.4. The standard numerical techniques are used to utilize the preferred numerical package available for needed computations and the stochastic analysis. The stationary distribution of the Markov chains is $\Pi_i = (X_0, ..., X_{n-1})(i = 1, 2 \text{ and } n = 6, 8)$ for the protocols and corresponding equation as defined in *definition 3* and solution of the following linear equations of each proposed protocol is presented in subsections 9.2.7.

$$\Pi Q = 0 \; and \sum_{i=0}^{n-1} X_i = 1. \tag{9.4}$$

9.2.6 Reward Structure of a Protocol as Process

A process P performance is measured by associating it with a reward structure based on (Hillston 2005). The reward structure of the performance model is a function that associates a value with any state passed through in a computation of P this due to the performance model being CTMC (Nottegar et al. 2001).

Definition 4 *Given a function $\rho\theta$ associating as a transition reward with each transition θ in a transition system, the reward of a state P is (Bodei et al. 2005*b*)*

$$\rho p = \sum_{P \xrightarrow{\theta} Q} \rho\theta. \tag{9.5}$$

The reward structure of a process P is described as a vector of rewards with many elements in relations to the number of derivatives of P. From that and the stationary distribution Π of $CTMC$ of a process P performance measures are computed (Bodei et al. 2005*b*).

Definition 5 *Let Π be the stationary distribution of $CTMC(P)$. The total reward of a process P is computed as (Bodei et al. 2005*b*)*

$$R(P) = \sum_{P_i \in d(P)} \rho P_i \; X \; \Pi(P_i). \tag{9.6}$$

The utilization of an encryption scheme is achieved by getting the sum of the values of Π multiplied by the equivalent reward structure. This adds up to the time spent in the states with encryption scheme enabled. Even though the reward structure is just a function that associates a reward with a state going through a computation of process P, we can also compute rewards from rates of transitions (Nottegar et al. 2001). This is achieved by measuring the throughput of the system in terms of amount of work accomplished per unit time using non-zero reward value against the rate of corresponding transition (Bodei et al. 2005*a*).

Definition 6 *Let process P reward structure be $\rho\theta = \rho\theta(0), ..., \rho\theta(n-1)$. The total reward of process P is computed as (Bodei et al. 2005*b*)*

$$R(P) = \sum\nolimits_i \; \rho(i).X_i. \tag{9.7}$$

$$\mathbf{Q1} = \begin{array}{c|ccccccccccccccccc|}
 & l1 & l2 & l3 & l4 & l5 & l6 & l7 & l8 & l9 & l10 & l11 & l12 & l13 & l14 & l15 & l16 & l17 \\
l1 & -b & b & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
l2 & 0 & -2d & 2d & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
l3 & 0 & 0 & -c & c & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
l4 & 0 & 0 & 0 & -3d & 3d & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
l5 & 0 & 0 & 0 & 0 & b & b & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
l6 & 0 & 0 & 0 & 0 & 0 & -2d & 2d & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
l7 & 0 & 0 & 0 & 0 & 0 & 0 & -a & a & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
l8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -d & d & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
l9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -a & a & 0. & 0 & 0 & 0 & 0 & 0 & 0 \\
l10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -d & d & 0 & 0 & 0 & 0 & 0 & 0 \\
l11 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -g & g & 0 & 0 & 0 & 0 & 0 \\
l12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -5d & 5d & 0 & 0 & 0 & 0 \\
l13 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -b & b & 0 & 0 & 0 \\
l14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2d & 2d & 0 & 0 \\
l15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -c & c & 0 \\
l16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -3d & 3d \\
l17 & s & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -s
\end{array} \tag{9.8}$$

Figure 9.2: SAP-AKA Matrix

9.2.7 Performance Evaluation with Markov Chain

Having specified the protocols, defined the labelled enhanced operation semantics and cost function, this section evaluates performance of our proposed protocols using Markov chain as discussed in section 9.2.4 based on the cost definition in Table 9.3. We recall the SAP-AKA, NS-FId, DCSS protocols in chapter 7, DDSec, and DDACap protocols in chapter 8. The stationary distribution of the Markov chains for these protocols and corresponding equation (9.2) as defined in subsection 9.2.4 and solution of the linear equation (9.4) subsection 9.2.5 of each protocol are presented in following subsections. The protocols detailed labels and matrix are in Appendices H and I, respectively.

SAP-AKA Protocol

The state of transition and labels are shown in Fig. 9.1, detailed labels in Appendix H.1 and matrix Q1 is shown in Fig. 9.2, the cost and transition association metrics in Tables 9.4 and 9.6. Consider the transition system which is finite and has cyclic initial states so that it has stationary distributions. The following generator matrix $Q1$ = CTMC (SAP-AKA) is derived and the stationary distributions is Π_1, where $A = 20s + 19e + 19d$.

$$\Pi_1 = \left[\frac{A}{b}, \frac{A}{2d}, \frac{A}{c}, \frac{A}{3d}, \frac{A}{b}, \frac{A}{2d}, \frac{A}{a}, \frac{A}{d}, \frac{A}{a}, \frac{A}{d}, \frac{A}{g}, \frac{A}{5d}, \frac{A}{b}, \frac{A}{2d}, \frac{A}{c}, \frac{A}{3d}, \frac{A}{s}\right] \tag{9.9}$$

Table 9.4: Cost Metrics for the Enhanced Labels of the Proposed Protocols

SAP-AKA	**NS-FId**	**DCSS**	**DDSec**	**DDACap**
c1 = 2s + 2e	c1 = 2s + 2e	c1 = 4s + 4e	c1 = 2s +2e	c1 = 3s
c2 = 2d	c2 = 2d	c2 = 4d	c2 = 2d	c2 = 6s + 6e
c3 = 3s + 3e	c3 = 2s + 2e	c3 = 2s + 2e	c3 = 2s + 2e	c3 = 6d
c4 = 3d	c4 = 2d	c4 = 2d	c4 = 2d	c4 = 6s + 6e
c5 = 2s + 2e	c5 = 3s + 3e	c5 = 2s + 2e	c5 = 5s + 5e	c5 = 6d
c6 = 2d	c6 = 3d	c6 = 2d	c6 = 5d	c6 = 3s + 3e
c7 = s + e	c7 = 3s + 3e	c7 = 2s + 2e	c7 = 7s + 7e	c7 = 3d
c8 = d	c8 = 3d	c8 = 2d	c8 = 7d	c8 = 4s + 4e
c9 = s + e	c9 = 3s + 3e	c9 = 4s + 4e	c9 = 3s + 3e	c9 = 4d
c10 = d	c10 = 3d	c10 = 4d	c10 = 3d	
c11 = 5s + 5e	c11 = 5s + 5e	c11 = 2s + 2e	c11 = 4s + 4e	
c12 = 5d	c12 = 5d	c12 = 2d	c12 = 4d	
c13 = 2s + 2e	c13 = 2s + 2e	c13 = 2s + 2e	c13 = 4s + 4e	
c14 = 2d	c14 = 2d	c14 = 2d	c14 = 4d	
c15 = 3s + 3e	c15 = 4s + 4e	c15 = 2s + 2e		
c16 = 3d	c16 = 4d	c16 = 2d		
c17 = s	c17 = 2s + 2e			
	c18 = 2d			
	c19 = 2s + 2e			
	c20 = 2d			

where:

$$A = \frac{17A}{20s + 19e + 19d} \tag{9.10}$$

NS-FId Protocol

The state of transition with labels, detailed labels, matrix in Appendices K, H.2, I.1 respectively, the cost and transition association metrics in Tables 9.4 and 9.6. Consider the transition system, with the following generator matrix $Q2$ = CTMC (NS-FId) and the stationary distributions Π_2, where $B = 28s + 28e + 28d$.

$$\Pi_2 = \left[\frac{B}{b}, \frac{B}{2d}, \frac{B}{b}, \frac{B}{2d}, \frac{B}{c}, \frac{B}{3d}, \frac{B}{c}, \frac{B}{3d}, \frac{B}{c}, \frac{B}{3d}, \frac{B}{g}, \frac{B}{5d}, \frac{B}{b}, \frac{B}{2d}, \frac{B}{f}, \frac{B}{4d}, \frac{B}{b}, \frac{B}{2d}, \frac{B}{b}, \frac{B}{2d}\right] \tag{9.11}$$

where:

$$B = \frac{20B}{28s + 28e + 28d} \tag{9.12}$$

DCSS Protocol

The state of transition with labels, detailed labels, matrix in Appendices K, H.3, I.2 respectively. The cost and transition association metrics in Tables 9.4 and 9.6. Consider the transition system, with the following generator matrix $Q3$ = CTMC (DCSS) and the stationary distributions Π_3, where $C = 20s + 20e + 20d$.

Table 9.5: Cost Metrics for the Enhanced Labels of Benchmark Protocols

5G-AKA	**D2D-5G (SEOK)**
c1 = 2s + e	c1 = s
c2 = 3s	c2 = s
c3 = 3s	c3 = 5G-AKA
c4 = d	c4 = 5G-AKA
c5 = 5s + 3e	c5 = 5G-AKA
c6 = 3s	c6 = 5G-AKA
c7 = 2s	c7 = s + e
c8 = 3d	c8 = s + e
c9 = s + e	c9 = 2s
c10 = s	c10 = 2s
c11 = s	c11 = 2s
c12 = d	c12 = 2s
c13 = 2s	c13 = d
	c14 = d
	c15 = s
	c16 = s
	c15 = s + e
	c16 = d

Table 9.6: Metrics Variables

Variable	**Description**
a	s + e
b	2s+2e
c	3s+3e
f	4s+4e
g	5s+5e
h	6s+6e
i	7s+7e

$$\Pi_3 = \left[\frac{C}{f}, \frac{C}{4d}, \frac{C}{b}, \frac{C}{2d}, \frac{C}{b}, \frac{C}{2d}, \frac{C}{b}, \frac{C}{2d}, \frac{C}{f}, \frac{C}{4d}, \frac{C}{b}, \frac{C}{2d}, \frac{C}{b}, \frac{C}{2d}, \frac{C}{b}, \frac{C}{2d}\right] \tag{9.13}$$

where:

$$C = \frac{16C}{20s + 20e + 20d} \tag{9.14}$$

DDSec Protocol

The state of transition and labels, detailed labels, matrix in Appendices K, H.4, I.3 respectively, the cost and transition association metrics in Tables 9.4 and 9.6. Consider the transition system with the following generator matrix $Q4$ = CTMC (DDSec) and stationary distributions Π_4, where $D = 27s + 27e + 27d$.

$$\Pi_4 = \left[\frac{D}{b}, \frac{D}{2d}, \frac{D}{b}, \frac{D}{2d}, \frac{D}{g}, \frac{D}{5d}, \frac{D}{i}, \frac{D}{7d}, \frac{D}{c}, \frac{D}{3d}, \frac{D}{f}, \frac{D}{4d}, \frac{D}{f}, \frac{D}{4d}\right] \tag{9.15}$$

where:

$$D = \frac{14D}{27s + 27e + 27d} \tag{9.16}$$

DDACap Protocol

The state of transition and labels, detailed labels, matrix in Appendices K, H.5, I.4 respectively, the cost and transition association metrics in Tables 9.4 and 9.6. Consider the transition system with the following generator matrix $Q5$ = CTMC (DDACap) and the stationary distributions Π_5, where $E = 26s + 23e + 23d$.

$$\Pi_5 = \left[\frac{E}{3s}, \frac{E}{h}, \frac{E}{6d}, \frac{E}{h}, \frac{E}{6d}, \frac{E}{c}, \frac{E}{3d}, \frac{E}{f}, \frac{E}{4d}\right] \tag{9.17}$$

where:

$$E = \frac{9E}{22s + 19e + 19d} \tag{9.18}$$

9.2.8 Analytical Model Performance Results

Efficiency

The relative efficiency of each protocol, in terms of their utilization of cryptographic procedures are defined in section 9.2.6 *definition 5 and 6* with equations (9.6) and (9.7) respectively. It assigns value 1 (a non-zero transition reward) to any transition in which the decryption is enabled and assigns value 0 to any other transition. Value 1 is assigned to the following:

1. the 2nd, 4th, 6th, 8th, 10th, 12th, 14th, 16th transitions in SAP-AKA
2. the 2nd, 4th, 6th, 8th, 10th, 12th, 14th, 16th, 18th, 20th transitions in NS-FId
3. the 2nd, 4th, 6th, 8th, 10th, 12th, 14th, 16th transitions in DCSS
4. the 2nd, 4th, 6th, 8th, 10th, 12th, 14th transitions in DDSec
5. the 3rd, 5th, 7th, 9th transitions in DDACap

Using the performance measure R, the performance of the protocols is as follows:

$$R(SAP-AKA) = \frac{A}{8d} \tag{9.19}$$

$$R(NS-FId) = \frac{B}{10d} \tag{9.20}$$

$$R(DCSS) = \frac{C}{8d} \tag{9.21}$$

$$R(DDSec) = \frac{D}{7d} \tag{9.22}$$

$$R(DDACap) = \frac{E}{4d} \tag{9.23}$$

It is possible to prove that one protocol is more costly than the other based on s, d and e and depending on encryption scheme, since same quantitative measure were used for performance evaluation. It can also be measured and compared to the performance of different versions of the same protocol for efficiency.

Protocol Throughput

The throughput is the result of associating a transition reward to a rate and a transition of an activity. Since a transition is run once in a system, CTMC is cyclic and each transaction is represented by a label then throughput of all transactions is the same. The last transaction is chosen to compute the throughput of the protocol by associating a transition reward to be equal to the rate with the last protocol communication and then give zero transition reward to all the other communications. It is assumed encryption and decryption have the same cost, while point multiplication consumes more time than decryption. It is also known as stated in (Hodjat & Verbauwhede 2002), that time complexity of cryptographic algorithm is related to its energy consumption as the results indicate.

The reward structure and total rewards discussed in section 9.2.6 are computed as follows:

$$\rho1 = (0......C_{16}), \quad (C_{16}) = 3d \quad R(SAP - AKA) = \frac{3d}{20s + 19e + 19d} \tag{9.24}$$

$$\rho2 = (0......C_{20}), \quad (C_{20}) = 2d, \quad R(NS - FId) = \frac{2d}{28s + 28e + 28d} \tag{9.25}$$

$$\rho3 = (0......C_{16}), \quad (C_{16}) = 2d, \quad R(DCSS) = \frac{2d}{20s + 20e + 20d} \tag{9.26}$$

$$\rho4 = (0......C_{14}), \quad (C_{14}) = 4d, \quad R(DDSec) = \frac{4d}{27s + 27e + 27d} \tag{9.27}$$

$$\rho5 = (0......C_{11}), \quad (C_{9}) = 4d, \quad R(DDACap) = \frac{4d}{22s + 19e + 19d} \tag{9.28}$$

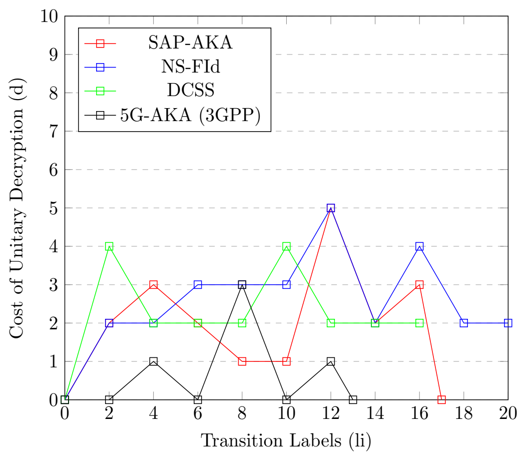
9.2.9 Performance Analysis

There are many studies on 5G security that proposed security protocols, but few evaluated the protocols' performance. Generally, performance evaluation is based on communication and computational overheads, but also quantitative measurements from mathematical models can be used. The authors in (Haddad et al. 2020) proposed a secure authentication and key agreement protocol for 5G based on blockchain. They also evaluated the performance of the protocol in relation to communication and computational overheads, with results indicating that it is more efficient than the current schemes. In (Gupta et al. 2018), the authors present a generic construction for the efficient and secure AKA protocol in a 5G network, with the performance evaluation showing less communication and computational overhead. In (Seok et al. 2020), the author proposed a secure D2D communication system based on ECC and lightweight authenticated encryption with associated data (AEAD) ciphers to cover resource-constrained IoT devices.

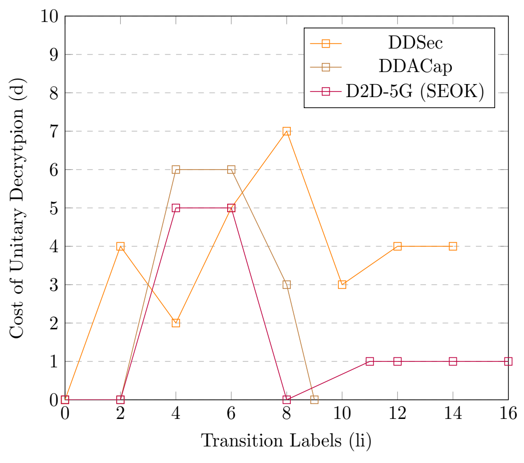
To evaluate our proposed security protocols based on analytical modelling, the cost metrics in Tables 9.4 and 9.5 defined in subsection 9.2.3 are used as inputs to measure output (s), encryption (e), decryption (d) cost. This study uses the 5G-AKA protocol in chapter 6 as a benchmark for

Table 9.7: Protocols Performance Evaluation

Protocols	**Efficiency**	**Throughput**
5G Protocols		
3GPP-5G-AKA (3GPP 2020*f*)	$\frac{3GPPP-5G-AKA}{3d}$	$\frac{d}{21s+5e+5d}$
SAP-AKA	$\frac{A}{8d}$	$\frac{3d}{20s+19e+19d}$
NS-FId	$\frac{B}{10d}$	$\frac{2d}{28s+28e+28d}$
DCSS	$\frac{C}{8d}$	$\frac{2d}{20s+20e+20d}$
5G D2D Protocols		
D2D-5G-Seok (Seok et al. 2020)	$\frac{D2D-5G-Seok}{5d}$	$\frac{d}{15s+3e+3d}$
DDSec	$\frac{D}{7d}$	$\frac{4d}{27s+27e+27d}$
DDAcap	$\frac{E}{4d}$	$\frac{4d}{22s+19e+19d}$



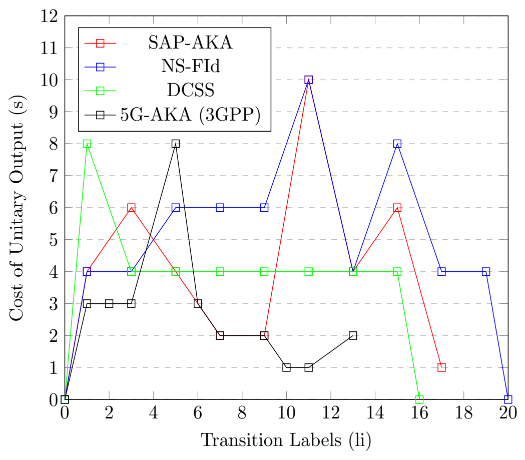
(a) SLS Effeciency



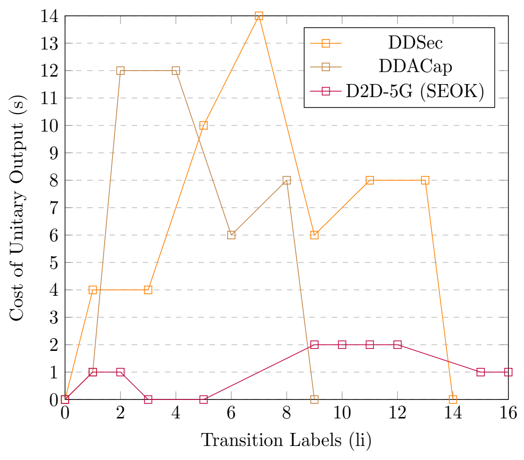
(b) DDS Effeciency

Figure 9.3: Efficiency Comparison of the Proposed Protocols based on Analytical Modelling

protocols proposed in chapter 7 since it is 3GPP's specified protocol to address NAC, and its cost is shown in Table 9.5 with stationary distribution as $3GPP - 5G - AKA = 20s + 13e + 13d$. The 5G-AKA synchronization and re-authentication phases are excluded from all performance evaluation, since not all protocols are modelled with those two phases. The results in Table 9.7 and Fig. 9.3 indicate that 3GPP-5G-AKA has similar performance costs in terms of efficiency and throughput as SAP-AKA, NS-FId and DCSS. However, the little spikes on SLS protocols performance are due to the additional protection added such as non-repudiation and SSO, as 5G-AKA cannot be used



(a) SLS Throughput



(b) DDS Throughput

Figure 9.4: Throughput Comparison of the Proposed Protocols based on Analytical Modelling

outside the HN or with third-party networks. This study also compares the D2D security protocols in chapter 8 with the D2D-5G-Seok protocol (Seok et al. 2020) and its cost, shown in Table 9.5 with the stationary distribution as $D2D - 5G - Seok = 25s + 20e + 20d$. The results indicate that D2D-5G-Seok as efficient as DDSec and DDACap protocols, as shown in Table 9.7 and Fig. 9.4. However, graph 9.4 shows that it has better performance on throughput but this is due to the exclusion of 5G-AKA cost from the total sum of the D2D-5G-Seok protocol performance cost. It uses 5G-AKA to authenticate the UEs and generate security tokens. Moreover, it also uses some of its security context like SUCI, but our proposed protocols do not, as that would increase the attack vector on our integrated security solution, and it is against 5G security standard specifications 3GPP (2020*f*).

The performance evaluation of the proposed solution was achieved using an enhanced operational semantics with bisimilarity labelled transitions and associated rates. The general distribution of an activity was based on enhanced labels (Priami 1996), deriving CMTC associated with transition system of a process, and assigning rates to transitions. Using its stationary distribution, the performance of the process was evaluated using continuous time approach. The study adopted the same approach as (Bodei et al. 2005*b*) but relied on ProVerif and applied *pi* calculus for security properties and bisimilarity labelling, respectively, to conduct security behavioural and quantitative analysis. The analytical modelling used numerical results to illustrate the effectiveness of the model and system linear equation to calculate some results based on some assumptions.

Additionally, the system model architecture information and the cryptographic schemes enabled the acquisition of actual values to evaluate the cryptography primitives and schemes of the proposed protocols. Each cryptographic scheme has a different cost based on its resources and time consumption. Moreover, the algorithm behaviour and cost can be influenced by the cryptographic scheme and the protocol design and vice versa, such as in symmetric and asymmetric cryptography.

Table 9.8: Approximate Time for Cryptographic Operations

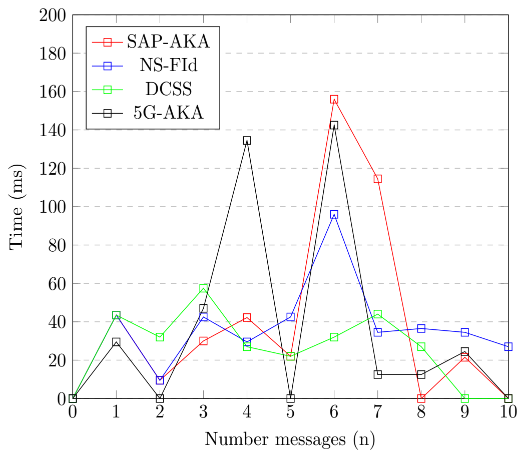
Notation	**Description (time to compute)**	**Rough computation time (ms)**
T_{Av}	authentication vectors	33.5
T_h	hash function	5
T_{Se}	symmetric encryption	4
T_{Sd}	symmetric decryption	5.5
T_{Ae}	asymmetric encryption	8
T_{Ad}	asymmetric decryption	9.5
T_{Tn}	token	5
T_{Ts}	timestamp	5
T_{KDF}	NAC/SL key	12.0
T_{D2D}	D2D key	20.0
T_E	execute	21.5
T_V	verify	12.5

9.3 Performance Evaluation based on a Simulation Model Approach

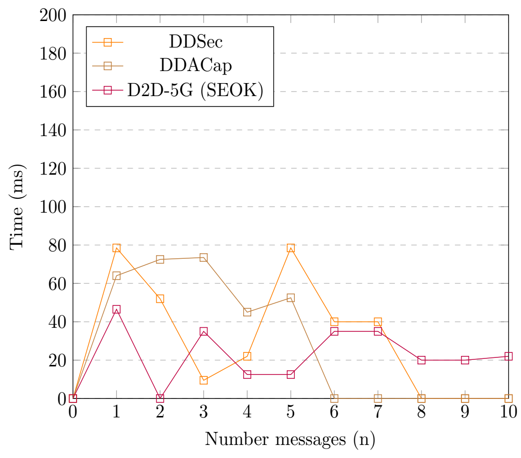
This section evaluates the performance of proposed protocols in chapters 6, 7 and 8 and intends to measure the network impact of these protocols based on NS-3 (3.33) simulation model. As presented in section 3.3.9, the simulation model was built using C++ programming language based on NS-3 5G and LTE modules (Nsnam 2021), (Mezzavilla et al. 2018) and (Banerjee et al. 2020). As real representation of 5G non-standalone implementation, whereby 5G is being deployed with 5G radio technology and LTE as the core network. The NS-3 simulation is made up of different modules that are used to programme and run a successful simulation. In order to simulate 5G network communication, the nodes, net device and topology helpers modules were modified to represent communication between 5G protocols. In addition, the computational and communication costs of the protocols are evaluated with the assumption that all protocols are using 5G cryptographic primitives and algorithms recommended by 3GPP (3GPP 2020*f*).

9.3.1 Computational and Communication Cost

The evaluation of the computational and communication costs of the proposed security protocols follows a similar method as in section 9.2.7, the SLS protocols are compared with 3GPP's specified 5G-AKA protocol since it is the recommended network protocol for 5G HN. The DDS protocols are compared with D2D-5G-SEOK protocol as it was also developed for D2D in 5G.



(a) SLS Computational Cost



(b) DDS Computational Cost

Figure 9.5: Comparison of Computational for SLS and DDS Protocols based on Simulation Modelling

Table 9.9: Computational Cost of the Proposed Protocol

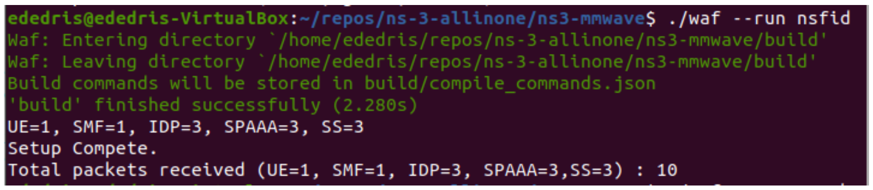
Protocols	**Computational Time (ms)**	**Total Time**
	(ms)	**(ms)**
5G Protocols		
SAP-AKA	$T_E + 6T_{Se} + T_{Ae} + 6T_{Sd} + T_{Ad} + 1T_{Av} + 13T_{KDF} + 6T_V$	439.2
NS-FId	$T_E + 6T_{Se} + 4T_{Ae} + 6T_{Sd} + 4T_{Ad} + 1T_{Av} + 2T_{KDF} + 5T_h + 2T_n + 10T_V$	396
DCSS	$T_E + 8T_{Se} + 8T_{Sd} + 4T_h + 2T_n + 10T_V$	285
5G-AKA (3GPP 2020*f*)	$T_E + 6T_{Se} + T_{Ae} + 6T_{Sd} + T_{Ad} + 1T_{Av} + 8T_{KDF} + 2T_h + 11T_V$	545.5
5G D2D Protocols		
DDSec	$T_E + 4T_{Se} + 3T_{Ae} + 4T_{Sd} + 3T_{Ad} + 2T_{K_{D2D}} + 7T_h + 1T_n + 2T_s + 7T_V$	360.5
DDACap	$T_E + 5T_{Ae} + 5T_{Ad} + 2T_{K_{D2D}} + 7T_h + 1T_s + 8T_V$	307.5
D2D-5G-Seok (Seok et al. 2020)	$(T_E + 7T_{Se} + 2T_{Ae} + 7T_{Sd} + 2T_{Ad} + 2T_{K_{D2D}} + 8T_{KDF} + 2T_n + 2T_h + 6T_V)$ + (5G-AKA)	238.5 + 545.5 = 784

Computational Cost

The time cost of security vectors and primitives generation are defined as T_{AV}, T_h, T_{SE}, T_{SD}, T_{AE}, T_{AD}, T_{Tn}, T_{Ts}, T_{KDF}, and $T_{K_{D2D}}$ as shown in Table 9.8. Moreover, these are the estimated times in milliseconds (ms) needed for computing the respective cryptographic primitives and messages. The total computational cost of each protocol is summarised in Table 9.9 and performance comparison is shown in Fig.9.5.

Communication Cost

The protocol cryptographic primitive, scheme and the message used as parameters with values as shown in Tables 9.10 and 9.11, which are used to define the cost of a protocol. Security context used such as AMF, $synch\_fail/mac\_fail$ $authzgrant$ code, $dataname$, and $success$ message in 5G-AKA, NS-FId and SAP-AKA, respectively are represents as strings. The message sent between entities is defined as m and the n is the total sum of m in a protocol, $n = (m1, m2, m3, m4....)$ measured in bits. However the value of n may vary depending on the number of message and



```
ededris@ededris-VirtualBox:~/repos/ns-3-allinone/ns3-mmwave$ ./waf --run nsfid
Waf: Entering directory `/home/ededris/repos/ns-3-allinone/ns3-mmwave/build'
Waf: Leaving directory `/home/ededris/repos/ns-3-allinone/ns3-mmwave/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (2.280s)
UE=1, SMF=1, IDP=3, SPAAA=3, SS=3
Setup Compete.
Total packets received (UE=1, SMF=1, IDP=3, SPAAA=3,SS=3) : 10
```

Figure 9.6: NS-FId simulation results

Table 9.10: Cryptographic primitive size

Primitive	**Value**
Symmetric key	128 bits
Asymmetric key	256 bits
SHA256	256 bits
Token	128 bits
Nonce	128 bits
5G IDs	64 bits
D2D IDs	256 bits
Nonce key	256 bits
D1	256 bits
Strings	32 bits
MAC	64 bits
SQN	48 bits
Timestamp	16 bits
RES	256 bits

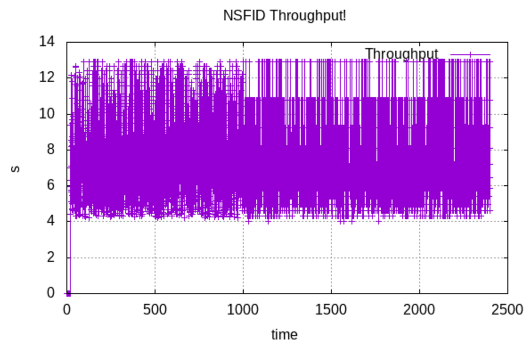
Table 9.11: Evaluation Metrics

Parameters	**Values**
Throughput	bits/ms
Latency	ms
m	messages primitive cost
n	total sum of m

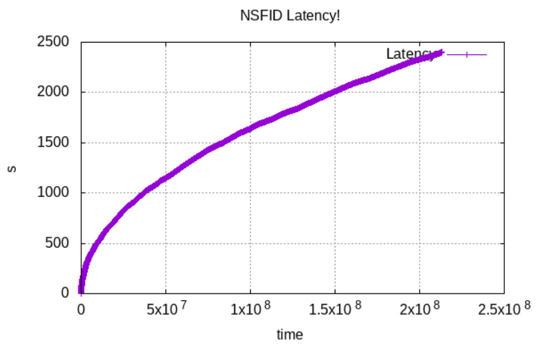
the primitives used. n is used to get the communication cost of the protocols by measuring the throughput (bits/ms) and latency (ms) as performance metrics during the protocol simulation in NS-3. The total communication costs of the proposed protocols are summarised in Table 9.12.

9.3.2 Simulation Performance Model Results

To run the simulation, `./waf --run scratch/sapaka` command is used on the terminal and `Mobileusernode` is defined as the `UdpEchoClient` pointing to the `serveNode` as the `UdpEchoServer` installed on the node to run the simulation successfully. Based on the analysis of the trace pcap, and XML files generated in NS-3, measurement results of our proposed protocols are obtained. The

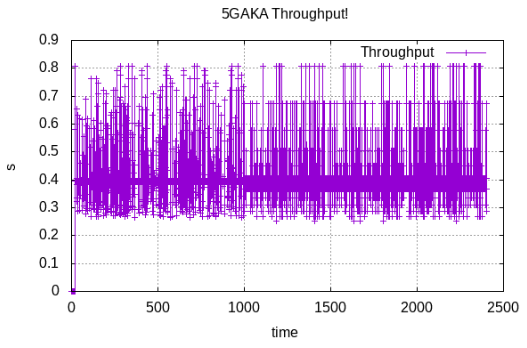


(a) Throughput

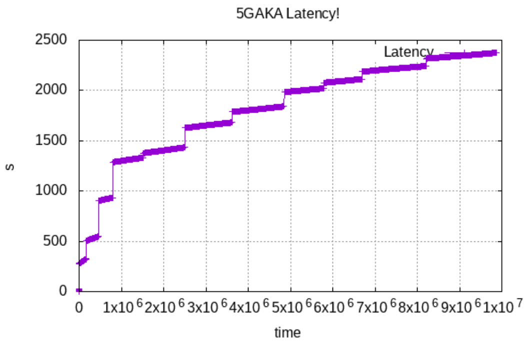


(b) Latency

Figure 9.7: Communication Cost for NS-FId Protocol



(a) Throughput



(b) Latency

Figure 9.8: Communication Cost for 5G-AKA Protocol

values and metrics in Tables 9.10 and 9.11 are used as inputs for NS-3 simulation. Figs. 9.6 shows a successful simulation run and 9.7 shows throughput/latency results for NS-FId protocol graphically. Other protocols' results are in Appendices M and N. The simulation results also illustrate the exchange of packets of data (protocol messages) between the nodes (protocol entities) and the relevant packets received by the devices.

9.3.3 Performance Analysis

For simulation analysis, similarly, the 3GPP's 5G-AKA protocol (3GPP 2020*f*) is used as a benchmark for evaluating the proposed SLS protocols and D2D-5G-Seok protocol (Seok et al. 2020) for proposed DDS security protocols. The computational cost of the proposed protocols is summarised in Table 9.9, with Fig. 9.5 (a) indicating that the proposed SLS protocols have a lower computational cost than the 5G-AKA protocol. In addition, DDS protocols also have lower computational costs compared to the D2D-5G-SEOK protocol, due to D2D-5G-Seok using the 5G-AKA protocol for authentication and token generation, as shown in Fig. 9.5 (b).

Table 9.12: Communicational Cost of the Proposed Protocol based on Simulation Modelling

Protocols	**Total Communication Cost (bits)** (n)	**Number of messages** (m)
5G Protocols		
SAP-AKA	3136	9
NS-FId	5472	10
DCSS	4096	8
3GPP 5G-AKA (3GPP 2020*f*)	5898	10
5G D2D Protocols		
DDSec	7904	7
DDACap	5760	5
D2D-5G-Seok (Seok et al. 2020)	2016 + 5898 (5G-AKA) = 7914	9 + 10 = 19

The communications cost of the proposed protocols are summarised in Table 9.12, the plot graphs for throughput and latency generated directly from NS-3 simulation are in Appendix N. The results indicate that the proposed SLS protocols have lower communication costs than 5G-AKA. It is interesting to note that both NS-FId and 5G-AKA have 10 m but have 5472 bits and 5898 bits n, receptively. Figs. 9.7 and 9.8 show variation in their NS-3 output. For DDS protocols, they have lower communication costs than the D2D-5G-Seok protocol(Seok et al. 2020), due to its reliance on the 5G-AKA protocol. Hence, increasing its communication cost n to 7914 bits and messages m to 19, as shown in Table 9.12. Therefore, these approaches use cost factors to influence the design and efficiency of a protocol for a particular security solution. Both analytical and simulation modelling analysis show that our proposed SLS protocols have similar performances to 5G-AKA protocol and our DDS protocols have better performance than D2D-5G-Seok protocol.

The next section discusses the integration of proposed solutions in this research into a unified solution and how these protocols can be implemented.

9.4 Security Protocols Integration

This section presents the integration of proposed protocols. As discussed in chapter 5, the proposed NSS framework addresses security for NSD in 5G enabled D2D communications from when the UE requests access to the network via the wireless access to when it is authorized to share the service with another UE. The purpose of the NSS framework is to protect the entities that are involved in the communication and the data being shared on the communication channels in different security domains and scenarios from threats discussed in chapter 4. The solution consists of three levels NAC, SLS and DDS which were introduced in chapter 5, they are as follows:

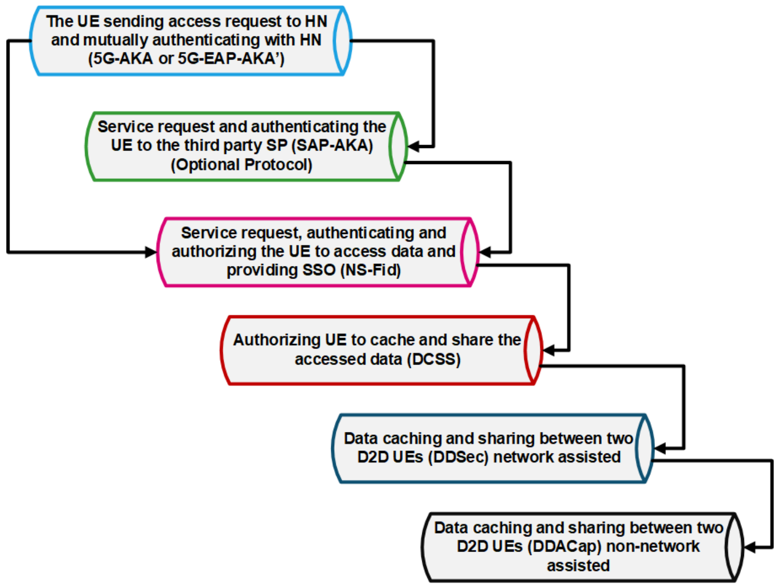

The UE sending access request to HN and mutually authenticating with HN (5G-AKA or 5G-EAP-AKA')
Service request and authenticating the UE to the third party SP (SAP-AKA) (Optional Protocol)
Service request, authenticating and authorizing the UE to access data and providing SSO (NS-Fid)
Authorizing UE to cache and share the accessed data (DCSS)
Data caching and sharing between two D2D UEs (DDSec) network assisted
Data caching and sharing between two D2D UEs (DDACap) non-network assisted

Figure 9.9: Security Protocols Interfaces

- NAC is concerned with 5G network access security and it provides primary authentication. It protects the data on wireless connection, the entities and the communication between UE, SN and HN.
- SLS is concerned with service authorization of UE, it provides secondary authentication and authorization. It protects data, entities, and communication between the UE, HN and SP in different domains.
- DDS is concerned with D2D communication security, it provides authentication and authorization between two UEs in proximity of each other, it enables data sharing both in networks assisted and non-network assisted communication. It protects the data, entities, and the communication between two UEs and network.

The next section discusses how the protocols presented in this thesis are applied in their respective levels and how the protocols interface each other to provide a continuous security for the UE at all levels of mobile network communications.

9.4.1 Security Model

This subsection recalls the security model reflecting on network, service and D2D security levels as introduced in chapter 5. The underlying security protocols of the security framework address security on three levels of communications i.e., network, service and D2D communications. These security levels interconnect with the protocols interfaces and the protocols are encapsulated while addressing the security requirements from one level to another through protocol interfacing as shown in Fig. 9.9.

The security model levels each consists of the following security protocols,
NAC:

1. 5G-AKA achieves primary AKA.
2. EAP-AKA' achieves primary AKA.

SLS:

1. SAP-AKA achieves the secondary authentication by providing authentication and session key.
2. NS-FId achieves federated authentication, authorization and SSO.
3. DCSS achieves data caching and sharing authorization.

DDS:

1. DDSec achieves authentication, authorization and federated delegation with network assistance.
2. DDACap achieves authentication, authorization and federated delegation without network assistance.

9.4.2 Connection of Different Levels of NSS Model in 5G

The user via their UE starts by requesting network access, the closest SEAF will initiate primary authentication. The process for the UE to access the network and services is described in chapter 4. The security model is intended to establish three secure connections between the UE and HN, UE and SP as well as UE and UE in different stages as described below:

- The first stage, the UE sends network access request whereby a primary authentication protocol invokes choosing either 5G-AKA or EAP-AKA' protocol described in chapter 6 to facilitate AKA procedure between UE and HN.
- The second stage, after a successful primary authentication the UE sends a service request that triggers either a secondary authentication or authorization procedure, which is handled by SMF in HN and SPAAA in the SP network. At this stage SAP-AKA or NS-FId protocols can be chosen depending on registration status and security policies described in chapter 7.
- The third stage, after being granted access to the services, the UE sends another request for data caching and sharing authorization which invokes DCSS protocol also described in chapter 7.
- The fourth stage, the UE authorized to cache and share the data then can publish the data by broadcasting the data name to other UEs in proximity. An interested UE sends its data interest to another UE which invokes the DDSec or DDACap protocols as discussed in chapter 8.

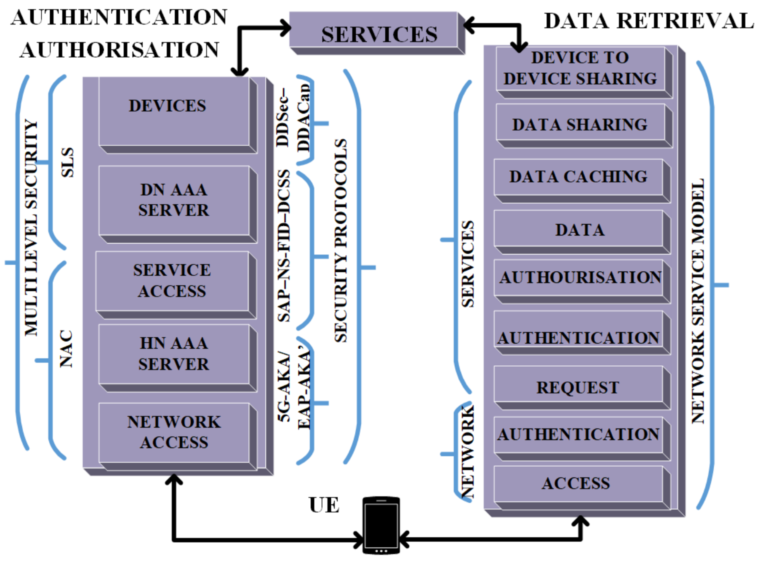


Figure 9.10: Integrated Security Solution

9.4.3 Federated Security in 5G

This subsection discusses how federated security is integrated in 5G and how the UE achieves SSO. FIdM in 5G was presented in chapter 7, it explained the advantages of using FId in mobile communications and how it eliminates the need for the UE from continuous re-authentication and re-authorization for services. In fact, the UE might need to re-authenticate to the network or perform handover authentication while roaming but due to SSO, tokens and caching data, such security processes are reduced when the proposed solutions in this research are implemented in 5G enabled D2D communications network.

Next, we demonstrate how federated security is invoked in 5G communication and it is as follows:

- Step 1: After UE is authenticated to the network, it requests service authorization to the SP.
- Step 2: The SP via SMF redirects the UE to IdP, which generates the FId and assigns it to UE.
- Step 3: The IdP and UE perform federated authentication procedure that assigns UE with a new ID and a token.
- Step 4: The UE use ID token to request access token from the SP. The SPAAA assigns the access token/refresh token and achieves SSO.
- Step 5: The UE uses access token to request access to service, which granted by the SS if the access token is valid.
- Step 6: After gaining access to the services, the UE requests data caching and sharing authorization with other UE, hence the generation of cache and share tokens.

The interface between the underlying security protocols of the security framework provide an integrated security solution addressing security threats at different levels of the system model as shown in Fig. 9.10.

9.5 Possible Implementation of the Proposed Protocols

Security protocols formal modelling and verifications have advanced that the development of protocols have reduced the attacks on developed protocols before being utilized. The implementation of security protocols in mobile networks are challenging and error prone despite the advance in verification methods as discussed in chapter 3. This is due to the complexity of direct integration between verification and implementation tools. Tools like ProVerif should be able to interact with any Integrated Development Environment (IDE) platform to prevent implementation caused vulnerabilities and incorrect software development (Garcia & Modesti 2017), (Blanchet 2016). The adoption of formal modelling directly in real world scenarios can be achieved by integrating formal modelling and software development methods supported by IDE. The use of formal methods automated tools together with compilers and code generators can improve the generation of codes using programming languages such as C and Java.

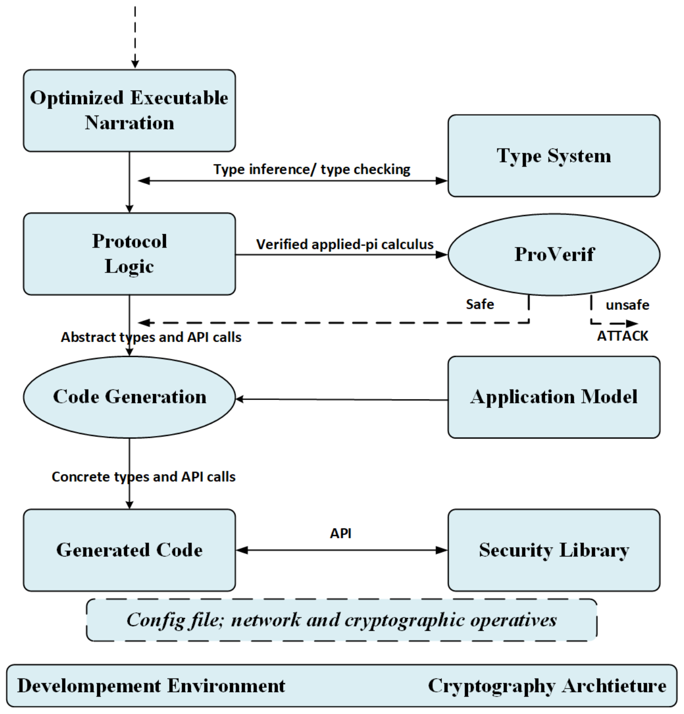

Optimized Executable Narration
Type System
Type inference/ type checking
Protocol Logic
Verified applied-pi calculus
ProVerif
Safe
unsafe
ATTACK
Abstract types and API calls
Code Generation
Application Model
Concrete types and API calls
Generated Code
API
Security Library
Config file; network and cryptographic operatives
Develompement Environment
Cryptography Archtieture

Figure 9.11: ProVerif, Complier and Code Generator

9.5.1 Compiler and Code Generator

Different approaches have been suggested to integrate protocol modelling and implementation using the back-end and front-end tools (Garcia & Modesti 2017), (Aizatulin et al. 2011), (Sisto et al. 2018). To generate any code, the language uses an independent intermediate format for protocol logic to parametrize the translation and simplify the hard-core programming language as shown in Fig. 9.11. Type system refers to type expression, terms and variables for well-typed code and helps in the evaluation of incoming messages with the protocol specifications (Garcia & Modesti 2017). The Applicatoion Programing Interface (API) calls are bound to the protocol logic and the verification is achieved by translating protocol logic to applied *pi* calculus in ProVerif from which the implementable code is extracted. Proverif was developed in OCaml programming language with a byte-code compiler ocamlc and a native-code compiler ocamlopt for generating executable code. Therefore, C or Java functions can be called in OCaml code and verse versa (Leroy et al. 2019). This is achieved by deploying user-defined primitives in C or Java, linking it with OCaml code. However, the code generation, interpretation and protocol implementation are beyond the scope of this study.

9.5.2 Analysis

The threats and vulnerabilities in 5G enabled D2D communications have been discussed and possible solutions explored in chapter 4. To address these issues security model, framework and underlying security protocols were proposed in chapters 5, 6, 7, and 8. The security protocols were encapsulated to achieve a unified integrated, robust, and efficient security solution for NSD in 5G enabled D2D communications network at network, service and D2D levels. The underlying security protocols at each level of the system model are configured as an interface using cryptographic primitives and security context to create a refined security mechanism. These include the 3GPP specified protocols, however, the compromise of these protocol should not affect the security of the proposed security protocols. The proposed protocols have been simulated against the DY adversary, verified in ProVerif with the results in sections 6.3, 7.3, 7.8, 8.4.1, and 8.6.1 indicating the they can protect 5G enabled D2D communications from the defined threat adversary capabilities and attacks presented in sections 3.3.2 and 4.3, respectively. Attacks such as eavesdropping, data fabrication, control data, impersonation, free-riding, privacy violation, content poisoning,cache pollution, unauthorized access, cache misuse, false accusation, location spoofing, session hijacking, data leakage can be prevented. However attacks such as communication monitoring, jamming attack, IP spoofing and DoS might require the control measure at the BBU and edge network entities. The proposed solution in this thesis provide and multi-layered security for an integrated 5G system.

9.6 Summary

This chapter discussed the performance evaluation of the proposed security protocols based on analytical and simulation modelling. The analytical modelling relies on protocols specifications in applied *pi* calculus, enhanced semantic operations, and Markov chain model to quantify the security properties and ProVerif processes to measure the efficiency and throughput of the protocols. The simulation modelling uses NS-3 modules, parameters set using c++ to simulate the message exchange entities as predefined packets sizes, and estimation of computational time to measure protocols' computational time, latency, and throughput. The chapter also discussed how the security model and framework support the 5G communications and how the proposed protocols interface with each other using federated security to provide an integrated solution for 5G. Moreover, it secures the UE access to the network and services in 5G enabled D2D communications network, including non-3GPP access. The chapter also explores how the intuitive and functional programming languages can be integrated and supported by IDE platform to verify security protocols using formal methods, an automated protocol verifier, and automatically interpret and generate codes in C and Java for implementation. With all the mathematical and simulation analyses in this chapter considered, the proposed protocols in this research study have good performance compared with the similar protocols and as analysed in chapters 7 and 8, they are proved to be secure as an integrated security solution as discussed in this chapter.

Chapter 10

Conclusions and Future Work

10.1 Introduction

This chapter summarises the main ideas, results, achievements, and future work of the proposed novel concepts. It describes the main theme of this study and how the research questions were answered successfully. This chapter also lists the limitations of the thesis.

10.2 How were the key research questions answered?

The research identified crucial gaps in addressing the issues of providing security in 5G network. It disclosed how security is imperative in providing NSD in 5G enabled D2D communications network. It also revealed how D2D communication is going to play a momentous role as an under technology in delivering NS to the end users as well pushing the data traffic from the back backhaul to the fronthaul of 5G network. In addition, the research highlighted the need to address security on different levels of 5G network and in different scenarios. These issues were included in one main question **'How to provide secure communications for use cases in 5G enabled D2D communications network?'**, which led to four important research questions, and they are as follows:

'How to introduce an integrated NSD framework for the 5G enabled D2D communications network and what are the main supporting technologies that are required in this model?'
The answer to these questions, a networks services delivery framework was introduced in chapter 4. It defined the main entities required for NSD provisioning and its security. Furthermore, it defines the technologies that are used to support 5G functionality. These entities enable service access and delivery in 5G enabled D2D communications network based on different scenarios in HN and VN. 5G defines three main processes to access the services: Registration to MNO, authentication to the network, and authorization to the services. While D2D communication has the following process: Discovery, D2D link, and D2D communication. They are discussed in detail in chapters 2 and 4.

'What are the security vulnerabilities and threats in the NSD framework that the UE can be exposed to in 5G enabled D2D communications network?'
To answer this question as stated in chapter 4, the research provided a comprehensive investigation

into the security issues that affect UE in a 5G enabled D2D communications network. The X.805 was used to define the security threats and vulnerabilities, it was also used to evaluate the security requirements of the system model.

'What are the underlying protocols of the NSS framework and how these protocols are integrated together?'
To answer this question as stated in chapter 5, the research defined a security framework that specified the security model, levels, and the underlying protocols in chapters 6, 7 and 8. It also presented authentication and authorization procedures that provide authentication, key establishment, and authorization at network, service, and D2D-level in different domains such as HN, DN, and D2D communications. That is why the D2D-level of security was introduced in this research. All the security protocols were verified by using formal methods, ProVerif, and applied *pi* calculus. Furthermore, the security performance of the proposed protocols was evaluated with quantitative measures using labelled semantics and Markov chains as well as a network simulator NS-3 in chapter 9. Then the integration between these protocols was defined by authentication and federated authorization mechanisms.

How could the proposed security protocols interface into an integrated security mechanism to provide security for 5G enabled D2D communications network?'
The answer to this question was answered in chapter 9. After designing the security protocols for authentication and authorization at network access, service, and D2D levels, an interface was set at each level between protocols to accomplish an integrated comprehensive and robust security solution. This was defined in the security framework's authentication and federated authorization mechanisms.

10.3 Main Contributions

As presented in chapter 1, the main contributions of this research are as follows:

- A critical review of existing legacy systems, 5G and D2D communications has been presented. Furthermore, it reviewed the existing technologies that can support NSD in 5G enabled D2D communications network. Consequently, this review also found 5G lacked a robust security mechanism for NSD, and the security standards defined by 3GPP were under specified for some security guarantees provided by AKA procedures.
- Introduced abstraction model of NS in 5G enabled D2D communications network that maps the NS with mobile network architectures and 5G protocol stack.
- Introduced a delivery framework that defines the system entities and describes the service overview and content dissemination.
- Explored a threat model to identify, enumerate, and prioritize vulnerabilities with an adversary point of view.
- Performed a security analysis and evaluated the requirement using a systematic and comprehensive approach in a modular format based on X.805 security framework. It identified the threats and vulnerabilities that needed to be addressed.
- Proposed a hybrid security solution that protects the entities, the communication channels, and the content object from any form of attack.

- Proposed NSS framework with an abstract security model, the framework defined three security levels; NAC, SLS, and D2D to address the security threats in 5G enabled D2D communications network.
- Formally analysed the 3GPP specified protocols used for primary authentication procedure using ProVerif.
- Proposed a FIdM model that can be integrated with a 5G network to facilitate federated authentication and authorization providing SSO.
- Proposed security protocols that can be integrated with a multi-level functional system and are interoperable with underlying supporting platforms.
- Formalized and verified the proposed security protocols with formal methods, ProVerif and applied *pi* calculus.
- Evaluated the security properties of the proposed security protocols with two taxonomies.
- Evaluated the security performance of the proposed protocols using Markov chain stochastic process for quantitative measures of the protocols' actions and cryptographic operations as well as measuring the performance in NS-3 with predefined message packets.

10.4 Elaboration on the Main Contributions

10.4.1 Identification of crucial gaps in knowledge in the field of delivering Network Services and providing security for 5G enabled D2D Communications network

The research carried out a comprehensive literature survey of related works on 5G, D2D communications, and security. It highlighted the momentous flaws in the investigated approaches:

- The security of related works on 5G, D2D communications, and CCN have been addressed separately in various studies without considering one unified abstracted multilayered solution.
- The D2D communications in 5G ambiguity, leading to abstract solutions not reflecting into 5G's SBA and scenario-specific solutions that include the service and D2D-level security solutions.
- Not recognising the role of FIdM in 5G; this versatile feature should be considered in the security architecture design and development of the security mechanisms to standardize its design and implementation.
- Not noticing that the enablement of virtualization in 5G supported by cloud services backend, also enables content-aware services which allow the integration of CDN such as CCN in the network. Hence, it should be considered in the design of the next generation mobile architecture but also consider the security benefits and challenges of such architecture.

10.4.2 Defining Network Services and Abstraction Levels for 5G enabled D2D Communications network.

We have introduced a NS abstract in chapter 2, which elaborated more on NS in 5G and explained how the NS solution needs to be aligned with the 5G protocol stack for developing solutions for a specific level.

10.4.3 Defining the NSD framework for 5G enabled D2D Communications network to retrieve and share services

To propose a practical mechanism for addressing security requires a system model. Since there is no system model that supports NS integration as discussed in this research, a service delivery framework that integrates both D2D communication and CCN leveraging on C-RAN was introduced in chapter 4. It specified how the supporting technologies benefit each other to provide an effective NSD process. Also, it specified how the UE can access the services in different scenarios concurrent with the 5G objectives.

10.4.4 Defining the security threats and vulnerabilities in an integrated system model for 5G enabled D2D Communications network

After defining the system model that defines access and delivery of services in HN and SP environment, to provide a secure system, a comprehensive security analysis has been provided in chapter 4 using X.805 security framework that evaluated the security and privacy requirements of the proposed system model. It highlighted on the security issues that are faced by an integrated system model. The results indicated that there was a need to address the following security requirements; access control, authentication, non-repudiation, data confidentiality, communication security, data integrity, availability, and privacy in a modular form focussing on infrastructure and services layers.

10.4.5 Defining a security framework for an integrated system model in Next Generation mobile network.

Having conducted a security analysis, and we identified threats and vulnerabilities that need to be addressed, a security framework was proposed in chapter 5, it defined three security levels, which are NAC, SLS, and DDS in chapters 6, 7, and 8 respectively. The framework addressed the security threats and vulnerabilities in 5G enabled D2D Communications network at different levels. First, we had to ensure that the 3GPP AKA framework discussed in chapter 2 provided the security guarantees as specified as it was going to be used as the security foundation of the proposed security solution. So, this research formally analysed and verified the specified security protocols; 5G-AKA and 5G EAP-AKA' protocols in chapter 6 to evaluate the security guarantees provided by the primary authentication as specified by the security standard TS 33.501. The security framework consisted of the following levels:

1. NAC: Two protocols were introduced under the 3GPP AKA framework to provide primary authentication between UE and HN at the network level.
2. SLS: Three protocols were proposed to provide security between the UE and SP. The first protocol provides secondary authentication, while the second protocol provides the federated

authentication, authorization, and SSO to the UE granting access to services. While the third protocol provides data caching and sharing authorization to the UE.

3. DDS: Two protocols were proposed to provide security in two different D2D communications scenarios.

10.4.6 Providing security for 5G enabled D2D Communications network.

Having introduced the security framework that defined different security levels, security was provided by the proposed underlying security protocols. The security protocols were formally verified, analysed using ProVerif and *pi* calculus. ProVerif was chosen as it is still supported and a widely used formal verification tool for many security protocols as stated in (Blanchet 2016). These include two protocols specified by 3GPP and five protocols developed as a result of this research and they are as follows:

1. 5G-AKA in chapter 6 is based on EPS AKA and defined by 3GPP, it provides authentication between UE and SN to HN and session for UE and SN.

2. 5G EAP-AKA' in chapter 6 is based on the EAP framework also defined by 3GPP, it provides authentication between UE and SN to HN and session key for UE and SN.

3. SAP-AKA in chapter 7 provides secondary authentication and session key between HN and SP in 5G communications.

4. NS-Fid protocol in chapter 7 provides federated authentication, authorization, and SSO to a UE from 5G HN to SP services.

5. DCSS protocol in chapter 7 provides data caching and sharing authorization in 5G communications

6. DDSec protocol in chapter 8 provides authentication, data caching and sharing authorization in a network-assisted D2D communication.

7. DDACap protocol in chapter 8 provides authentication, data caching, and sharing authorization in a non-network assisted D2D communication.

10.5 Future Improvements to the proposed solutions that can benefit this study

A set of improvements to the proposed solutions are as follows:

- Analysing the security protocols performances using simulation, emulation tools, and testbed implementations.

- Integrating the security protocols from OCaml to C or Java so that it can be implemented within D2D communication, designing, simulation, and implementation for operation applicability.

10.6 Limitations of the Research

5G is still early stages of deployment but not fully standardized and D2D communication has been considered in many applications yet and it was never standardized for any previous mobile network generations, even though the legacy system specified some ProSe functionalities. Moreover, related work discussed the security issues in D2D communications, but few discuss service authorization. This thesis is limited by some other practical issues, such as the availability of simulation tools and 5G testbeds. For example, it was not possible to perform an E2E simulation of D2D communications in 5G stand-alone network as the current simulation tools only offer 5G modules with mmWave as radio technology and LTE as core network. Also, the implementation was not possible due lack of 5G testbeds.

10.7 Future Works that can be pursued based on this study

Crucial security mechanisms have been discovered as the result of the proposed solutions in this study, improving the user's QoE, QoS and providing secure communication for 5G enabled D2D communications network. New opportunities have been produced because of the successful development of these new mechanisms, improving research interest in different areas of network services, 5G, D2D communications, and security management. Future work will also include the implementation of these protocols using a 5G testbed, allowing us to check for any new attacks as a result of implementation. Furthermore, the open issues will motivate future research trends including security on SDN/NFV, cryptographic protection below layer 2 in 5G network and integrated systems as well as federated-based network slices security.

10.8 Concluding Remarks

This thesis has addressed the crucial concerns of providing security for NSD in 5G enabled D2D communications network. With this contribution, I hope the future development of next generation mobile networks, will benefit from this contribution.

Bibliography

3GPP (2010), Feasibility study on the security aspects of remote provisioning, change of subscription for machine to machine (m2m) equipment, Technical specification (TS) 3GPP TR 33.812 V9.2.0 (2010-06), Third Generation Partnership Project. **URL:** *https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2307. (Accessed 10 April 2018)*

3GPP (2014), Study on architecture enhancements to support proximity-based services (prose), Technical specification (TS) 3GPP TR 23.703 V12.0.0 (2014-02), Third Generation Partnership Project. **URL:** *https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=864. (Accessed 08 June 2018)*

3GPP (2016), Study on architecture for next generation system, Technical specification (TS) 3GPP TR 23.799 V14.0.0 (2016-12), Third Generation Partnership Project. **URL:** *https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3008. (Accessed 16 April 2018)*

3GPP (2017), Study on the security aspects of the next generation system, Technical specification (TS) 3GPP TR 33.899 V1.3.0 (2017-08), Third Generation Partnership Project. **URL:** *https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3045. (Accessed 14 April 2018)*

3GPP (2020*a*), 3g security; security architecture, Technical specification (TS) 3GPP TS 33.102 V16.0.0(2020-07), Third Generation Partnership Project. **URL:** *https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2262. (Accessed 05 August 2020)*

3GPP (2020*b*), 3gpp system architecture evolution (sae); security architecture, Technical specification (TS) 3GPP TS 33.401 V16.3.0 (2020-07), Third Generation Partnership Project. **URL:** *https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2296. (Accessed 05 August 2020)*

3GPP (2020*c*), 3gpp system architecture evolution (sae) system aspects, security aspects of non-3gpp accesses, Technical specification (TS) 3GPP TS 33.402 V16.0.0(2020-07), Third Generation Partnership Project. **URL:** *https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2297. (Accessed 26 August 2020)*

3GPP (2020*d*), 5g system; technical realization of service based architecture, Technical specification (TS) 3GPP TS 29.500 V17.1.0 (2020-12), Third Generation Partnership Project. **URL:** *https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3338. (Accessed 08 January 2021)*

3GPP (2020*e*), Generic bootstrapping architecture (gba), Technical specification (TS) 3GPP TS 33.220 V17.0.0 (2020-12), Third Generation Partnership Project. **URL:** *https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2280. (Accessed 08 January 2021)*

3GPP (2020*f*), Security architecture; procedures for 5g system, Technical specification (TS) 3GPP TS 33.501 V17.0.0 (2020-12), Third Generation Partnership Project. **URL:** *https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3169. (Accessed 05 January 2021)*

3GPP (2020*g*), System architecture for the 5g system, Technical specification (TS) 3GPP TS 23.501 V16.7.0 (2020-12), Third Generation Partnership Project. **URL:** *https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3144. (Accessed 02 January 2021)*

5GPPP (2017), Deliverable d2.7 security architecture (final), Technical report, 5G Enablers for Network. **URL:** *https://5gensure.eu/sites/default/files/5G-ENSURE_D2.7_SecurityArchitectureFinal.pdf. (Accessed 20 August 2020)*

Abadi, M., Blanchet, B. & Fournet, C. (2017), 'The applied pi calculus: Mobile values, new names, and secure communication', *Journal of the ACM (JACM)* **65**(1), 1–41. ID: hal_soai_HAL_hal_01423924v1.

Abd-Elrahman, E., Ibn-khedher, H. & Afifi, H. (2015), D2d group communications security, *in* '2015 International Conference on Protocol Engineering (ICPE) and International Conference on New Technologies of Distributed Systems (NTDS)', IEEE, pp. 1–6.

AbdAllah, E. G., Hassanein, H. S. & Zulkernine, M. (2015), 'A survey of security attacks in information-centric networking', *IEEE Communications Surveys & Tutorials* **17**(3), 1441–1454.

Agiwal, M., Roy, A. & Saxena, N. (2016), 'Next generation 5g wireless networks: A comprehensive survey', *IEEE Communications Surveys & Tutorials* **18**(3), 1617–1655. ID: ieee_s7414384.

Ahlgren, B., Dannewitz, C., Imbrenda, C., Kutscher, D. & Ohlman, B. (2012), 'A survey of information-centric networking', *IEEE Communications Magazine* **50**(7), 26–36.

Aiash, M. & Loo, J. (2015), A formally verified access control mechanism for information centric networks, *in* '2015 12th International Joint Conference on e-Business and Telecommunications (ICETE)', Vol. 4, IEEE, Colmar, France, pp. 377–383.

Aiash, M., Mapp, G., Lasebae, A. & Loo, J. (2014), A secure framework for communications in heterogeneous networks, *in* '2014 28th International Conference on Advanced Information Networking and Applications Workshops', IEEE, pp. 841–846.

Aizatulin, M., Gordon, A. D. & Jürjens, J. (2011), Extracting and verifying cryptographic models from c protocol code by symbolic execution, *in* 'Proceedings of the 18th ACM conference on Computer and communications security', pp. 331–340.

Alliance, N. (2016), '5g security recommendations package #2: Network slicing', *White paper* pp. 1–12.

Altmann, V., Skodzik, J., Danielis, P., Mueller, J., Golatowski, F. & Timmermann, D. (2014), A dht-based scalable approach for device and service discovery, *in* '- 2014 12th IEEE International Conference on Embedded and Ubiquitous Computing', pp. 97–103. ID: 1.

Andersen, D. G., Balakrishnan, H., Feamster, N., Koponen, T., Moon, D. & Shenker, S. (2007), Holding the internet accountable, *in* 'HotNets', Citeseer. **URL:** *http://repository.cmu.edu/compsci/66. (Accessed 16 January 2021)*

Anderson, R. (2004), 'Cryptography and competition policy issues with "trusted computing"', *Computer Security Journal* **20**(1), 1–13.

Anggorojati, B., Mahalle, P. N., Prasad, N. R. & Prasad, R. (2012), Capability-based access control delegation model on the federated iot network, *in* 'The 15th International Symposium on Wireless Personal Multimedia Communications', IEEE, pp. 604–608.

Arkko, J., Eronen, P., Lehtovirta, V. & Torvinen, V. (2018), Improved extensible authentication protocol method for 3rd generation authentication and key agreement (eap-aka), Rfc, IETF. **URL:** *https://tools.ietf.org/html/draft-ietf-emu-rfc5448bis-03. (Accessed 10 December 2019)*

Arkko, J., Eronen, P., Lehtovirta, V. & Torvinen, V. (2019), Improved extensible authentication protocol method for 3gpp mobile network authentication and key agreement (eap-aka), Rfc, IETF. **URL:** *https://tools.ietf.org/html/draft-ietf-emu-rfc5448bis-05. (Accessed 10 December 2019)*

Arkko, J. & Haverinen, H. (2006), Extensible authentication protocol method for 3rd generation authentication and key agreement (eap-aka), Rfc, IETF. **URL:** *https://tools.ietf.org/html/rfc4187. (Accessed 10 December 2019)*

Arkko, J., Norrman, K., Näslund, M. & Sahlin, B. (2015), A usim compatible 5g aka protocol with perfect forward secrecy, *in* '2015 IEEE Trustcom/BigDataSE/ISPA', Vol. 1, pp. 1205–1209.

Arkko, J., Zorn, G., Fajardo, V. & Loughney, J. (2012), Diameter base protocol, Rfc, IETF. **URL:** *https://tools.ietf.org/html/rfc6733. (Accessed 10 January 2021)*

Armando, A., Basin, D. A., Boichut, Y., Chevalier, Y., Compagna, L., Cuellar, J. R., Drielsma, P. H., Heam, P. C., Kouchnarenko, O., Mantovani, J., Modersheim, S. A., Oheimb, D. V., Rusinowitch, M., Santiago, J., Turuani, M., Vigano, L. & Vigneron, L. (2005), 'The avispa tool for the automated validation of internet security protocols and applications', *Computer Aided Verification, Proceedings* **3576**, 281–285. ID: wos000230755800027.

Austin, K. D., Schoppert, B. J. & Almond, M. (2013), 'Identity broker configured to authenticate users to host services', Google Patents. US Patent 8,402,527.

Banerjee, S., Odelu, V., Das, A. K., Chattopadhyay, S. & Park, Y. (2020), 'An efficient, anonymous and robust authentication scheme for smart home environments', *Sensors* **20**(4), 1215.

Basin, D., Dreier, J., Hirschi, L., Radomirović, S., Sasse, R. & Stettler, V. (2018), A formal analysis of 5g authentication, *in* 'Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security', pp. 1383–1396.

Bertino, E. & Takahashi, K. (2010), *Identity management: Concepts, technologies, and systems*, Artech House, London, UK.

Blanchet, B. (2016), 'Modeling and verifying security protocols with the applied pi calculus and proverif', *Found. Trends Priv. Secur.* **1**(1-2), 1–135.

Blanchet, B., Smyth, B., Cheval, V. & Sylvestre, M. (2020), 'Proverif 2.01: automatic cryptographic protocol verifier, user manual and tutorial'. **URL:** *htps://prosecco.gforge.inria.fr/personal/bblanche/proverif/. (Accessed 05 June 2020)*

Bodei, C., Curti, M., Degano, P., Buchholtz, M., Nielson, F., Nielson, H. R. & Priami, C. (2005*a*), On evaluating the performance of security protocols, *in* 'International Conference on Parallel Computing Technologies', Springer, pp. 1–15.

Bodei, C., Curti, M., Degano, P., Buchholtz, M., Nielson, F., Nielson, H. R. & Priami, C. (2005*b*), 'Performance evaluation of security protocols specified in lysa', *Electronic Notes in Theoretical Computer Science* **112**, 167–189.

Bogale, T. E. & Le, L. B. (2016), 'Massive mimo and mmwave for 5g wireless hetnet: Potential benefits and challenges', *IEEE Vehicular Technology Magazine* **11**(1), 64–75.

Boneh, D. & Franklin, M. (2003), 'Identity-based encryption from the weil pairing', *SIAM journal on computing* **32**(3), 586–615.

Boyd, C. & Mao, W. (1994), On a limitation of ban logic, *in* 'Workshop on the Theory and Application of of Cryptographic Techniques', Springer, Springer-Verlag, Berlin, Heidelberg, p. 240–247. **URL:** *http://dl.acm.org/citation.cfm?id=188307.188350*

Burrows, M., Abadi, M. & Needham, R. M. (1989), A logic of authentication, *in* G. R. Andrews, ed., 'Proceedings of the Twelfth ACM Symposium on Operating System Principles, SOSP 1989, The Wigwam, Litchfield Park, Arizona, USA, December 3-6, 1989', ACM, pp. 1–13.

Chandrasekaran, G., Wang, N., Hassanpour, M., Xu, M. & Tafazolli, R. (2018), 'Mobility as a service (maas): A d2d-based information centric network architecture for edge-controlled content distribution', *IEEE Access* **6**, 2110–2129.

Checko, A., Christiansen, H., Yan, Y., Scolari, L., Kardaras, G., Berger, M. & Dittmann, L. (2015), 'Cloud ran for mobile networks-a technology overview', *IEEE Communications Surveys & Tutorials* **17**(1), 405–426.

Cisco (2017), Cisco visual networking index: Global mobile data traffic forecast update, 2016–2021 white paper, Technical report, Cisco. **URL:** *https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html. (Accessed 09 September 2018)*

Cohn-Gordon, K., Cremers, C. & Garratt, L. (2016), On post-compromise security, *in* '2016 IEEE 29th Computer Security Foundations Symposium (CSF)', Vol. 2016-, pp. 164–178. ieee_s7536374.

Crampton, J. & Khambhammettu, H. (2008), 'Delegation in role-based access control', *International Journal of Information Security* **7**(2), 123–136.

Dehnel-Wild, M. & Cremers, C. (2018), 'Security vulnerability in 5g-aka draft', *Department of Computer Science, University of Oxford, Tech. Rep* .

Dennis, J. B. & Horn, E. C. V. (1983), 'Programming semantics for multiprogrammed computations', *Communications of the ACM* **26**(1), 29–35.

Dick, H. (2012), The oauth 2.0 authorization framework, Rfc 6749, IETF. **URL:** *https://tools.ietf.org/html/rfc6749. (Accessed 10 December 2019)*

Dijk, M. V., Gentry, C., Halevi, S., Vaikuntanathan, V. & Gilbert, H. (2010), Fully homomorphic encryption over the integers, *in* 'Annual International Conference on the Theory and Applications of Cryptographic Techniques', Vol. 6110, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 24–43.

Dolev, D. & Yao, A. C.-C. (1983), 'On the security of public key protocols', *IEEE Transactions on Information Theory* **30**(2), 198–208.

Edris, E. K. K., Aiash, M. & Loo, J. (2019), Investigating network services abstraction in 5g enabled device-to-device (d2d) communications, *in* '2019 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)', IEEE, Leicester, UK, pp. 1660–1665.

Edris, E. K. K., Aiash, M. & Loo, J. (2020*a*), The case for federated identity management in 5g communications, *in* '5th IEEE International Conference on Fog and Mobile Edge Computing (FMEC 2020)', IEEE, Paris, France.

Edris, E. K. K., Aiash, M. & Loo, J. (2020*b*), Formal verification and analysis of primary authentication based on 5g-aka protocol, *in* 'The Third International Symposium on 5G Emerging Technologies (5GET 2020)', IEEE, Paris, France.

Edris, E. K. K., Aiash, M. & Loo, J. (2020*c*), Network service federated identity (ns-fid) protocol for service authorization in 5g network, *in* '5th IEEE International Conference on Fog and Mobile Edge Computing (FMEC 2020)', IEEE, Paris, France.

Edris, E. K. K., Aiash, M. & Loo, J. (2021*a*), 'Dcss protocol for data caching and sharingsecurity in 5g network', *Networks* .

Edris, E. K. K., Aiash, M. & Loo, J. (2021*b*), 'Formal verification of authentication and service authorization protocols in 5g enabled device-to-device communications using proverif', *Electronics* .

Edris, E. K. K., Aiash, M. & Loo, J. (2021*c*), Security in network services delivery for 5g enabled d2d communications: Challenges and solutions, *in* D. R. Montasari, P. H. Jahankhani & D. H. Al-Khateeb, eds, 'Challenges in the IoT and Smart Environments - A Practitioners' Guide to Security, Ethics and Criminal Threats', 1 edn, Advanced Sciences and Technologies for Security Applications, Springer.

Edris, E. K. K., Aiash, M. & Loo, J. (Submitted for Publication*a*), '5g security protocol performance evaluation using applied pi calculus and markov chain', *International Journal of Communication Networks and Distributed Systems* .

Edris, E. K. K., Aiash, M. & Loo, J. (Submitted for Publication*b*), 'A security framework for 5g enabled d2d communications network', *IEEE Journal of Internet of Things* .

Edris, E. K. K., Aiash, M., Loo, J. & Alhakeem, M. S. (In Press), 'Formal verification of secondary authentication protocol for 5g secondary authentication', *International Journal of Security and Networks* .

Edwall, T. (2011), The network of information: Architecture and applications, Technical report, SAIL Project Team. **URL:** *https://sail-project.eu/wp-content/uploads/2011/08/SAIL_DB1_v1_0_final-Public.pdf. (Accessed 02 June 2019)*

Engel, T. (2014), Ss7: Locate. track. manipulate, *in* 'Talk at 31st Chaos Communication Congress'.

Enisa (2018), Signalling security in telecom ss7/diameter/5g, Technical report, Enisa. **URL:** *https://www.enisa.europa.eu/publications/signalling-security-in-telecom-ss7-diameter-5g (Accessed 05 September 2019)*

ETSI (2014), Network functions virtualisation (nfv); architectural framework, Technical Report ETSI GS NFV 002 V1.2.1 (2014-12), ETSI Industry Specification Group (ISG). **URL:** *http://www.etsi.org/standards/. (Accessed 10 June 2018)*

Fang, D., Qian, Y. & Hu, R. Q. (2018), 'Security for 5g mobile wireless networks', *IEEE Access* **6**, 4850–4874.

Fang, D. & Ye, F. (2018), Identity management framework for e-health systems over 5g networks, *in* '2018 IEEE International Conference on Communications (ICC)', IEEE, pp. 1–6.

Fanian, A., Berenjkoub, M., Saidi, H. & Gulliver, T. A. (2010), A scalable and efficient key establishment protocol for wireless sensor networks, *in* '2010 IEEE Globecom Workshops', IEEE, Miami, FL, USA, pp. 1533–1538.

Fodor, G., Dahlman, E., Mildh, G., Parkvall, S., Reider, N., Miklós, G. & Turányi, Z. (2012), 'Design aspects of network assisted device-to-device communications', *IEEE Communications Magazine* **50**(3).

Gandotra, P., Jha, R. K. & Jain, S. (2017), 'A survey on device-to-device (d2d) communication: Architecture and security issues', *Journal of Network and Computer Applications* **78**, 9–29.

Gao, Y., Hu, S., Tang, W., Li, Y., Sun, Y., Huang, D., Cheng, S. & Li, X. (2018), 'Physical layer security in 5g based large scale social networks: Opportunities and challenges', *Access, IEEE* **6**, 26350–26357.

Garcia, R. & Modesti, P. (2017), An ide for the design, verification and implementation of security protocols, *in* '2017 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)', IEEE, pp. 157–163.

Ghali, C., Tsudik, G. & Uzun, E. (2014*a*), 'Elements of trust in named-data networking', *ACM SIGCOMM Computer Communication Review* **v44**, 12–19.

Ghali, C., Tsudik, G. & Uzun, E. (2014*b*), Needle in a haystack: Mitigating content poisoning in named-data networking, *in* 'NDSS Symposium'.

Girault, M. (1991), Self-certified public keys, *in* 'Workshop on the Theory and Application of of Cryptographic Techniques', Springer, Berlin, Heidelberg, pp. 490–497.

Goldwasser, S. & Micali, S. (1984), 'Probabilistic encryption', *Journal of computer and system sciences* **28**(2), 270–299.

Golrezaei, N., Molisch, A. F., Dimakis, A. G. & Caire, G. (2013), 'Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution', *Communications Magazine, IEEE* **51**(4), 142–149.

Group, O. (1998), 'Distributed audit service (xdas)'. **URL:** *http://www.opengroup.org/security/das/xdas_int.htm. (Accessed 05 June 2019)*

Gupta, A. & Jha, R. K. (2015), 'A survey of 5g network: Architecture and emerging technologies', *IEEE Access* **3**, 1206–1232.

Gupta, S., Parne, B. L. & Chaudhari, N. S. (2018), A generic construction for efficient and secure aka protocol in 5g network, *in* '2018 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)', pp. 1–6.

Haddad, Z., Fouda, M. M., Mahmoud, M. & Abdallah, M. (2020), Blockchain-based authentication for 5g networks, *in* '2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)', IEEE, pp. 189–194.

Haus, M., Waqas, M., Ding, A. Y., Li, Y., Tarkoma, S. & Ott, J. (2017), 'Security and privacy in device-to-device (d2d) communication: A review', *IEEE Communications Surveys & Tutorials* **19**(2), 1054–1079.

Hillston, J. (2005), *A compositional approach to performance modelling*, Vol. 12, Cambridge University Press.

Hodjat, A. & Verbauwhede, I. (2002), The energy cost of secrets in ad-hoc networks (short paper), *in* 'Proc. IEEE Circuits and Systems Workshop (CAS)', Citeseer.

Huang, H., Ahmed, N. & Karthik, P. (2011), 'On a new type of denial of service attack in wireless networks: The distributed jammer network', *IEEE Transactions on Wireless Communications* **10**(7), 2316–2324.

Hughes, J., Cantor, S., Hodges, J., Hirsch, F., Mishra, P., Philpott, R. & Maler, E. (2005), 'Profiles for the oasis security assertion markup language (saml) v2. 0', *OASIS standard* .

Hussain, S. R., Chowdhury, O., Mehnaz, S. & Bertino, E. (2018), Lteinspector: A systematic approach for adversarial testing of 4g lte, *in* 'Network and Distributed Systems Security (NDSS) Symposium 2018'.

ITU-T (2019), 'Itu-t recommendation x. 509— iso/iec 9594-8:" information technology-open systems interconnection-the directory: Public-key and attribute certificate frameworks"', *Tecnical report* .

Jacobson, V. (2009), 'A description of content-centric networking (ccn)', *Future Internet Summer School (FISS)* **2018**(Nov 28,). **URL:** *https://named-data.net/publications/van-ccn-bremen-description/. (Accessed 10 March 2021)*

Jin, H., Xu, D., Zhao, C. & Liang, D. (2017), 'Information-centric mobile caching network frameworks and caching optimization: a survey', *EURASIP Journal on Wireless Communications and Networking* **2017**(1), 1–32.

Jones, M. B., Bradley, J. & Sakimura, N. (2015), Rfc 7519: Json web token (jwt), Rfc, IETF. **URL:** *https://tools.ietf.org/html/rfc7519 . (Accessed 10 December 2019)*

Kim, C., Lee, S. & Lee, S. (2017), 'Multi-device to multi-device (md2md) content-centric networking based on multi-rat device', *Symmetry* **9**(12).

Kodali, R. K., Gundabathula, S. K. & Boppana, L. (2014), Multi level secure leach protocol model using ns-3, *in* '2014 First International Conference on Networks & Soft Computing (ICNSC2014)', IEEE, pp. 198–202.

Kodali, R. K. & Kirti, B. (2020), Ns-3 model of an iot network, *in* '- 2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA)', pp. 699–702. ID: 1.

Koutsos, A. (2019), The 5g-aka authentication protocol privacy, *in* '2019 IEEE European Symposium on Security and Privacy (EuroS&P)', IEEE, pp. 464–479. ID: proquest2135414227.

Kuriharay, J., Uzun, E. & Wood, C. A. (2015), An encryption-based access control framework for content-centric networking, *in* '2015 IFIP networking conference (IFIP networking)', IEEE, pp. 1–9.

Kuroda, M., Yoshida, M., Ono, R., Kiyomoto, S. & Tanaka, T. (2004), 'Secure service and network framework for mobile ethernet', *Wireless Personal Communications; An International Journal* **29**(3), 161–190. ID: springer_jourWIRE.0000047061.87235.8b.

Küsters, R. & Truderung, T. (2009), Using proverif to analyze protocols with diffie-hellman exponentiation, *in* '2009 22nd IEEE Computer Security Foundations Symposium', IEEE, pp. 157–171. ID: ieee_s5230620.

Lamport, L. (1994), 'The temporal logic of actions', *ACM Trans. Program. Lang. Syst.* **16**(3), 872–923. **URL:** *http://doi.acm.org/10.1145/177492.177726*

Leroy, X., Damein, D., Alain, F., Jacques, G., Didier, R. & Vouillon, J. (2019), The OCaml system release 4.09: Documentation and user's manual, PhD thesis, Inria.

Leung-Yan-Cheong, S. & Hellman, M. E. (1978), 'The gaussian wire-tap channel', *IEEE Transactions on Information Theory* **24**(4), 451–456.

Liang, C. (2015), 'Wireless virtualization for next generation mobile cellular networks', *IEEE Wireless Communications Magazine* **22**(1), 61–69.

Liang, C. & Yu, F. (2015), 'Wireless network virtualization: A survey, some research issues and challenges', *IEEE Communications Surveys & Tutorials* **17**(1), 358–380.

Lichtman, M., Rao, R., Marojevic, V., Reed, J. & Jover, R. P. (2018), 5g nr jamming, spoofing, and sniffing: Threat assessment and mitigation, *in* '2018 IEEE International Conference on Communications Workshops (ICC Workshops)', IEEE, pp. 1–6.

Lior, A. & DeKok, A. (2013), Remote authentication dial in user service (radius) protocol extensions, Rfc, IETF. **URL:** *https://tools.ietf.org/html/rfc6929. (Accessed 05 February 2019)*

Liu, D., Chen, B., Yang, C. & Molisch, A. F. (2016), 'Caching at the wireless edge: design aspects, challenges, and future directions', *Communications Magazine, IEEE* **54**(9), 22–28.

Liu, H., Chen, Z., Tian, X., Wang, X. & Tao, M. (2014), 'On content-centric wireless delivery networks', *IEEE Wireless Communications Magazine* **21**(6), 118–125.

Liu, L. & Yu, W. (2018), 'A d2d-based protocol for ultra-reliable wireless communications for industrial automation', *Wireless Communications, IEEE Transactions on* **PP**(99), 1.

Loo, J. & Aiash, M. (2015), 'Challenges and solutions for secure information centric networks: A case study of the netinf architecture', *Journal of Network and Computer Applications* **50**, 64–72.

Lowe, G. (1997), A hierarchy of authentication specifications, *in* 'Proceedings 10th Computer Security Foundations Workshop', IEEE, pp. 31–43. ID: ieee_s596782.

Malnar, M. & Jevtić, N. (2020), 'A framework for performance evaluation of vanets using ns-3 simulator', *Promet-Traffic&Transportation* **32**(2), 255–268.

Mao, W. (2004), *Modern cryptography : theory and practice*, Prentice Hall PTR, Upper Saddle River, N.J.

Martinez-Julia, P. & Gomez-Skarmeta, A. (2012), 'Using identities to achieve enhanced privacy in future content delivery networks', *Computers and Electrical Engineering* **38**(2), 346–355.

MATLAB (2019), *MATLAB (R2019a)*, The MathWorks Inc., Natick, Massachusetts.

Meier, S., Schmidt, B., Cremers, C. & Basin, D. (2013), The tamarin prover for the symbolic analysis of security protocols, *in* N. Sharygina & H. Veith, eds, 'Computer Aided Verification', Vol. 8044, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 696–701. **URL:** *https://www.research-collection.ethz.ch/handle/20.500.11850/68108*

Menezes, A. J., Oorschot, P. C. V. & Vanstone, S. A. (2018), *Handbook of applied cryptography*, CRC Press, Boca Raton, Florida, USA. Includes bibliographical references and index. ID: alma991001301199704781.

Mezzavilla, M., Zhang, M., Polese, M., Ford, R., Dutta, S., Rangan, S. & Zorzi, M. (2018), 'End-to-end simulation of 5g mmwave networks', *IEEE Communications Surveys & Tutorials* **20**(3), 2237–2263.

Mwangama, J., Ventura, N., Willner, A., Al-Hazmi, Y., Carella, G. & Magedanz, T. (2015), Towards mobile federated network operators, *in* 'Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)', IEEE, pp. 1–6.

Norma, G. (2016), Ran architecture components – intermediate report, Technical report, 5GPPP.

Nottegar, C., Priami, C. & Degano, P. (2001), 'Performance evaluation of mobile processes via abstract machines', *IEEE Transactions on Software Engineering* **27**(10), 867–889.

Nsnam (2021), 'ns-3 a discrete-event network simulator for internet systems'.

Nunes, I. O. & Tsudik, G. (2018), Krb-ccn: Lightweight authentication & access control for private content-centric networks, *in* 'International Conference on Applied Cryptography and Network Security', Springer, pp. 598–15.

O'Hanlon, P., Borgaonkar, R. & Hirschi, L. (2017), Mobile subscriber wifi privacy, *in* 'IEEE Security and Privacy Workshops (SPW)', Vol. 2017-, pp. 169–178. ID: ieee_s8227304.

Ohlman, B., Davies, E., Spirou, S., Pentikousis, K. & Boggia, G. (2014), 'Information-centric networking: Evaluation methodology', *IETF (The Internet Engineering Task Force) Request for Comments* . **URL:** *https://tools.ietf.org/html/draft-irtf-icnrg-evaluation-methodology-01. (Accessed 05 January 2021)*

Panaousis, E., Alpcan, T., Fereidooni, H. & Conti, M. (2014), Secure message delivery games for device-to-device communications, *in* R. Poovendran & W. Saad, eds, 'Decision and Game Theory for Security', Springer International Publishing, Switzerland, pp. 195–215.

Pardo, J. L. G. (2013), *Introduction to Public-Key Cryptography: The Diffie–Hellman Protocol*, Introduction to Cryptography with Maple, Springer, Berlin, Heidelberg, pp. 399–417.

Park, Y. & Park, T. (2007), A survey of security threats on 4g networks, *in* 'IEEE Globecom Workshops', IEEE, pp. 1–6.

Parvez, I., Rahmati, A., Guvenc, I., Sarwat, A. I. & Dai, H. (2018), 'A survey on low latency towards 5g: Ran, core network and caching solutions', *IEEE Communications Surveys & Tutorials* **20**(4), 3098–3130.

Piro, G., Baldo, N. & Miozzo, M. (2011), An lte module for the ns-3 network simulator., *in* 'SimuTools', pp. 415–422.

Pratas, N. & Popovski, P. (2013), 'Low-rate machine-type communication via wireless device-to-device (d2d) links', *arXiv preprint arXiv:1305.6783* . ID: proquest2085242805.

Priami, C. (1996), Stochastic pi-calculus with general distributions, *in* 'Proc. of the 4th Workshop on Process Algebras and Performance Modelling (PAPM'96)', Citeseer, pp. 41–57.

Priya, V. & Sakthisaravanan, B. (2015), Information centric network for secure data transmission in dtn, *in* 'International Confernce on Innovation Information in Computing Technologies', IEEE, pp. 1–4.

Qin, B., Liu, S., Sun, S., Deng, R. H. & Gu, D. (2017), 'Related-key secure key encapsulation from extended computational bilinear diffie–hellman', *Information Sciences* **406**, 1–11.

Ravindran, R. (2019), 'Enabling icn in 3gpp's 5g nextgen core architecture', *IETF (The Internet Engineering Task Force) Request for Comments* **2019**(Jan 5,). **URL:** *https://tools.ietf.org/id/draft-ravi-icnrg-5gc-icn-00.html. (Accessed 03 March 2021)*

Ravindran, R., Chakraborti, A., Amin, S. O., Azgin, A. & Wang, G. (2017), '5g-icn: Delivering icn services over 5g using network slicing', *IEEE Communications Magazine* **55**(5), 101–107.

Recordon, D. & Fitzpatrick, B. (2006), 'Openid authentication 1.1', *Finalized OpenID Specification* . **URL:** *https://openid.net/foundation/*

Research, G. (2013), 'Cloud services brokerage'. **URL:** *http://www.gartner.com/it-glossary/cloud-services-brokerage-csb. (Accessed 07 August 2020)*

RIFS, G. (2016), Diameter roaming security - proposed permanent reference document, Technical report, GSMA.

Riley, G. F. & Henderson, T. R. (2010), *The ns-3 network simulator*, Modeling and tools for network simulation, Springer, pp. 15–34.

Rupprecht, D., Dabrowski, A., Holz, T., Weippl, E. & Popper, C. (2018), 'On security research towards future mobile network generations', *IEEE Communications Surveys & Tutorials* **20**(3), 2518–2542. ID: ieee_s8329226.

Sandhu, R. S. & Samarati, P. (1994), 'Access control: principle and practice', *IEEE communications magazine* **32**(9), 40–48.

Sbai, O. & Elboukhari, M. (2019), A simulation analyses of manet's attacks against olsr protocol with ns-3, *in* 'The Proceedings of the Third International Conference on Smart City Applications', Springer, pp. 605–618.

SECG (2009), 'Sec 1: Recommended elliptic curve cryptography'. **URL:** *http://www.secg.org/sec1-v2.pdf. (Accessed 10 January 2020)*

SECG (2010), 'Sec 2: Recommended elliptic curve domain parameters'. **URL:** *http://www.secg.org/sec1-v2.pdf. (Accessed 10 January 2020)*

Seok, B., Sicato, J. C. S., Erzhena, T., Xuan, C., Pan, Y. & Park, J. H. (2020), 'Secure d2d communication for 5g iot network based on lightweight cryptography', *Applied Sciences* **10**(1), 217.

Shamir, A. (1985), Identity-based cryptosystems and signature schemes, *in* 'Advances in Cryptology', Springer-Verlag New York, Inc., New York, NY, USA, p. 47–53.

Sharma, S. K., Woungang, I., Anpalagan, A. & Chatzinotas, S. (2019), 'Towards tactile internet in beyond 5g era: Recent advances, current issues and future directions', *arXiv preprint arXiv:1908.07337* .

Shoup, V. (2001), 'A proposal for an iso standard for public key encryption (version 2.1)', *IACR e-Print Archive* **112**.

Sisto, R., Copet, P. B., Avalle, M. & Pironti, A. (2018), 'Formally sound implementations of security protocols with javaspi', *Formal Aspects of Computing* **30**(2), 279–317.

Smart, N. P. (1999), 'The discrete logarithm problem on elliptic curves of trace one', *Journal of Cryptology* **12**(3), 193–196.

Stewart, W. J. (1994), *Introduction to the numerical solution of Markov chains*, Princeton University Press.

Sun, L. & Du, Q. (2017), 'Physical layer security with its applications in 5g networks: A review', *Communications, China* **14**(12), 1–14.

Targali, Y., Choyi, V. & Shah, Y. (2013), Seamless authentication and mobility across heterogeneous networks using federated identity systems, *in* '2013 IEEE International Conference on Communications Workshops (ICC)', IEEE, pp. 1232–1237.

Thanh, T. Q., Rebahi, Y. & Magedanz, T. (2014), A diameter based security framework for mobile networks, *in* '2014 International Conference on Telecommunications and Multimedia (TEMU)', IEEE, pp. 7–12.

Tourani, R., Misra, S., Mick, T. & Panwar, G. (2018), 'Security, privacy, and access control in information-centric networking: A survey', *IEEE Communications Surveys & Tutorials* **20**(1), 566–600.

Varga, A. (2010), *Modeling and tools for network simulation*, Springer, chapter OMNeT++, pp. 35–59.

VirtuWind (2017), Deliverable d3.2 detailed intra-domain sdn & nfv architecture, Technical report, VirtuWind Project.

Vo, T. Q. N.-S., Duong, A., Hoang, D. T. & Kortun, A. (2018), 'Optimal video streaming in dense 5g networks with d2d communications', *IEEE Access* **6**, 209–223. ID: ieee10.1109/ACCESS.2017.2761978.

Vollbrecht, J. R., Aboba, B., Blunk, L. J., Levkowetz, H. & Carlson, J. (2004), Extensible authentication protocol (eap), Rfc, IETF. **URL:** *https://tools.ietf.org/html/rfc3748. (Accessed 05 February 2020)*

Wang, K., Yu, F. R., Li, H. & Li, Z. (2017), 'Information-centric wireless networks with virtualization and d2d communications', *IEEE Wireless Communications* **24**(3), 104–111.

Wang, M. & Yan, Z. (2017), 'A survey on security in d2d communications', *MOBILE NETWORKS & APPLICATIONS* **22**(2), 195–208.

Wang, Y., Xu, M., Feng, Z., Li, Q. & Li, Q. (2014), Session-based access control in information-centric networks: Design and analyses, *in* '2014 IEEE 33rd International Performance Computing and Communications Conference (IPCCC)', IEEE, pp. 1–8.

Woo, T. Y. & Lam, S. S. (1998), Designing a distributed authorization service, *in* 'Proceedings. IEEE INFOCOM'98, the Conference on Computer Communications. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Gateway to the 21st Century (Cat. No. 98', Vol. 2, IEEE, pp. 419–429.

Wu, X., Tavildar, S., Shakkottai, S., Richardson, T., Li, J., Laroia, R. & Jovicic, A. (2013), 'Flashlinq: a synchronous distributed scheduler for peer-to-peer ad hoc networks', *IEEE/ACM Transactions on Networking (TON)* **21**(4), 1215–1228. ID: acm2525583.

Wu, Y., Khisti, A., Xiao, C., Caire, G., Wong, K.-K. & Gao, X. (2018), 'A survey of physical layer security techniques for 5g wireless networks and challenges ahead', *IEEE Journal on Selected Areas in Communications* **36**(4), 679–695.

Xu, R., Chen, Y., Blasch, E. & Chen, G. (2018), A federated capability-based access control mechanism for internet of things (iots), *in* 'Sensors and Systems for Space Applications XI', Vol. 10641, International Society for Optics and Photonics, p. 106410U.

Yang, C., Chen, Z., Xia, B. & Wang, J. (2015), 'When icn meets c-ran for hetnets: an sdn approach', *Communications Magazine, IEEE* **53**(11), 118–125.

Yi, X., Paulet, R. & Bertino, E. (2013), 'Private information retrieval', *Synthesis Lectures on Information Security, Privacy, and Trust* **4**(2), 1–114.

Zeltsan, Z. (2005), 'Security architecture for systems providing end-to-end communications'. **URL:** *http://www.itu.int/ITU-T/worksem/ngn/200505/presentations/s5- zeltsan.pdf. (Accessed 05 January 2019)*

Zhang, A. & Lin, X. (2017), 'Security-aware and privacy-preserving d2d communications in 5g', *Network, IEEE* **31**(4), 70–77.

Zhang, J., Yang, L., Cao, W. & Wang, Q. (2020), 'Formal analysis of 5g eap-tls authentication protocol using proverif', *IEEE Access* .

Zhang, T., Fang, X., Liu, Y. & Nallanathan, A. (2019), 'Content-centric mobile edge caching', *IEEE Access* **8**, 11722–11731.

Zhang, X., Chang, K., Xiong, H., Wen, Y., Shi, G. & Wang, G. (2011), Towards name-based trust and security for content-centric network, *in* '2011 19th IEEE International Conference on Network Protocols', IEEE, pp. 1–6.

Zhou, Z., Yu, H., Xu, C., Zhang, Y., Mumtaz, S. & Rodriguez, J. (2018), 'Dependable content distribution in d2d-based cooperative vehicular networks: A big data-integrated coalition game approach', *IEEE Transactions on Intelligent Transportation Systems* **19**(3), 953–964.

Appendices

Appendix A

A.1 5G-AKA Protocol (Model A) ProVerif Source Code

```
(*This file is part of a PhD ProVerif Simulation**Author: Ed Kamya Kiyemba Edris*)
(*5G Authentication and Key Agreement Protocol*) (*UE-SEAF-AUSF-ARPF*)
(*Primary Authentication*)

(*Public channel between the UE and the SN*)
free pubChannel :channel.
(*Secure channel between the SN and the HN*)
free secChannel:channel [private].

(*types*)
type key.
type id.
type nonce.
type msgMac.
type msgHdr.
type pkey.
type skey.

(*constant message headers*)
const REG: msgHdr.
const NAUSF_AUTH_REQ: msgHdr.
const NUDM_GET_REQ: msgHdr.
const NUDM_GET_RES: msgHdr.
const NAUSF_AUTH_RES: msgHdr.
const AUTH_REQ: msgHdr.
const AUTH_RES: msgHdr.
const NAUSF_REQ_RES: msgHdr.
const NAUSF_AUTH_RESULT: msgHdr.
const ERR_MSGmac:msgHdr.
const ERR_MSGsynch:msgHdr.
```

```
(*Functions*)
fun f2(key, nonce): bitstring.
fun f3(key, nonce): bitstring.
fun f4(key, nonce): bitstring.
fun f5(key, nonce): bitstring.
fun h(bitstring) : bitstring.
fun SHA256(nonce,bitstring) : bitstring.
fun hash(bitstring, bitstring) : bitstring.
fun kdf_ausf(bitstring, bitstring, id): key.
fun kdf_seaf(key, id): key.
(* Public key encryption *)
fun pk(skey): pkey.
fun encrypt(bitstring, pkey): bitstring.
reduc forall x: bitstring, y: skey; decrypt(encrypt(x,pk(y)),y) = x.
(* Mac *)
fun f1(bitstring, key):bitstring.
(* XOR *)
fun xor(bitstring, bitstring) : bitstring.
equation forall m1: bitstring, m2: bitstring; xor(m1,xor(m1,m2)) = m2.

(*Type  Converter*)
fun nonce_to_bitstring(nonce): bitstring [data, typeConverter].
fun bitstring_to_key(bitstring): key [data, typeConverter].
fun bitstring_to_id (bitstring): id [data, typeConverter].
fun macFail():bitstring.
fun synchFail():bitstring.
fun Fail(bitstring):bitstring.

(*the table id/keys shared between UE & HN*)
table keys(id, key).

(*Queries*)
free SecretUE: bitstring [private].
query attacker(SecretUE).
free SecretSEAF: bitstring [private].
query attacker(SecretSEAF).
free SecretAUSF: bitstring [private].
query attacker(SecretAUSF).
free supi:id [private].
query attacker (supi).
free ki:key [private].
query attacker (ki).
free kseaf:key [private].
query attacker (kseaf).

(*Events used to specify correspondence assertions*)
```

```
event begUE(id, id, key).
event endUE(id, id, key).
event beginSEAF(bitstring).
event endSEAF(bitstring).
event e3(bitstring).
event e2(bitstring).
event e1(bitstring,nonce).

(*Check authentication of UE to SN/HN and SN/HN to UE*)
query x1: id, x2: id, x3: key; event (endUE(x1, x2, x3)) ==>
event(begUE(x1, x2, x3)).
query x1: bitstring; event (endSEAF(x1)) ==> event (beginSEAF(x1)).
query x1: id, x2: id, x3: key; inj-event (endUE(x1, x2, x3)) ==>
inj-event (begUE(x1, x2, x3)).
query x1: bitstring, x2:nonce;  inj-event(endSEAF (x1)) ==>
(inj-event(beginSEAF(x1)) && (inj-event (e3(x1)) ==>
(inj-event (e2(x1)) ==> (inj-event (e1(x1,x2)))))).

(*Processes*)
let processUE (supi:id, hnid_ue:id,ki:key)=
new sqn_ue:bitstring;
new suci:bitstring;
new snn_ue:id;
new rand:nonce;
insert keys(supi, ki);
new pkhn:pkey;
new skhn:skey;
let (=supi, =hnid_ue) = encrypt(suci,pkhn) in
out(pubChannel, (suci)); (*[Msg1]*)
in(pubChannel, (x:bitstring)); (*[Msg6]*)
event e2(x);
let (xrand:nonce, xautn:bitstring) = x in
let (xored_sqn:bitstring,amf:bitstring, mac:bitstring) = xautn in
let ak = f5(ki, xrand) in
let xsqn = xor (xored_sqn, ak) in
let xmac = f1((xsqn, xrand,amf), ki) in
if xmac = mac then
if xsqn <> sqn_ue then
let res = f2(ki, xrand)in
let ck = f3(ki, xrand)in
let ik = f4(ki, xrand)in
let kausf_ue = kdf_ausf(ck, ik, snn_ue)in
let kseaf = kdf_seaf(kausf_ue, snn_ue) in
event beginSEAF(res);
out(pubChannel, (SecretUE, res));  (*[Msg7]*)
event endUE(supi, snn_ue, kseaf)
```

```
else
out(pubChannel, (macFail)) (*[Msg10]*)
else
let ak_ue = f5(ki, xrand) in
let mac_ue = f1((sqn_ue, xrand), ki) in
let AUTS = (xor(sqn_ue,ak_ue),mac_ue) in
out (pubChannel, (synchFail,AUTS)). (*[Msg11]*)

let processSEAF (snn_sn:id)=
new rand_sn:nonce;
new kseaf:key;
in(pubChannel, (suci: id)); (*[Msg1]*)
out(secChannel, (suci, snn_sn));      (*[Msg2]*)
in(secChannel,(autn:bitstring,rand:nonce, hxres:bitstring)); (*[Msg5]*)
event begUE(supi, snn_sn, kseaf);
out(pubChannel, (autn, rand)); (*[Msg6]*)
event e1(autn, rand_sn);
in(pubChannel, (res_sn:bitstring)); (*[Msg7 ]*)
let hres = SHA256(rand,res_sn) in
if hres = hxres then
out(secChannel, (SecretSEAF, res_sn));   (*[Msg8]*)
event endSEAF(res_sn);
in(secChannel, (supi:id,kseaf:key))   (*[Msg9]*)
else
in(pubChannel, (macFail:bitstring)) (*Msg10*)
else
in(pubChannel, (synchFail:bitstring,AUTS:bitstring)); (*[Msg11]*)
out (secChannel, (synchFail,AUTS)). (*[Msg12]*)

let processAUSF =
in(secChannel,(supi: bitstring,snn:id)); (*[Msg2]*)
out(secChannel,(supi, snn)); (*[Msg3]*)
in(secChannel,(autn:bitstring, rand:nonce,
xres:bitstring, kausf_hn:key, supi:id));      (*[Msg4]*) (* 5G HE AV*)
let hxres = SHA256(rand,xres)in
out(secChannel,(SecretAUSF, autn,rand,hxres));  (*[Msg5]*) (* 5G SE AV*)
in(secChannel, (res_hn:bitstring)); (*[Msg8]*)
if res_hn = xres then
let kseaf = kdf_seaf(kausf_hn, snn) in
event e3(res_hn);
out(secChannel, (supi,kseaf))           (*[Msg9]*)
else
in(secChannel, (synchFail:bitstring,AUTS:bitstring)); (*[Msg12]*)
out (secChannel, (synchFail,AUTS,rand)). (*[Msg13]*)

let processARPF (supi:id,ki:key,amf:bitstring) =
new sqn_ue:bitstring;
in(secChannel,(suci :bitstring,
snn:id)); (*[Msg3]*)
new hnid:id;
new skhn:skey;
let pkhn = pk(skhn)in
let(=supi, =hnid) = decrypt(suci, skhn) in
new rand_hn: nonce;
new amf:bitstring;
new sqn_hn: bitstring;
get keys(=supi, =ki)in
let ak_hn = f5(ki, rand_hn)in
let xored_sqn=xor(f5(ki, rand_hn), sqn_hn) in
let mac_hn = f1 ((sqn_hn, rand_hn, amf), ki)in
let xres = f2 (ki, rand_hn)in
let ck_hn = f3 (ki, rand_hn)in
let ik_hn = f4 (ki,  rand_hn)in
let autn = (xor (sqn_hn,ak_hn),amf, mac_hn) in
let kausf_hn = kdf_ausf(ck_hn, ik_hn, snn)in
out(secChannel,(autn, rand_hn, xres, kausf_hn)) (*[Msg4]*)
else
in (secChannel, (synchFail:bitstring,AUTS:bitstring, rand_hn:nonce)); (*[Msg13]*)
let ak_ue = f5(ki, rand_hn) in
let mac_ue = f1((sqn_ue, rand_hn), ki) in
let AUTS = (xor(sqn_ue,ak_ue),mac_ue) in
if mac_ue = mac_hn then
if sqn_hn <> sqn_ue then
new new_AV:bitstring;
out (secChannel, (new_AV)). (*[Msg14]*)

process
new snn_sn:id;
new supi_sn:id;
new kausf_hn:key;
new snn_hn:id;
let kseaf = kdf_seaf(kausf_hn, snn_hn) in
out (secChannel,kseaf);
insert keys(supi, kseaf);
new ki:key;
new amf:bitstring;
new hnid_ue:id;
((!processUE(supi,hnid_ue,ki))|(!processSEAF(snn_sn))|(!processAUSF)|
(!processARPF(supi,ki,amf)))
```

A.2 5G-AKA Protocol (Model B)

```
(* 5G_AKA 3 Entities - Mutual Authentication and secrecy properties. Result: ok.*)

(*Public channel between the UE, SN and the HN*)
free pubChannel :channel.
(*Secure channel between the SN and the HN*)
free secChannel:channel [private].

(*types*)
type key.
type id.
type nonce.
type msgMac.
type msgHdr.
type pkey.
type skey.

(*constant message headers*)
const REG: msgHdr.
const NAUSF_AUTH_REQ: msgHdr.
const NAUSF_AUTH_RESP: msgHdr.
const AUTH_REQ: msgHdr.
const AUTH_RESP: msgHdr.
const NAUSF_REQ_RESULT: msgHdr.
const NAUSF_AUTH_RESULT: msgHdr.
const ERR_MSGmac:msgHdr.
const ERR_MSGsynch:msgHdr.

(*Functions*)
fun f2(key, nonce): bitstring.
fun f3(key, nonce): bitstring.
fun f4(key, nonce): bitstring.
fun f5(key, nonce): bitstring.
fun h(bitstring) : bitstring.
fun SHA256(nonce,bitstring) : bitstring.
fun hash(bitstring, bitstring) : bitstring.
fun kdf_ausf(bitstring, bitstring, id): key.
fun kdf_seaf(key, id): key.
(* Public key encryption *)
fun pk(skey): pkey.
fun encrypt(bitstring, pkey): bitstring.
reduc forall x: bitstring, y: skey; decrypt(encrypt(x,pk(y)),y) = x.
(* Mac *)
fun f1(bitstring, key):bitstring.
(* XOR *)
```

```
fun xor(bitstring, bitstring) : bitstring.
equation forall m1: bitstring, m2: bitstring; xor(m1,xor(m1,m2)) = m2.

(*Type  Converter*)
fun nonce_to_bitstring(nonce): bitstring [data, typeConverter].
fun bitstring_to_key(bitstring): key [data, typeConverter].
fun bitsring_to_id (bitstring): id [data, typeConverter].
fun macFail():bitstring.
fun synchFail():bitstring.
fun Fail(bitstring):bitstring.

(*the table id/keys shared btn UE & HN*)
table keys(id, key).

(*Queries*)
free SecretUE: bitstring [private].
query attacker(SecretUE).
free SecretSN: bitstring [private].
query attacker(SecretSN).
free SecretHN: bitstring [private].
query attacker(SecretHN).
free supi:id [private].
query attacker (supi).
free ki:key [private].
query attacker (ki).
free kseaf:key [private].
query attacker (kseaf).

(*Events used to specify correspondence assertions*)
event begUE(id, id, key).
event endUE(id, id, key).
event begSN(bitstring).
event endSN(bitstring).
event e3(bitstring).
event e2(bitstring).
event e1(bitstring,nonce).

(*Check authentication of UE to SN/HN and SN/HN to UE *)
query x1: id, x2: id, x3: key; event (endUE(x1, x2, x3)) ==>
event(begUE(x1, x2, x3)).
query x1: bitstring; event (endSN(x1)) ==> event (begSN(x1)).
query x1: id, x2: id, x3: key; inj-event (endUE(x1, x2, x3)) ==>
inj-event (begUE(x1, x2, x3)).
query x1: bitstring, x2:nonce;  inj-event(endSN (x1)) ==>
(inj-event(begSN(x1)) && (inj-event (e3(x1)) ==>
(inj-event (e2(x1)) ==> (inj-event (e1(x1,x2)))))).
```

```
(*Processes*)
let processUE (supi:id, hnid_ue:id,ki:key)=
new sqn_ue:bitstring;
new suci_ue:bitstring;
new snn_ue:id;
new rand_ue:nonce;
insert keys(supi, ki);
new pkhn:pkey;
new skhn:skey;
let (=supi, =hnid_ue) = encrypt(suci_ue,pkhn) in
out(pubChannel, (suci_ue));   (*[Msg 1]*)
in(pubChannel, (x:bitstring)); (*[Msg 4]*)
event e2(x);
let (xrand:nonce, xautn:bitstring) = x in
let (xored_sqn:bitstring,amf:bitstring, mac:bitstring) = xautn in
let ak = f5(ki, xrand) in
let xsqn = xor (xored_sqn, ak) in
let xmac = f1((xsqn, xrand), ki) in
if xmac = mac then
if xsqn <> sqn_ue then   (*Authentication HN to UE*)
let res = f2(ki, xrand)in
let ck = f3(ki, rand_ue)in
let ik = f4(ki, rand_ue)in
let kausf_ue = kdf_ausf(ck, ik, snn_ue)in
let kseaf = kdf_seaf(kausf_ue, snn_ue) in
event begSN(res);
out(pubChannel, (SecretUE, res)); (*[Msg 5]*)
event endUE(supi, snn_ue, kseaf)
else
out(pubChannel, (macFail))
else
let ak_ue = f5(ki, xrand) in
let mac_ue = f1((sqn_ue, xrand), ki) in
let AUTS = (xor(sqn_ue,ak_ue),mac_ue) in
out (pubChannel, (synchFail,AUTS)).

let processSN (snn_sn:id) =
new kseaf:key;
in(pubChannel, (suci_sn: id)); (*[Msg 1]*)
out(secChannel, (suci_sn, snn_sn));(*[Msg 2]*)
in(secChannel,(autn:bitstring,rand_sn:nonce,
hxres_sn:bitstring));     (*[Msg 3]*)
event begUE(supi, snn_sn, kseaf);
out(pubChannel, (autn, rand_sn)); (*[Msg 4]*)
event e1(autn, rand_sn);
```

```
in(pubChannel, (res_sn:bitstring)); (*[Msg 5]*)
let hres = SHA256(rand_sn,res_sn) in
if hres = hxres_sn then   (*Authentication UE to HN-SN view*)
out(secChannel, (SecretSN, res_sn));         (*[Msg 6]*)
event endSN(res_sn);
in(secChannel, (supi_sn:id,kseaf:key)) (*[Msg 7]*)
else
in(pubChannel, (macFail:bitstring))
else
in(pubChannel, (synchFail:bitstring,AUTS:bitstring));
out (secChannel, (synchFail,AUTS)).

let processHN (supi:id, ki:key, amf:bitstring) =
new sqn_ue:bitstring;
in(secChannel,(suci_hn :bitstring,
snn_hn:id)); (*[Msg 2]*)
new hnid_hn:id;
new skhn:skey;
let pkhn = pk(skhn)in
let(=supi, =hnid_hn) = decrypt(suci_hn, skhn) in
new rand_hn: nonce;
new sqn_hn: bitstring;
get keys(=supi, =ki)in
let ak_hn = f5(ki, rand_hn)in
let xored_sqn=xor(f5(ki, rand_hn), sqn_hn) in
let mac_hn = f1 ((sqn_hn, rand_hn), ki)in
let xres = f2 (ki, rand_hn)in
let ck_hn = f3 (ki, rand_hn)in
let ik_hn = f4 (ki,  rand_hn)in
let autn = (xor (sqn_hn,ak_hn),amf, mac_hn) in
let hxres_hn = SHA256(rand_hn,xres)in
let kausf_hn = kdf_ausf(ck_hn, ik_hn, snn_hn)in
out(secChannel,(SecretHN, autn, rand_hn, hxres_hn, kausf_hn));   (*[Msg 3]*)
in(secChannel, (res_hn:bitstring)); (*[Msg 6]*)
if res_hn = xres then    (*Authentication UE to HN-HN view*)
let kseaf = kdf_seaf(kausf_hn, snn_hn) in
event e3(res_hn);
out(secChannel, (supi,kseaf)) (*[Msg 7]*)
else
in (secChannel, (synchFail:bitstring,AUTS:bitstring));
let ak_ue = f5(ki, rand_hn) in
let mac_ue = f1((sqn_ue, rand_hn), ki) in
let AUTS = (xor(sqn_ue,ak_ue),mac_ue) in
if mac_ue = mac_hn then
if sqn_hn <> sqn_ue then
out (secChannel, ( synchFail)).
```

```
process
new snn_sn:id;
new supi_sn:id;
new kausf_hn:key;
new snn_hn:id;
let kseaf = kdf_seaf(kausf_hn, snn_hn) in out (pubChannel,kseaf);
insert keys(supi, kseaf);
new ki:key;
new amf:bitstring;
new hnid_ue:id;
((!processUE(supi,hnid_ue,ki))|(!processSN(snn_sn))|(!processHN(supi,ki,amf)))
```

Appendix B

B.1 5G EAP-AKA' Protocol

```
(*This file is part of a PhD ProVerif Simulation**Author: Ed Kamya Kiyemba Edris*)
(*5G Extensible Authetication Protocol*) (*UE-SEAF-AUSF-ARPF*)
(*Primary Authentication*)

free c: channel.
type key.
type nonce.
type host.
type id.

(*Functions*)
fun nonce_to_bitstring(nonce): bitstring [data, typeConverter].
fun bitstring_to_key(bitstring): key [data, typeConverter].
fun bitstring_to_id (bitstring): id [data, typeConverter].
fun macFail():bitstring.
fun synchFail():bitstring.
fun Fail(bitstring):bitstring.
(* encryption *)
fun encrypt(bitstring, key): bitstring.
reduc forall x: bitstring, k: key; decrypt(encrypt(x, k), k) = x.
(*Hash Funtions*)
fun h(bitstring):   bitstring.
fun hash(bitstring):   bitstring.
fun SHA_256(bitstring): bitstring.
fun PRF(key, key, bitstring, id): key.
fun HMAC_SHA_256(key,   bitstring): bitstring.
fun f1(key, bitstring):bitstring.
fun f2(key,  nonce):   bitstring.
fun f3(key,  nonce):   bitstring.
fun f4(key,  nonce):   bitstring.
fun f5(key,  nonce):   bitstring.
fun kdf ():key.
fun at_kdf_ausf(key): key.
fun at_kdf_seaf(key, id): key.
fun at_kdf_ck1(bitstring, bitstring, id, bitstring):key.
fun at_kdf_ik1(bitstring, bitstring, id, bitstring):key.
(* XOR *)
fun xor(bitstring, bitstring) : bitstring.
equation forall m1: bitstring, m2: bitstring; xor(m1,xor(m1,m2)) = m2.
(**Constant*)
const ReqId: bitstring [data].
const ResId: bitstring [data].
const Success: bitstring [data].
free hostA, hostU, hostR, hostS: host.
(* table *)
table keys(host, host, key).

(* events *)
event beginAUSF(host, host, nonce, key).
event endAUSF(host, host, nonce, key).
event beginARPF(host, host, nonce, key).
event endARPF(host, host, nonce, key).
event beginSEAF(id, key).
event endSEAF(id, key).
event beginUE(host, host, nonce, key).
event endUE(host, host, nonce, key).
(*Queries*)
free secretAUSF, secretUE: bitstring [private].
query attacker(secretAUSF); attacker(secretUE).
free supi:id [private].
query attacker (supi).
free kseaf:key [private].
query attacker (kseaf).
free k:key [private].
query attacker (k).

(*Events*)
query u: host, a: host, r: nonce, kseaf:key, k: key;
event(endAUSF(u, a, r, k)) ==> event(beginUE(u, a, r, k)).
query supi: id, kseaf:key; event(endSEAF(supi, kseaf)) ==>
event(beginSEAF(supi, kseaf)).
query u: host, a: host, r: nonce, kseaf:key, k: key;
inj-event(endAUSF(u, a, r, k)) ==> inj-event(beginUE(u, a, r, k)).
query supi: id, kseaf:key; inj-event(endSEAF(supi, kseaf)) ==>
inj-event(beginSEAF(supi, kseaf)).

(*Processes*)
```

```
let procSEAF(s: host)=
out(c, ReqId); (*msg1*)
in(c, (=ResId, u: host)); (*msg2*)
out(c, (ResId, u)); (*msg3*)
in(c, (at_rand:nonce, at_autn:bitstring, at_mac:bitstring, at_kdf:bitstring,
at_kdf_input:bitstring)); (*msg6*)
out(c, (at_rand, at_autn, at_mac, at_kdf, at_kdf_input));  (*msg7*)
in (c, (at_res:bitstring, at_mac2:bitstring));  (*msg8*)
out(c, (at_res, at_mac2));  (*msg9*)
event beginSEAF(supi, kseaf);
in(c, (=Success, kseaf_sn:key, supi_sn:id)); (*msg10*)
out(c, Success); (*msg11*)
event endSEAF(supi, kseaf);
in(c, (macFail:bitstring)); (*msg12*)
in(c, (synchFail:bitstring,AUTS:bitstring)) ;(*msg12*)
out (c, (synchFail,AUTS)). (*msg13*)

let procUE(u: host) =
new supi_ue:id;
new sqn: bitstring;
new amf: bitstring;
new eap_aka:bitstring;
new snn:id;
in(c, a: host);
get keys(=u, =a, k)in
in(c, =ReqId); (*msg1*)
out(c,(ResId, u));  (*msg2*)
in(c,(at_rand: nonce, at_autn: bitstring, at_mac: bitstring, at_kdf:bitstring,
at_kdf_input:bitstring ));  (*msg7*)
let ck = f3(k, at_rand) in
let ik = f4(k, at_rand)in
let ak = f5(k, at_rand)in
let mac = f1 (k, (sqn, at_rand))in
let ik' = at_kdf_ik1(ik, ck, snn, xor(sqn,ak)) in
let ck' = at_kdf_ck1(ik, ck, snn, xor(sqn,ak)) in
let kausf = at_kdf_ausf (PRF(ck',ik',eap_aka,supi)) in
let k_aut = PRF(ck',ik',eap_aka,supi)   in
let kseaf_ue = at_kdf_seaf(kausf, snn) in
if at_mac = HMAC_SHA_256(k_aut,(at_rand,at_autn)) &&
at_autn = (xor (sqn,ak),amf, at_mac) then
let at_res = f2(k, at_rand) in
let at_mac2 = HMAC_SHA_256(k_aut, at_res)in
event beginUE(u,a,at_rand, k_aut);
out(c, (at_res, at_mac2));  (*msg8*)
in(c, =Success);  (*msg11*)
if a = hostA then
```

```
out(c, encrypt(secretUE, k_aut))  (*msge*)
else
out(c, (macFail))  (*msg12*)
else
let ak = f5(k, at_rand) in
let mac = f1(k,(sqn, at_rand)) in
let AUTS = (xor(sqn,ak),mac) in
out (c, (synchFail,AUTS)).   (*msg12*)

let procAUSF(a: host) =
in(c, (=ResId, u: host));  (*msg3*)
get keys(=u, =a, k)in
out(c, (ResId, u));  (*msg4*)
new supi_hn:id;
new at_rand:   nonce;
new sqn_hn: bitstring;
new amf: bitstring;
new eap_aka:bitstring;
new snn:id;
let ck = f3(k, at_rand) in
let ik = f4(k, at_rand) in
let ak = f5(k, at_rand) in
let mac_hn = f1 (k, (sqn_hn, at_rand))in
let ik' = at_kdf_ik1(ik, ck, snn, xor(sqn_hn,ak)) in
let ck' = at_kdf_ck1(ik, ck, snn, xor(sqn_hn,ak)) in
let autn = (xor (sqn_hn,ak),amf, mac_hn) in
let kausf = at_kdf_ausf (PRF(ck',ik',eap_aka,supi)) in
let k_aut =  PRF(ck',ik',eap_aka,supi)   in
let at_kdf_input = (snn) in
let at_autn = (autn) in
let at_kdf = (kdf) in
let at_mac = HMAC_SHA_256(k_aut, (at_rand, at_autn)) in
in (c, (at_rand:nonce, at_autn:bitstring, at_mac:bitstring, at_kdf:bitstring,
at_kdf_input:bitstring));  (*msg5*)
out(c, (at_rand, at_autn, at_mac, at_kdf, at_kdf_input));  (*msg6*)
in(c, (at_res: bitstring, at_mac2: bitstring));  (*msg9*)
if  at_mac2 = HMAC_SHA_256(k_aut, at_res)   &&
at_res = f2(k, at_rand)   then
out(c, (Success, kseaf, supi));  (*msg10*)
if u = hostU then
out(c, encrypt(secretAUSF, k_aut));  (*msg1*)
event endAUSF(u, a, at_rand, k_aut)  (*msg1*)
else
in (c, (synchFail:bitstring,AUTS:bitstring)); (*msg13*)
let ak_ue = f5(k, at_rand) in
new sqn:bitstring;
```

```
let mac= f1(k ,(sqn, at_rand)) in
let AUTS = (xor(sqn,ak),mac) in
if at_mac2 = at_mac then
if sqn <> sqn_hn then
out (c, (synchFail,AUTS, at_rand)). (*msg14*)

let procARPF(r: host) =
in(c, (=ResId, u: host));  (*msg4*)
get keys(=u,   =hostA,   k)   in
new supi_hn:id;
new at_rand: nonce;
new sqn_hn: bitstring;
new amf: bitstring;
new eap_aka:bitstring;
new snn:id;
let ck = f3(k, at_rand) in
let ik = f4(k, at_rand)   in
let ak = f5(k, at_rand)   in
let mac_hn = f1 (k, (sqn_hn, at_rand))in
let ik' = at_kdf_ik1(ik, ck, snn, xor(sqn_hn,ak)) in
let ck' = at_kdf_ck1(ik, ck, snn, xor(sqn_hn,ak)) in
let autn = (xor (sqn_hn,ak),amf, mac_hn) in
let kausf = at_kdf_ausf (PRF(ck',ik',eap_aka,supi)) in
let k_aut = PRF(ck',ik',eap_aka,supi)   in
let at_kdf_input = (snn) in
let at_autn = (autn) in
let at_kdf = (kdf) in
let at_mac = HMAC_SHA_256(k_aut, (at_rand, at_autn)) in
let kseaf_hn = at_kdf_seaf(kausf, snn) in
out(c, (at_rand,   at_autn,  at_mac, at_kdf, at_kdf_input));  (*msg5*)
event endARPF(u, r, at_rand, k_aut)
else
in (c, (synchFail:bitstring,AUTS:bitstring, at_rand:nonce)); (*msg14*)
new AV:bitstring;
out (c, (AV)).  (*msg15*)

let keyRegistration  =
in(c, (h1: host, h2: host, k: key));
if (h1, h2) <> (hostU, hostA, hostR) then
insert keys(h1, h2, k).

process
new kseaf: key;
insert keys(hostU, hostA, kseaf);
((!procUE(hostU))|(!procAUSF(hostA))|(!procSEAF (hostS))|(!procARPF(hostR))|
(!keyRegistration))
```

Appendix C

C.1 SAP-AKA Protocol

```
(*This file is part of a PhD ProVerif Simulation**Author: Ed Kamya Kiyemba Edris*)
(*Secondary Authentication Protocol - Authentication and Key Agreement*)
(*UE-SMF-SPAAA*) (*Secondary Authentication*)

free c: channel.
type key.
type pkey.
type skey.
type nonce.
type host.
type id.

(*Functions*)
fun nonce_to_bitstring(nonce): bitstring [data, typeConverter].
fun bitstring_to_key(bitstring): key [data, typeConverter].
fun bitstring_to_id (bitstring): id [data, typeConverter].
(* encryption *)
fun encrypt(bitstring, key): bitstring.
reduc forall x: bitstring, k: key; decrypt(encrypt(x, k), k) = x.
(* Public key encryption *)
fun pk(skey): pkey.
fun aencrypt(bitstring, pkey): bitstring.
reduc forall x: bitstring, y: skey; adecrypt(aencrypt(x,pk(y)),y) = x.
(*Hash Funtions *)
fun h(bitstring):   bitstring.
fun hash(bitstring):   bitstring.
fun SHA_256(bitstring): bitstring.
fun PRF(key, key, bitstring, id): key.
fun HMAC_SHA_256(key,   bitstring): bitstring.
fun f1(key, bitstring):bitstring.
fun f2(key, nonce): bitstring.
fun f3(key, nonce): bitstring.
fun f4(key, nonce): bitstring.
fun f5(key, nonce): bitstring.
fun kdf ():key.
fun at_kdf_ausf(key): key.
fun at_kdf_ue3a (id, id):key.
fun at_kdf_ck1(bitstring, bitstring, id, bitstring):key.
fun at_kdf_ik1(bitstring, bitstring, id, bitstring):key.
(* XOR *)
fun xor(bitstring, bitstring) : bitstring.
equation forall m1: bitstring, m2: bitstring; xor(m1,xor(m1,m2)) = m2.

(* table *)
table keys(host, host, key).

(*Constant)
const ReqId: bitstring [data].
const ResId: bitstring [data].
const Success: bitstring [data].
const ServSsReq: bitstring [data].
const ServSsResp: bitstring [data].
const ServReq: bitstring [data].
const ServResp: bitstring [data].
free hostA, hostU, hostS: host.

(* events *)
event endAAA(host, host, nonce, key).
event beginUE(host, host, nonce, key).
event endUE(host, host, nonce, key).
event beginAAA(host, host, nonce, key).

(*queries*)
free secretAAA, secretUE: bitstring [private].
query attacker(secretAAA); attacker(secretUE).
free eid:id [private].
query attacker (eid).
free kue3a:key [private].
query attacker (kue3a).
free k:key [private].
query attacker (k).
query u: host, a: host, r: nonce, kue3a:key, k: key;
event(endAAA(u, a, r, k)) ==> event(beginUE(u, a, r, k)).
query u: host, a: host, r: nonce, kue3a:key, k: key;
event(endUE(u, a, r, k)) ==> event(beginAAA(u, a, r, k)).
query u: host, a: host, r: nonce, kue3a:key, k: key;
inj-event(endAAA(u, a, r, k)) ==> inj-event(beginUE(u, a, r, k)).
```

```
query u: host, a: host, r: nonce, kue3a:key, k: key;
inj-event(endUE(u, a, r, k)) ==> inj-event(beginAAA(u, a, r, k)).

(*Processes*)
let procUE(u: host, kamf:key) =
new gpsi:id;
new sqn: bitstring;
new amf: bitstring;
new eap_aka:bitstring;
new dnn:id;
new servname:id;
new sid:id;
new spid:id;
in(c, a: host);
get keys(=u,   =a, k)in
let ServSsReq = (servname, sid) in
out (c, encrypt(ServSsReq, kamf));  (*msg1*)
in(c,(=ServSsResp, m1:bitstring));  (*msg2*)
let (gpsi:id, spid:id, pksp:pkey, hostS: host) = decrypt(m1, kamf) in
let ServReq = (servname, sid) in
out (c, aencrypt(ServReq, pksp));  (*msg3*)
in(c,(=ReqId, at_any_id:bitstring));  (*msg4*)
out(c,(ResId, gpsi));  (*msg5*)
in(c,(at_rand: nonce, at_autn: bitstring, at_mac: bitstring, at_kdf:bitstring,
at_kdf_input:bitstring ));  (*msg6*)
let at_kdf_input = (dnn) in
new rand:nonce;
let at_rand = (rand) in
let ak = f5(k, at_rand)in
let at_autn = (xor (sqn,ak),amf, at_mac) in
let ck = f3(k, at_rand) in
let ik = f4(k, at_rand)in
let mac = f1 (k, (sqn, at_rand))in
let ik' = at_kdf_ik1(ik, ck, dnn, xor(sqn,ak)) in
let ck' = at_kdf_ck1(ik, ck, dnn, xor(sqn,ak)) in
let kausf = at_kdf_ausf (PRF(ck',ik',eap_aka,gpsi)) in
let k_aut = PRF(ck',ik',eap_aka,gpsi)   in
let k_enc = PRF(ck',ik',eap_aka,gpsi)   in
if at_mac = HMAC_SHA_256(k_aut,(at_rand,at_autn)) &&
at_autn = (xor (sqn,ak),amf, at_mac) then
let at_res = f2(k, at_rand) in
let at_mac2 = HMAC_SHA_256(k_aut, at_res)in
event beginUE(u,a,at_rand, k_aut);
out(c, (at_res, at_mac2));  (*msg7*)
new eid:id;
new kue3a:key;
```

```
in (c, (= Success, kue3a:key, eid:id, k_enc:key)); (*msg8*)
let kue3a = at_kdf_ue3a(eid,spid) in
if a = hostA then
out(c, encrypt(secretUE, k_aut))).

let procSMF(s: host, kamf:key, pksp:pkey)=
new gpsi:id;
new sid:id;
new spid:id;
new eid:id;
in (c, (=ServSsReq, m:bitstring));  (*msg1*)
let (servname:id, sid:id, hostU: host) = decrypt(m, kamf) in
let ServSsResp = (gpsi, spid, pksp) in
out(c,encrypt(ServSsResp, kamf));  (*msg2*)
in(c, (=ServReq, m2:bitstring));  (*msg3*)
out (c, (ServReq));  (*msg3*)
in(c, (=ReqId, at_any_id:bitstring));  (*msg4*)
out(c, (ReqId, at_any_id));  (*msg4*)
in(c, (=ResId, gpsi: id));  (*msg5*)
out(c, (ResId, gpsi));  (*msg5*)
let kue3a = at_kdf_ue3a(eid,spid) in
in (c, (= Success, kue3a:key, eid:id, k_enc:key)); (*msg8*)
out (c, (kue3a, eid, k_enc));  (*msg8*)
in (c, (=Success, eid:id)).  (*msg9*)

let procAAA(a: host, sksp:skey)   =
new at_any_id:bitstring;
in(c, (=ServReq, m2:bitstring));  (*msg3*)
let (servname:id, sid:id, hostU: host) = adecrypt(m2, sksp) in
out (c, (ReqId, at_any_id));  (*msg4*)
in(c, (=ResId, gpsi:id, u: host));  (*msg5*)
get keys(=u, =hostA, k)   in
new gpsi:id;
new at_rand: nonce;
new sqn: bitstring;
new amf: bitstring;
new eap_aka:bitstring;
new dnn:id;
new spid:id;
new sid:id;
new eid:id;
let ck = f3(k, at_rand) in
let ik = f4(k, at_rand) in
let ak = f5(k, at_rand) in
let mac_sp = f1 (k, (sqn, at_rand))in
let ik' = at_kdf_ik1(ik, ck, dnn, xor(sqn,ak)) in
let ck' = at_kdf_ck1(ik, ck, dnn, xor(sqn,ak)) in
let autn = (xor (sqn,ak),amf, mac_sp) in
let kausf = at_kdf_ausf (PRF(ck',ik',eap_aka,gpsi)) in
let k_aut = PRF(ck',ik',eap_aka,gpsi) in
let k_enc = PRF(ck',ik',eap_aka,gpsi) in
let at_kdf_input = (dnn) in
let at_autn = (autn) in
let at_kdf = (kdf) in
let at_mac = HMAC_SHA_256(k_aut, (at_rand, at_autn)) in
out(c, (at_rand, at_autn, at_mac, at_kdf, at_kdf_input));  (*msg6*)
in(c,  (at_res: bitstring, at_mac2: bitstring));  (*msg7*)
if at_mac2 = HMAC_SHA_256(k_aut, at_res)   &&
at_res = f2(k,   at_rand)   then
let kue3a = at_kdf_ue3a(eid,spid) in
out(c, (Success, kue3a, eid)); (*msg8*)
out (c, (Success, eid));  (*msg9*)
if u = hostU then
out(c, encrypt(secretAAA, k_aut));  (*msg*)
event endAAA(u, a, at_rand, k_aut).

let keyRegistration =
in(c, (h1:  host, h2: host, k: key));
if (h1, h2) <> (hostU, hostA) then
insert  keys(h1,   h2,   k).

process
new sksp: skey;
new kamf: key;
new kue3a:   key;
insert keys(hostU, hostS, kamf);
insert keys(hostU, hostA, kue3a);
let pksp = pk(sksp) in out (c, pksp);
((!procUE(hostU, kamf))|(!procAAA(hostA, sksp))|(!procSMF (hostS, kamf, pksp))|
(!keyRegistration))
```

Appendix D

D.1 NS-FId Protocol

```
(*This file is part of a PhD ProVerif Simulation**Author: Ed Kamya Kiyemba Edris*)
(*Network Service Federated Identity Protocol*) (*UE-SMF-IdP-SPAAA-SS*)
(*Federated Authentication and Authorization*)

free c: channel.

(*types*)
type host.
type key.
type pkey.
type skey.
type spkey.
type sskey.
type nonce.
type id.

(*Functions*)
fun nonce_to_bitstring(nonce): bitstring [data, typeConverter].
fun bitstring_to_key(bitstring): key [data, typeConverter].
fun bitstring_to_id (bitstring): id [data, typeConverter].
(*Hash Function*)
fun h(bitstring) : bitstring.
fun SHA256(nonce,bitstring) : bitstring.
fun hash(bitstring, bitstring) : bitstring.
(* Public key encryption *)
fun pk(skey): pkey.
fun aencrypt(bitstring, pkey): bitstring.
reduc forall x: bitstring, y: skey; adecrypt(aencrypt(x,pk(y)),y) = x.
(*Shared Encryption*)
fun encrypt(bitstring, key): bitstring.
reduc forall x: bitstring, k: key; decrypt(encrypt(x,k),k) = x.
```

```
(* Signatures *)
fun spk(sskey): spkey.
fun sign(bitstring, sskey): bitstring.
reduc forall x: bitstring, k: sskey; getmess(sign(x,k)) = x.
reduc forall x: bitstring, k: sskey; checksign(sign(x,k), spk(k)) = x.

(*table*)
table keys(host, host, key).

(*Msg Headers*)
const ServReq: bitstring [data].
const RedSp: bitstring [data].
const AuthzReq: bitstring [data].
const RedIdP: bitstring [data].
const IDTokenReq: bitstring [data].
const IDTokenResp: bitstring [data].
const AccessReq: bitstring [data].
const AccessResp: bitstring [data].
const GrantAccessReq: bitstring [data].
const GrantAccessResp: bitstring [data].

(* events *)
event begin().
event end().
event beginUE(host, host, pkey).
event endUE(host, host, pkey).
event beginSS(host, host, key).
event endSS(host, host, key).
event beginIDP(host, host, pkey).
event endIDP(host, host, pkey).
event beginAAA(host, host, key).
event endAAA(host, host, key, key).

(* free names *)
free hostU, hostF, hostA, hostI, hostS, hostSS: host.
free secretUE_AAA: bitstring [private].
free secretSS_UE: bitstring [private].
free secretIDP_UE: bitstring [private].
free kue3a:key [private].
free kuess:key [private].
free eid:id [private].
free fid:id [private].
free serv:bitstring [private].

(* queries *)
query attacker(secretUE_AAA).
query attacker(secretSS_UE).
query attacker(secretIDP_UE).
query attacker(kue3a).
query attacker(kuess).
query attacker(eid).
query attacker(fid).
query attacker (serv).
query U: host, I: host, K: pkey;
event(endUE(U, I, K)) ==> event(beginIDP(U, I, K)).
query U: host, I: host, K: pkey;
event(endIDP(U, I, K)) ==> event(beginUE(U, I, K)).
query U: host, I: host, K: pkey;
inj-event(endUE(U, I, K)) ==> inj-event(beginIDP(U, I, K)).
query U: host, I: host, K: pkey;
inj-event(endIDP(U, I, K)) ==> inj-event(beginUE(U, I, K)).

(* processes *)
let procUE(U:host, pkue:pkey, skue:skey, A:host, I:host, pkidp:pkey, pksidp:spkey,
pksp:pkey, pksaaa:spkey, SS:host, pksss:spkey)=
in(c, A': host);
new idpid:id;
new gpsi:id;
new spid:id;
new sid:id;
new kamf:key;
new kue3a':key;
new servname:bitstring;
let ServReq = (servname,sid) in
out (c,encrypt(ServReq, kamf)); (*msg1*)
in(c,(=RedSp, m:bitstring));(*msg2*)
let (gpsi':id, spid':id, pksp:pkey, hostS': host) = adecrypt(m, skue) in
let AuthzReq' =(servname, sid, gpsi, spid) in
out(c,aencrypt(AuthzReq, pksp)); (*msg3*)
in(c,(=RedIdP, m1:bitstring)); (*msg4*)
new Ts_1:bitstring;
let (AGr:bitstring, hm1:bitstring, hostA': host) = adecrypt(m1, skue) in
let (pksaaa':spkey, =hostA)= checksign(hm1,pksaaa)in
let hm1' = h(AGr)in
new AGrcode:bitstring;
let AGr'= (AGrcode, eid, idpid, Ts_1) in
new rand_1:nonce;
let IDTokenReq' = (AGr,rand_1)in
out(c, aencrypt(IDTokenReq,pkidp)); (*msg5*)
in(c,(=IDTokenResp, m3:bitstring)); (*msg6*)
let (IdT:bitstring, fid':id, rand_1':nonce,hm3:bitstring, hostI': host) =
adecrypt(m3, skue) in
```

```
let (pksidp':spkey, =hostI)= checksign(hm3,pksidp)in
new Exp_1:bitstring;
new Ts_2:bitstring;
let h3 = h(m3) in
let IdT' = (fid, idpid, rand_1, Ts_2, Exp_1) in
let AccessReq'= (IdT) in
out(c, aencrypt(AccessReq,  pksp));(*msg7*)
in(c,(=AccessResp,m5:bitstring, hm5:bitstring)); (*msg8*)
let (AcT:bitstring, hact:bitstring,kuess':key, hostA:host) = decrypt(m5, kue3a) in
let (pksaaa:spkey, =hostA)= checksign(hm5,pksaaa)in
new asid:id;
new dcap:bitstring;
new ucap:bitstring;
let label = (ucap,dcap) in
new Ts_3:bitstring;
new Exp_2:bitstring;
let AcT' = (fid, asid, label, Ts_3, Exp_2) in
let hact'= h(AcT)in
event beginIDP(U, I, pkidp);
let GrantAccessReq' = (AcT, hact)in
out(c, encrypt(GrantAccessReq, kuess));(*msg9*)
in(c, (=GrantAccessResp, m7:bitstring)); (*msg10*)
let (serv':bitstring, hserv:bitstring, hostSS':host) = decrypt(m7, kuess) in
let (pksss':spkey, =hostSS)= checksign(hserv,pksss)in
new csid:id;
new Exp_3:bitstring;
new Ts_4:bitstring;
new d1:id;
new data:bitstring;
let serv = (fid, d1, data, csid, label, Ts_4, Exp_3) in
let hserv'= h(serv)in
event endUE(U,I, pkidp);
event end().

let procSMF(S:host, U:host, pkue:pkey, pksp:pkey)=
new gpsi:id;
new spid:id;
new servname:id;
new kamf:key;
in (c, (=ServReq, m0:bitstring));  (*msg1*)
let (servname:id, sid:id, hostU: host) = decrypt(m0, kamf) in
let RedSP =(gpsi, spid, pksp)in
out(c, aencrypt(RedSp, pkue)); (*msg2*)
event end().

let procIDP(I:host, pkidp: pkey, skidp: skey,sksidp:sskey, U:host, pkue:pkey,
A:host, pksaaa:spkey)=
new gpsi:id;
new spid:id;
new sid:id;
new idpid:id;
new Exp_1:bitstring;
new Ts_2:bitstring;
event beginUE(U, I, pkidp);
in(c, (=IDTokenReq, m2:bitstring, hm2:bitstring)); (*msg5*)
let (AGr:bitstring, rand_1:nonce, hostA': host) = adecrypt(m2, skidp) in
let (pksaaa':spkey, =hostA)= checksign(hm2,pksaaa)in
let hm2' = h(AGr)in
new AGrcode:bitstring;
new Ts_1:bitstring;
new eid':id;
let AGr' = (AGrcode, eid, idpid, Ts_1)in
let IdT = (fid, idpid, rand_1, Ts_2, Exp_1) in
new fid':id;
let IDTokenResp'=(IdT,fid, rand_1) in
out(c,aencrypt(IDTokenResp, pkue)); (*msg6*)
event endIDP(U, I, pkidp);
event end().

let ProcSPAAA(A:host, sksaaa: sskey, skaaa:skey, U:host, pkue:pkey, pksue:spkey)=
get keys(=U, =A, kue3a') in
new gpsi:id;
new spid:id;
new sid:id;
new eid':id;
new asid:id;
new idpid:id;
new spid':id;
new sid':id;
new Exp_1:bitstring;
new Ts_1:bitstring;
new AGr:bitstring;
in(c,(=AuthzReq, m:bitstring)); (*msg3)
let (servname:id, sid:id, gpsi:id, spid:id, hm: bitstring, hostU: host) =
adecrypt(m, skaaa) in
let (pksue:spkey, =hostU)= checksign(hm,pksue)in
let hm = h(m)in
let AGr = (AGrcode, eid, idpid, Ts_1)in
let RedIdP = (AGr)in
out(c,aencrypt(RedIdP, pkue)); (*msg4*)
new dcap:bitstring;
new ucap:bitstring;
```

```
let label = (ucap,dcap) in
let IdT = (fid, label, Ts_1, Exp_1) in
in(c, (=AccessReq, m4:bitstring)); (*msg7*)
let (IdT':bitstring, hm4:bitstring, hostU': host) =decrypt(m4, kue3a) in
let (pksue':spkey, =hostU)= checksign(hm4,pksue)in
let hm4' =h(m4)in
new kuess':key;
new dcap':bitstring;
new ucap':bitstring;
let label' = (ucap,dcap) in
new Exp_2:bitstring;
new Ts_3:bitstring;
let AcT = (fid, asid, label, Ts_3, Exp_2) in
let hact= h(AcT)in
let AccesResp = (AcT, sign(hact, sksaaa),kuess) in
out(c,aencrypt(AccessResp,pkue)); (*msg8*)
event beginAAA(U, A, kue3a);
event begin();
event end().

let procSS(SS:host, pkss:spkey, U:host, A:host, pkue:pkey, pksp:spkey,
sksss:sskey) =
get keys(=U, =SS, kuess') in
new csid:id;
new idpid:id;
new spid:id;
new sid:id;
event begin();
event beginSS(U, SS, kuess);
in(c, (=GrantAccessReq, m6:bitstring));(*msg9*)
let (AcT:bitstring, hact:bitstring, hostU': host) = decrypt(m6, kuess) in
let (pkue':spkey, =hostA)= checksign(hact,pksp)in
new Exp_1:bitstring;
new Ts_1:bitstring;
new dcap:bitstring;
new ucap:bitstring;
new asid:id;
let label = (ucap,dcap) in
let AcT' = (fid, asid, label, Ts_1, Exp_1) in
let hact'= h(AcT)in
new Exp_2:bitstring;
new Ts_2:bitstring;
new d1:id;
new data:bitstring;
let serv' = (fid, d1, data, csid, label, Ts_2, Exp_2) in
let hserv= h(serv)in
let GrantAccessResp' = (serv, sign(hserv,sksss))in
out(c, encrypt(GrantAccessResp, kuess)); (*msg10*)
event endSS(U, SS, kuess);
event end().

let keyRegistration =
in(c, (h1: host, h2: host, k: key));
if (h1, h2) <> (hostU, hostSS) then
insert keys(h1, h2, k).

process
new skue: skey;
new sksue:sskey;
new sksss: sskey;
new skaaa: skey;
new sksaaa: sskey;
new sksp:sskey;
new skidp: skey;
new sksidp: sskey;
insert keys(hostU, hostSS, kuess);
let pkue = pk(skue) in out(c, pkue);
let pksp = pk(skaaa) in out(c, pksp);
let pkidp = pk(skidp) in out(c, pkidp);
let pksss = spk(sksss) in out(c, pksss);
let pksue = spk(sksue) in out(c, pksue);
let pksaaa = spk(sksaaa) in out(c, pksaaa);
let pksidp = spk(sksidp) in out(c, pksidp);

((!procUE(hostU, pkue, skue, hostA, hostI, pkidp, pksidp, pksp, pksaaa,hostSS,
pksss))|(!procSMF(hostS, hostU, pkue, pksp))|
(!procIDP(hostI, pkidp, skidp, sksidp, hostU, pkue, hostA, pksaaa))|
(!ProcSPAAA(hostA, sksaaa, skaaa, hostU, pkue, pksue))|
(!procSS(hostSS, pksss, hostU, hostA, pkue, pksaaa, sksss))|
(!keyRegistration))
```

Appendix E

E.1 DCSS Protocol

```
(*This file is part of a PhD ProVerif Simulation**Author: Ed Kamya Kiyemba Edris*)
(*Data Caching and Sharing Protocol*) (*UE-SPAAA-SS*)
(*5G Federated Caching and Sharing Authorization*)

free c: channel.

(*types*)
type host.
type key.
type pkey.
type skey.
type spkey.
type sskey.
type nonce.
type id.

(*Functions*)
fun nonce_to_bitstring(nonce): bitstring [data, typeConverter].
fun bitstring_to_key(bitstring): key [data, typeConverter].
fun bitstring_to_id (bitstring): id [data, typeConverter].
(*Hash Functions*)
fun h(bitstring) : bitstring.
fun SHA256(nonce,bitstring) : bitstring.
fun hash(bitstring, bitstring) : bitstring.
(* Public key encryption *)
fun pk(skey): pkey.
fun aencrypt(bitstring, pkey): bitstring.
reduc forall x: bitstring, y: skey; adecrypt(aencrypt(x,pk(y)),y) = x.
(*Shared Encryption*)
fun encrypt(bitstring, key): bitstring.
reduc forall x: bitstring, k: key; decrypt(encrypt(x,k),k) = x.
```

```
(* Signatures *)
fun spk(sskey): spkey.
fun sign(bitstring, sskey): bitstring.
reduc forall x: bitstring, k: sskey; getmess(sign(x,k)) = x.
reduc forall x: bitstring, k: sskey; checksign(sign(x,k), spk(k)) = x.

(*table*)
table keys(host, host, key).

(*Constant*)
const CachTokenReq: bitstring [data].
const CachTokenResp: bitstring [data].
const CachReq: bitstring [data].
const CachAck: bitstring [data].
const ShaTokenReq: bitstring [data].
const ShaTokenResp: bitstring [data].
const ShaReq: bitstring [data].
const ShaAck: bitstring [data].

(* events *)
event end().
event beginUE(host, host, key).
event endUE(host, host, key).
event beginSS(host, host, key).
event endSS(host, host, key).

(* free names *)
free hostU, hostA, hostSS: host.
free secretUE_AAA: bitstring [private].
free secretSS_UE: bitstring [private].
free kuess:key [private].
free kue3a:key [private].
free fid:id [private].
free d1:id [private].

(* queries *)
query attacker(secretUE_AAA).
query attacker(secretSS_UE).
query attacker (kue3a).
query attacker (kuess).
query attacker (fid).
query attacker (d1).
query U: host, SS: host, K: key;
event(endSS(U, SS, K)) ==> event(beginUE(U, SS, K)).
query U: host, SS: host, K: key;
event(endUE(U, SS, K)) ==> event(beginSS(U, SS, K)).
```

```
query U: host, SS: host, K: key;
inj-event(endSS(U, SS, K)) ==> inj-event(beginUE(U, SS, K)).
query U: host, SS: host, K: key;
inj-event(endUE(U, SS, K)) ==> inj-event(beginSS(U, SS, K)).

(* Processes *)
let procUE(U:host, pkue:pkey, A:host, pkaaa:spkey, SS:host, pkss:spkey)=
in(c, A': host);  (**)
get keys(=U, =A, kue3a) in
new skue:skey;
new d1:id;
new dataname:id;
new fid:id;
new spid:id;
new ssid:id;
new Exp_2:bitstring;
new Ts_3:bitstring;
new dcap:bitstring;
new ucap:bitstring;
let label = (ucap,dcap) in
let AcT = (fid, ssid, label, Ts_3, Exp_2) in
let hact= h(AcT)in
get keys(=U, =SS, kuess') in
let CachTokenReq =( fid, dataname, AcT) in
out(c, encrypt(CachTokenReq,kue3a)); (*msg1*)
in(c, (=CachTokenResp, m1:bitstring)); (*msg2*)
let (m1:bitstring, ChT:bitstring, hChT:bitstring, hostA': host) =
decrypt(m1, kue3a) in
let (pkaaa':spkey, =hostA)= checksign(hChT,pkaaa)in
let m1 = (spid, ChT, hChT) in
new Exp_3:bitstring;
new Ts_4:bitstring;
let CachReq = (fid, ChT, hChT) in
let ChT' = (fid, ssid, d1, label, Ts_4, Exp_3) in
let hChT'= h(ChT)in
out(c, encrypt(CachReq, kuess)); (*msg3*)
in (c, (=CachAck, m3:bitstring)); (*msg4*)
let (ssid:id, Ack_1: bitstring, hack_1:bitstring, hostSS: host) =
decrypt(m3, kuess) in
let m3 =(ssid, Ack_1, hack_1) in
let (pkss':spkey, =hostSS)= checksign(hack_1,pkss)in
new cach_d1: bitstring;
let Ack_1' =(cach_d1) in
let hack_1' = h(Ack_1)in
let Hack_1 = h(Ack_1)in
let ShaTokenReq =(fid, d1, Ack_1, Hack_1) in
```

```
out(c, encrypt(ShaTokenReq, kue3a)); (*msg5*)
in(c, (=ShaTokenResp, m5:bitstring)); (*msg6*)
let (ShT:bitstring, hsht:bitstring, hostA: host) = decrypt(m5, kue3a) in
let (pkaaa:spkey, =hostA)= checksign(hsht,pkaaa)in
let m5 = (ShT, hsht) in
let ShT'= (fid, d1, label, Ts_3, Exp_3) in
let hsht' = h(ShT) in
event beginSS(U, SS, kuess);
let ShaTokenReq =(ShT, hsht)in
out(c, encrypt(ShaTokenReq,kuess)); (*msg7*)
in (c, (=ShaAck, m7:bitstring)); (*msg8*)
let (AcK_2:bitstring, Hack_2:bitstring, hostSS: host) = decrypt(m7, kuess) in
let (pkss:spkey, =hostSS)= checksign(Hack_2,pkss)in
let m7 = (AcK_2, Hack_2) in
new Ack_2:bitstring;
let Hack_2' = h(Ack_2)in
event endUE(U, SS, kuess);
event end().

let procAAA(A:host, skaaa: sskey, U:host, SS:host, pkss:spkey)=
new spid:id;
new dataname:id;
new ssid:id;
get keys(=U, =A, kue3a') in
new kuess:key;
new Ts_3:bitstring;
new Exp_2:bitstring;
new ucap:bitstring;
new dcap:bitstring;
new AcT: bitstring;
let label = (ucap, dcap) in
let hact= h(AcT)in
let AcT = (fid, ssid, label, Ts_3, Exp_2) in
in(c, (=CachTokenReq, m0:bitstring)); (*msg1*)
let (dataname:id, fid:id,  AcT':bitstring, hostU': host) = decrypt(m0, kue3a) in
let m0=(dataname, fid, AcT)in
new Exp_3:bitstring;
new Ts_4:bitstring;
let ChT = (fid, ssid, d1, label, Ts_4, Exp_3) in
let hChT= h(ChT)in
let CachTokenResp = (ChT, sign(hChT, skaaa)) in
out(c, encrypt(CachTokenResp, kue3a)); (*msg2*)
new ShT:bitstring;
let label'= (ucap,dcap) in
in(c, (=ShaTokenReq, m4:bitstring)); (*msg5*)
let (Ack_1:bitstring, fid':id, d1:id, Hack_1:bitstring,hostU': host) =
decrypt(m4, kue3a) in
let (pkss':spkey, =hostSS)= checksign(Hack_1,pkss)in
let m=(Ack_1, fid', d1, Hack_1) in
let Hack_1' = h(Ack_1)in
let ShT' = (fid, d1,label, Ts_4, Exp_3) in
let hsht = h(ShT)in
let ShaTokenResp = (ShT, sign(hsht,skaaa)) in
out(c, encrypt(ShaTokenResp, kue3a)); (*msg6*)
event end().

let procSS(SS:host, pkss:spkey, U:host, pkaaa:spkey, pkue:pkey, skss: sskey) =
get keys(=U, =SS, kuess') in
new dataname:id;
new ucap:bitstring;
new dcap:bitstring;
let label = (ucap,dcap) in
new AcT: bitstring;
let hact= h(AcT)in
let (pkaaa':spkey, =hostA)= checksign(hact,pkaaa)in
new Ts_3:bitstring;
new Exp_2:bitstring;
new ssid:id;
let AcT' = (fid, ssid, label, Ts_3, Exp_2) in
event beginUE(U, SS, kuess);
in(c, (=CachReq, m2:bitstring)); (*msg3*)
let (ChT:bitstring,  hChT:bitstring, hostU': host) = decrypt(m2, kuess) in
let m3=(ChT, hChT) in
new Ts_4:bitstring;
new Exp_3:bitstring;
let ChT' = (fid, ssid, d1, label, Ts_4, Exp_3) in
new Ts_3':bitstring;
new cach_d1: bitstring;
let Ack_1 =(cach_d1) in
let hack_1 = h(Ack_1) in
let CachAck = (ssid, Ack_1, sign(hack_1, skss)) in
out(c, encrypt(CachAck, kuess)); (*msg4*)
in(c, (=ShaTokenReq, m7:bitstring)); (*msg7*)
let (ShT:bitstring, hsht:bitstring, hostA': host) = decrypt(m7, kuess) in
let (pkaaa:spkey, =hostA)= checksign(hsht,pkaaa)in
let m7 = (ShT, hsht) in
let hsht' = h(ShT) in
new Exp_3':bitstring;
let ShT' = (fid, d1, label, Ts_3, Exp_3) in
new Ack_2:bitstring;
new Hack_2:bitstring;
let Hack_2' = h(Ack_2) in
let ShaAck =(Ack_2, sign(Hack_2, skss)) in
out(c, encrypt(ShaAck, kuess)); (*msg8*)
event endSS(U, SS, kuess);
event end().

let keyRegistration =
in(c, (h1:   host, h2: host, k: key));
if (h1, h2) <> (hostU, hostSS) then
insert   keys(h1,   h2,   k).

process
new skue: skey;
new skss: sskey;
new skaaa: sskey;
insert keys(hostU, hostSS, kuess);
let pkue = pk(skue) in out(c, pkue);
let pkss = spk(skss) in out(c, pkss);
let pkaaa = spk(skaaa) in out(c, pkaaa);

((!procUE(hostU, pkue, hostU, pkaaa, hostSS, pkss))|
(!procAAA(hostA, skaaa, hostU, hostSS, pkss))|
(!procSS (hostSS, pkss, hostU, pkaaa, pkue, skss))|(!keyRegistration))
```

Appendix F

F.1 DDSec Protocol

```
(*This file is part of a PhD ProVerif Simulation**Author: Ed Kamya Kiyemba Edris*)
(*Device-to-Device Service Security Protocol*) (*UEA-UEB-gNB*)
(*Mutual Authentication, Caching and Sharing Authorization*)
(*With Network Assistance*)

free c: channel.

type host.
type nonce.
type key.
type pkey.
type skey.
type spkey.
type sskey.
type id.

const AdvMsg: bitstring [data].
const IntMsg: bitstring [data].
const DiscMsg: bitstring [data].
const LinkUpMsg: bitstring [data].
const PublMsg: bitstring [data].
const LookUp: bitstring [data].
const SendData: bitstring [data].

(*Funtions*)
fun nonce_to_bitstring(nonce): bitstring [data, typeConverter].
fun bitstring_to_key(bitstring): key [data, typeConverter].
fun bitstring_to_id (bitstring): id [data, typeConverter].
(*Hash Functions*)
fun h(bitstring) : bitstring.
fun SHA256(nonce,bitstring) : bitstring.
fun hash(bitstring, bitstring) : bitstring.
fun hash1(pkey): bitstring.
(* Public key encryption *)
fun pk(skey): pkey.
fun aencrypt(bitstring, pkey): bitstring.
reduc forall x: bitstring, y: skey; adecrypt(aencrypt(x,pk(y)),y) = x.
(* Shared key encryption *)
fun sencrypt(bitstring,key): bitstring.
reduc forall x: bitstring, y: key; sdecrypt(sencrypt(x,y),y) = x.
(* Signatures *)
fun spk(sskey): spkey.
fun sign(bitstring, sskey): bitstring.
reduc forall x: bitstring, k: sskey; getmess(sign(x,k)) = x.
reduc forall x: bitstring, k: sskey; checksign(sign(x,k), spk(k)) = x.

(* events *)
event end().
event beginUEA(host, host, pkey, nonce, nonce).
event endUEA(host, host, pkey, nonce, nonce).
event beginUEB(host, host, pkey, nonce, nonce).
event endUEB(host, host, pkey, nonce, nonce).

(* Host names *)
free A, B, G: host.

(*Table*)
table keys(host, pkey).

(* Free names *)
free Secret: bitstring [private].
query attacker (Secret).
free rand_a:nonce [private].
query attacker (rand_a).
free rand_b:nonce [private].
query attacker (rand_b).
free skuea:skey [private].
query attacker (skuea).
free skueb:skey [private].
query attacker (skueb).
free ua:id [private].
query attacker (ua).
free ub:id [private].
query attacker (ub).

(* Queries *)
query A: host, B: host, K: pkey, rand_a:nonce, rand_b:nonce;
```

```
event(endUEB(A, B ,K, rand_a, rand_b)) ==>
event(beginUEA(A, B, K, rand_a, rand_b)).
query A: host, B: host, K: pkey, rand_a:nonce, rand_b:nonce;
event(endUEA(A, B, K, rand_a, rand_b)) ==>
event(beginUEB(A, B, K, rand_a, rand_b)).
query A: host, B: host, K: pkey, rand_a:nonce, rand_b:nonce;
inj-event(endUEB(A, B, K, rand_a, rand_b)) ==>
inj-event(beginUEA(A, B, K, rand_a, rand_b)).
query A: host, B: host, K: pkey, rand_a:nonce, rand_b:nonce;
inj-event(endUEA(A, B, K, rand_a, rand_b)) ==>
inj-event(beginUEB(A, B, K, rand_a, rand_b)).

(*Processes*)
let processUEA(pkub:spkey, skuea:skey, skua:sskey, kgnba:key) =
let pkuea = pk(skuea) in
new ua:id;
new ub:id;
new dataname:bitstring;
new d1:bitstring;
new rand_na:nonce;
let d1 = SHA256(rand_a,dataname) in
new pkuea1:bitstring;
let hpkuea1 = h(pkuea1) in
let AdvMsg = (ua, rand_na, dataname, d1, pkuea, hpkuea1) in
out(c, sencrypt(AdvMsg, kgnba)); (*msg1*)
in (c, (=DiscMsg, m2:bitstring)); (*msg3*)
let (ub:id, rand_na:nonce, rand_nb:nonce, dataname:bitstring, pkueb:pkey,
hpkueb:bitstring, hostb: host) = sencrypt(m2, kgnba) in
new data:bitstring;
new rand_a:nonce;
new rand_b:nonce;
new Ts_1:bitstring;
let hPublmsg = h(PublMsg) in
let PublMsg = (ua, dataname, d1, rand_a, pkuea, Ts_1, sign(hPublmsg, skua)) in
out(c, aencrypt(PublMsg, pkueb)); (*msg5*)
event beginUEA(A,B, pkuea, rand_a, rand_b);
in (c, (=LookUp, m4:bitstring,hm4:bitstring)); (*msg6*)
let (ub:id, d1:bitstring, rand_a:nonce, rand_b:nonce, Ts_2:bitstring,
hm4:bitstring, hostB: host) = adecrypt(m4, skueb) in
let (pkub:spkey, =B)= checksign(hm4,pkub)in
new Ts_2:bitstring;
let m4 = (ub, d1, rand_a,rand_b, Ts_2) in
let hm4 = h(m4)in
let (pkueb: pkey, =B) = checksign(m4,pkub) in
new Ts_3:bitstring;
new Exp_1: bitstring;
```

```
let Acap = (ua,ub, dataname, rand_a, rand_b, Ts_3, Exp_1) in
new data:bitstring;
new Kd1:key;
new rand_1:nonce;
let Kd1 = SHA256(rand_1, d1) in
let dm = (Kd1, Acap)in
let hdm =(dm)in
let SendData = (data, rand_b, Kd1, Acap, sign(hdm, skua)) in
out (c, aencrypt(SendData, pkueb)); (*msg7*)
event endUEA(A, B, pkueb, rand_a, rand_b);
event end().

let processgNB(G:host, kgnba: key, kgnbb:key, A:host, B:host)=
in(c, (=AdvMsg, (m0:bitstring))); (*msg1*)
let (ua:id, rand_na:nonce, dataname:bitstring, d1:bitstring, pkuea:pkey,
hapkuea:bitstring, hosta: host) = sdecrypt(m0, kgnba) in
in(c, (=IntMsg, m1:bitstring)); (*msg2*)
let (ub:id, rand_nb:nonce, dataname:bitstring, pkueb:pkey, hpkueb:bitstring,
hostb: host) = sdecrypt(m1, kgnbb) in
let DiscMsg = (ub, dataname, pkueb) in
let LinkUpMsg = (ua, dataname, pkuea) in
out(c, sencrypt(DiscMsg, kgnba)); (*msg3*)
out(c, sencrypt(LinkUpMsg, kgnbb)); (*msg4*)
event end().

let processUEB(pkua:spkey, skueb:skey, skub:sskey, kgnbb:key) =
new ua:id;
new ub:id;
new dataname:bitstring;
let pkueb = pk(skueb) in
new rand_nb:nonce;
let hpkueb = hash1(pkueb)in
let IntMsg = (ub, rand_nb, dataname, pkueb, hpkueb) in
out(c, sencrypt(IntMsg, kgnbb)); (*msg2*)
in (c, (=LinkUpMsg, mx:bitstring)); (*msg4*)
let (ua:id, dataname:bitstring, pkuea:pkey, hostb: host) = sencrypt(mx, kgnbb) in
in (c, (=PublMsg, m3:bitstring, hm3:bitstring)); (*msg5*)
let (ua:id, dataname:bitstring, d1:bitstring,rand_a:nonce, pkuea:pkey,
Ts_1:bitstring, hostA: host) = adecrypt(m3, skueb) in
let m3= (ua, d1, rand_a, pkuea, Ts_1) in
let hm3 = h(m3) in
let (pkuea: pkey, =A) = checksign(hm3,pkua) in
new rand_a:nonce;
new rand_b:nonce;
new d1:bitstring;
let d1 = SHA256(rand_a, dataname)in
```

```
new Ts_2:bitstring;
let hLookup = h(LookUp)in
let LookUp = (ub, d1, rand_a, rand_b, Ts_2, sign(LookUp, skub)) in
out (c, aencrypt(LookUp, pkuea)); (*msg6*)
event beginUEB(A, B, pkueb, rand_a, rand_b);
in (c, (=SendData, (m5:bitstring, hm5:bitstring, pkueb:pkey))); (*msg7*)
let (data:bitstring, rand_b:nonce, Kd1:key, Acap:bitstring, hostA: host) =
adecrypt(m5, skueb) in
let (pkuea: pkey, =A) = checksign(hm5,pkua) in
new Exp_1: bitstring;
new Ts_3:bitstring;
let Acap = (ua,ub, dataname, rand_a, rand_b, Ts_3, Exp_1) in
new Data:bitstring;
let m5 = (data, rand_b, Kd1, Acap) in
let hm5= h(m5)in
event endUEB(A, B, pkueb, rand_a, rand_b);
event end().

let keyRegistration =
in(c, (h: host, k: pkey));
if h <> A && h <> B then insert keys(h,k).

process
new skuea: skey;
new skua: sskey;
new skueb: skey;
new skub: sskey;
new kgnba: key;
new kgnbb: key;
let pkuea = pk(skuea) in out(c, pkuea);
let pkua = spk(skua) in out(c, pkua);
let pkueb = pk(skueb) in out(c, pkueb);
let pkub = spk(skub) in out(c, pkub);
insert keys(A, pkuea);
insert keys(B, pkueb);

(!processUEA(pkub, skuea, skua, kgnba))|(!processUEB(pkua, skueb, skub, kgnbb))|
(!processgNB(G, kgnba, kgnbb, A, B))
```

Appendix G

G.1 DDACap Protocol

```
(*This file is part of a PhD ProVerif Simulation**Author: Ed Kamya Kiyemba Edris*)
(*Device-to-Device Attribute and Capability Security Protocol*) (* UEB-UEC*)
(*Mutual Authentication, Caching and Sharing Authorization*)
(*Without Network Assistance*)

free c: channel.

type host.
type nonce.
type key.
type pkey.
type skey.
type spkey.
type sskey.
type id.

(*Constant*)
const AdvMsg: bitstring [data].
const IntMsg: bitstring [data].
const PublMsg: bitstring [data].
const LookUp: bitstring [data].
const SendData: bitstring [data].

(*Functions*)
fun nonce_to_bitstring(nonce): bitstring [data, typeConverter].
fun bitstring_to_key(bitstring): key [data, typeConverter].
fun bitstring_to_id (bitstring): id [data, typeConverter].
(*Hash Functions*)
fun h(bitstring) : bitstring.
fun SHA256(nonce,bitstring) : bitstring.
fun hash(bitstring, bitstring) : bitstring.
fun hash1(pkey):bitstring.
(* Public key encryption *)
fun pk(skey): pkey.
fun aencrypt(bitstring, pkey): bitstring.
reduc forall x: bitstring, y: skey; adecrypt(aencrypt(x,pk(y)),y) = x.
(* Shared key encryption *)
fun sencrypt(bitstring,nonce): bitstring.
reduc forall x: bitstring, y: nonce; sdecrypt(sencrypt(x,y),y) = x.
(* Signatures *)
fun spk(sskey): spkey.
fun sign(bitstring, sskey): bitstring.
reduc forall x: bitstring, k: sskey; getmess(sign(x,k)) = x.
reduc forall x: bitstring, k: sskey; checksign(sign(x,k), spk(k)) = x.

(* events *)
event end().
event beginUEB(host, host, pkey, nonce, nonce).
event endUEB(host, host, pkey, nonce, nonce).
event beginUEC(host, host, pkey, nonce, nonce).
event endUEC(host, host, pkey, nonce, nonce).

(* Host names *)
free C, B: host.

(*Table*)
table keys(host, pkey).

(* Free names *)
free Secret: bitstring [private].
query attacker (Secret).
free rand_rc:nonce [private].
query attacker (rand_rc).
free rand_rb:nonce [private].
query attacker (rand_rb).
free skuec:skey [private].
query attacker (skuec).
free skueb:skey [private].
query attacker (skueb).
free ub:id [private].
query attacker (ub).
free uc:id [private].
query attacker (uc).

(* Queries *)
query C: host, B: host, K: pkey, rand_rc:nonce, rand_rb:nonce;
event(endUEC(C, B , K, rand_rc, rand_rb)) ==>
event(beginUEB(C, B, K, rand_rc, rand_rb)).
query C: host, B: host, K: pkey, rand_rb:nonce, rand_rc:nonce;
event(endUEB(C, B, K, rand_rb, rand_rc)) ==>
event(beginUEC(C, B, K, rand_rc, rand_rb)).
query C: host, B: host, K: pkey, rand_rc:nonce, rand_rb:nonce;
inj-event(endUEC(C, B, K, rand_rc, rand_rb)) ==>
inj-event(beginUEB(C, B, K, rand_rc, rand_rb)).
query C: host, B: host, K: pkey, rand_rc:nonce, rand_rb:nonce;
inj-event(endUEB(C, B, K, rand_rc, rand_rb)) ==>
inj-event(beginUEC(C, B, K, rand_rc, rand_rb)).

(*Processes*)
let ProcessUEB(pkuc:spkey, skueb:skey, skub:sskey) =
new ub:id;
new dataname:bitstring;
new d1:bitstring;
new uc:id;
let pkueb = pk(skueb) in
let hpkueb =hash1(pkueb) in
let d1 = SHA256(rand_rb, dataname) in
let AdvMsg = (ub, dataname, d1, pkueb, hpkueb) in
out(c, (AdvMsg)); (*msg1*)
in(c, (=IntMsg, m1:bitstring)); (*msg2*)
let (uc:id, dataname:bitstring, pkuec:pkey, hpkuec:bitstring, hostC: host) =
adecrypt(m1, skueb) in
let m1 = (uc, dataname, pkuec, hpkuec) in
let hpkuec =hash1(pkuec) in
new rand_rb:nonce;
new Ts_1:bitstring;
let hPublMsg= h(PublMsg)in
let PublMsg = (ub, d1, rand_rb, Ts_1, sign(hPublMsg, skub)) in
out(c, aencrypt(PublMsg, pkuec)); (*msg3*)
event beginUEB(C,B, pkueb, rand_rc, rand_rb);
in (c, (=LookUp, m3:bitstring,hm3:bitstring)); (*msg4*)
let (uc:id, d1:bitstring, rand_rb:nonce, rand_rc:nonce, Ts_2:bitstring,
hm3:bitstring, hostC: host) = adecrypt(m3, skueb) in
let (pkuc:spkey, =C)= checksign(hm3, pkuc)in
new Ts_2:bitstring;
new uec:id;
let m3 = (uc, d1, rand_rb,rand_rc, Ts_2) in
let hm3 = h(m3)in
new kd1:bitstring;
new Acap:bitstring;
new Ts_3:bitstring;
new Exp_1: bitstring;
let Acap = (uc,ub, dataname, rand_rc, rand_rb, Ts_3, Exp_1) in
let dm = (kd1, Acap) in
let hdm= h(dm)in
new data:bitstring;
let SendData = (data, kd1, Acap, sign(hdm, skub)) in
out (c, aencrypt(SendData, pkuec)); (*msg5*)
event endUEB(C, B, pkuec, rand_rc, rand_rb);
event end().

let ProcessUEC(pkub:spkey, skuec:skey, skuc:sskey) =
let pkuec = pk(skuec) in
in (c, (=AdvMsg, m0:bitstring, hm0:bitstring)); (*msg1*)
new pkueb:pkey;
new ub:id;
new dataname:bitstring;
new d1:bitstring;
new uc:id;
let hpkueb = hash1(pkueb) in
let hpkuec = hash1(pkuec) in
let m0= (ub, dataname, d1, pkueb, hpkueb) in
let IntMsg = (ub, dataname, pkuec, hpkuec) in
out(c, aencrypt(IntMsg, pkueb)); (*msg2*)
in (c, (=PublMsg, m2:bitstring, hm2:bitstring)); (*msg3*)
let (ub:id, d1:bitstring,rand_rb:nonce, hm2:bitstring, Ts_1:bitstring, hostB:
host) = adecrypt(m2, skuec) in
let (pkueb: pkey, =B) = checksign(hm2,pkub) in
new rand_rb:nonce;
new rand_rc:nonce;
let m2 = (ub, d1, rand_rb, Ts_1) in
let hm2= h(m2)in
new Ts_2:bitstring;
let hLookUp= h(LookUp)in
let LookUp = (uc, d1, rand_rb, rand_rc, Ts_2, sign(hLookUp, skuc), pkueb) in
out(c, aencrypt(LookUp, pkueb)); (*msg4*)
new Acap:bitstring;
new kd1: key;
new rand_1: nonce;
let kd1 = SHA256(rand_1, d1)in
event beginUEC(C, B, pkuec, rand_rc, rand_rb);
in (c, (=SendData, (m4:bitstring, hm4:bitstring, pkuec:pkey))); (*msg5*)
let (Data:bitstring, ChT_2:bitstring,ShT_2:bitstring, hostC: host) =
adecrypt(m4, skuec) in
let (pkueb: pkey, =B) = checksign(hm4,pkub) in
new Ts_3:bitstring;
new Exp_1: bitstring;
let Acap = (uc,ub, dataname, rand_rc, rand_rb, Ts_3, Exp_1) in
new Data:bitstring;
```

```
let m4 = (Data, kd1, Acap) in
let hm4= h(m4)in
event endUEC(C, B, pkuec, rand_rc, rand_rb);
event end().

let keyRegistration =
in(c, (h: host, k: pkey));
if h <> C && h <> B then insert keys(h,k).

process
new skuec: skey;
new skuc: sskey;
new skueb: skey;
new skub: sskey;
let pkuec = pk(skuec) in out(c, pkuec);
let pkuc = spk(skuc) in out(c, pkuc);
let pkueb = pk(skueb) in out(c, pkueb);
let pkub = spk(skub) in out(c, pkub);
insert keys(C, pkuec);
insert keys(B, pkueb);

(!ProcessUEB(pkuc, skueb, skub))|(!ProcessUEC(pkub, skuec, skuc))
```

Appendix H

H.1 SAP-AKA Enhanced Labels

$l_1 = \langle \vartheta_{UE}(\{ServName, SID\}_{K_{AMF}}), \vartheta_{SMF}(a^{UE}_{enc})\rangle$ (msg1)
$l_2 = \langle \vartheta'_{SMF}(b^{UE}_{ServName,SID})\rangle$
$l_3 = \langle \vartheta_{SMF}(\{GPSI, SPID, PK_{SP}\}_{K_{AMF}}), \vartheta'_{UE}(c^{SMF}_{enc})\rangle$ (msg2)
$l_4 = \langle \vartheta'_{UE}(d^{SMF}_{GPSI,SPID,PK_{SP}})\rangle$
$l_5 = \langle \vartheta_{UE}(\{ServName, SID\}_{PK_{SP}}), \vartheta_{SPAAA}(e^{UE}_{enc}\rangle$ (msg3)
$l_6 = \langle \vartheta'_{SPAAA}(f^{UE}_{ServName,SID})\rangle$
$l_7 = \langle \vartheta_{SPAAA}(\{Identity\}_{PK_{UE}}), \vartheta_{UE}(g^{SPAAA}_{enc})\rangle$ (msg4)
$l_8 = \langle \vartheta'_{UE}(h^{SPAAA}_{Identity})\rangle$
$l_9 = \langle \vartheta_{UE}(\{GPSI\}_{PK_{SP}})), \vartheta_{SPAAA}(i^{UE}_{enc})\rangle$ (msg5)
$l_{10} = \langle \vartheta'_{SPAAA}(j^{UE}_{GPSI})\rangle$
$l_{11} = \langle \vartheta_{SPAAA}(\{AT\_RAND, AT\_AUTN, AT\_MAC, AT\_KDF, AT\_KDF\_INPUT\}_{K_{ENC}}),$
$\vartheta_{UE}(k^{SPAAA}_{enc})\rangle$ (msg6)
$l_{12} = \langle \vartheta'_{UE}(l^{SPAAA}_{AT\_RAND,AT\_AUTN,AT\_MAC,AT\_KDF,AT\_KDF\_INPUT})\rangle$
$l_{13} = \langle \vartheta_{UE}(\{AT\_RES, AT\_MAC2\}_{K_{ENC}}), \vartheta_{SPAAA}(m^{UE}_{enc})\rangle$ (msg7)
$l_{14} = \langle \vartheta'_{SPAAA}(n^{UE}_{enc})\rangle$
$l_{15} = \langle \vartheta_{SPAAA}(\{SUCCESS, K_{UE3A}, EID\}_{K_{ENC}}), \vartheta'_{UE}(o^{SPAAA}_{enc})\rangle$ (msg8)
$l_{16} = \langle \vartheta'_{UE}(p^{SPAAA}_{SUCCESS,K_{UE3A},EID})\rangle$
$l_{17} = \langle \vartheta_{SPAAA}(SUCCESS), \vartheta_{SMF}(r_{EID})\rangle$ (msg9)

H.2 NS-FId Enhanced Labels

$l_1 = \langle \vartheta_{UE}(\{ServName, SID\}K_{AMF}), \vartheta_{SMF}(s^{UE}_{enc})\rangle$ (msg1),
$l_2 = \langle \vartheta'_{SMF}(d^{UE}_{ServName,SID})\rangle$
$l_3 = \langle \vartheta_{SMF}(\{GPSI, SPID\}K_{AMF}), \vartheta_{UE}(t^{SMF}_{enc})\rangle$ (msg2)
$l_4 = \langle \vartheta'_{UE}(u^{SMF}_{GPSI,SPID})\rangle$
$l_5 = \langle \vartheta_{UE}(\{ServName, SID, SPID\}_{PK_{SP}}), \vartheta_{SPAAA}(v^{UE}_{enc}\rangle$ (msg3)
$l_6 = \langle \vartheta'_{SPAAA}(x^{UE}_{ServName,SID,SPID})\rangle$
$l_7 = \langle \vartheta_{SPAAA}(\{AuthzGrant, EID, K_{UE3A}\}_{PK_{UE}}), \vartheta_{UE}(y^{SPAAA}_{enc})\rangle$ (msg4)
$l_8 = \langle \vartheta'_{UE}(z^{SPAAA}_{AuthzGrant,EID,K_{UE3A}})\rangle$

$l_9 = \langle \vartheta_{UE}(\{AuthzGrant, R1, GPSI\}_{PK_{IDP}})), \vartheta_{IDP}(a^{UE}_{enc})\rangle$ (msg5)
$l_{10} = \langle \vartheta'_{IDP}(b^{UE}_{AuthzGrant,R1,GPSI})\rangle$
$l_{11} = \langle \vartheta_{IDP}(\{FID, IDT, (hash(IDT), SK_{IDP}), R1, (hash(IDT, (hash(IDT),$
$SK_{IDP}), R1), SK_{IDP})\}_{PK_{UE}}), \vartheta_{UE}(c^{IDP}_{enc})\rangle$ (msg6)
$l_{12} = \langle \vartheta'_{UE}(d^{IDP}_{FID,IDT,(hash(IDT),SK_{IDP}),R1,(hash(IDT,(hash(IDT),SK_{IDP}),R1),SK_{IDP}})\rangle$
$l_{13} = \langle \vartheta_{UE}(\{IDT, (hash(IDT), SK_{IDP})\}_{K_{UE3A}}), \vartheta_{SPAAA}(e^{UE}_{enc})\rangle$ (msg7)
$l_{14} = \langle \vartheta'_{SPAAA}(f^{UE}_{IDT,(hash(IDT),SK_{IDP})})\rangle$
$l_{15} = \langle \vartheta_{SPAAA}(\{(AcT, (hash(AcT), SK_{AAA}), K_{UESS}, (hash(AcT,$
$(hash(AcT), SK_{AAA}), K_{UESS}), SK_{AAA})\}_{K_{UE3A}}), \vartheta_{UE}(g^{SPAAA}_{enc})\rangle$ (msg8)
$l_{16} = \langle \vartheta'_{UE}(h^{SPAAA}_{(AcT,(hash(AcT),SK_{AAA}),K_{UESS},(hash(AcT,(hash(AcT),SK_{AAA}),K_{UESS}),SK_{AAA})})\rangle$
$l_{17} = \langle \vartheta_{UE}(\{AcT, (hash(AcT), SK_{AAA})\}_{K_{UESS}}), \vartheta_{SS}(i^{UE}_{enc})\rangle$ (msg9)
$l_{18} = \langle \vartheta'_{SS}(j^{UE}_{AcT,(hash(AcT),SK_{AAA})})\rangle$
$l_{19} = \langle \vartheta_{SS}(\{SERV, (hash(SERV), SK_{SS})\}_{K_{UESS}}), \vartheta_{UE}(k^{SS}_{enc})\rangle$ (msg10)
$l_{20} = \langle \vartheta'_{UE}(l^{SS}_{SERV,(hash(SERV),SK_{SS})})\rangle$

H.3 DCSS Enhanced Labels

$l_1 = \langle \vartheta_{UE}(\{FID, DATANAME, AcT, (hash(AcT), SK_{AAA})\}_{K_{UE3A}}),$
$\vartheta_{SPAAA}(r^{UE}_{enc})\rangle$ (msg1),
$l_2 = \langle \vartheta\prime_{SPAAA}(s^{UE}_{FID,DATANAME,AcT,(hash(AcT),SK_{AAA}})\rangle$
$l_3 = \langle \vartheta_{SPAAA}(\{ChT, (hash(AcT), SK_{AAA})\}_{K_{UE3A}}), \vartheta_{UE}(t^{SPAAA}_{enc}\rangle$ (msg2)
$l_4 = \langle \vartheta\prime_{UE}(u^{SPAAA}_{ChT,(hash(AcT),SK_{AAA})})\rangle$
$l_5 = \langle \vartheta_{UE}(\{ChT, (hash(ChT), SK_{AAA})\}_{K_{UESS}}), \vartheta_{SS}(v^{SPAAA}_{enc})\rangle$ (msg3)
$l_6 = \langle \vartheta'_{SS}(w^{UE}_{ChT,(hash(ChT),SK_{AAA})})\rangle$
$l_7 = \langle \vartheta_{SS}(\{Ack_1, (hash(Ack_1), SK_{AAA})\}_{K_{UESS}}), \vartheta_{UE}(x^{SS}_{enc})\rangle$ (msg4)
$l_8 = \langle \vartheta'_{UE}(y^{SS}_{Ack_1,(hash(Ack_1),SK_{AAA})})\rangle$
$l_9 = \langle \vartheta_{UE}(\{FID, DATANAME, Ack_1, (hash(Ack_1), SK_{AAA})\}_{K_{UE3A}}), \vartheta_{SPAAA}(z^{UE}_{enc})\rangle$ (msg5)
$l_{10} = \langle \vartheta'_{SPAAA}(a^{UE}_{FID,DATANAME,Ack_1,(hash(Ack_1),SK_{AAA})})\rangle$
$l_{11} = \langle \vartheta_{SPAAA}(\{ChT, (hash(AcT), SK_{AAA})\}_{K_{UE3A}}), \vartheta_{UE}(b^{SPAAA}_{enc})\rangle$ (msg6)
$l_{12} = \langle \vartheta'_{UE}(c^{SPAAA}_{ChT,(hash(AcT),SK_{AAA})})\rangle$
$l_{13} = \langle \vartheta_{UE}(\{ShT, (hash(ShT), SK_{AAA})\}_{K_{UESS}}), \vartheta_{SS}(d^{UE}_{enc})\rangle$ (msg7)
$l_{14} = \langle \vartheta'_{SS}(e^{UE}_{ShT,(hash(ShT),SK_{AAA})})\rangle$
$l_{15} = \langle \vartheta_{SS}(\{Ack_2, (hash(Ack_2), SK_{AAA})\}_{K_{UESS}}), \vartheta_{UE}(f^{SS}_{enc})\rangle$ (msg8)
$l_{16} = \langle \vartheta'_{UE}(g^{SS}_{Ack_2,(hash(Ack_2),SK_{AAA})})\rangle$

H.4 DDSec Enhanced Labels

$l_1 = \langle \vartheta_{UEB}(\{UEB, DATANAME\}_{K_{gNBb}}), \vartheta_{gNB}(x^{UEB}_{enc})\rangle$ (msg1),
$l_2 = \langle \vartheta'_{gNB}(v^{UEB}_{UEB,DATANAME})\rangle$
$l_3 = \langle \vartheta_{gNB}(\{UEB, DATANAME\}_{K_{gNBa}}), \vartheta_{UEA}(x^{UEB}_{enc})\rangle$ (msg2),
$l_4 = \langle \vartheta'_{UEA}(v^{gNB}_{UEB,DATANAME})\rangle$

$l_5 = \langle \vartheta_{UEA}(\{UEA, UA, d1, Rand_a, (hash(UEA, UA, d1, Rand_a), SK_{UEA})\}_{PK_{UEB}}),$
$\vartheta'_{UEB}(s^{UEA}_{enc})\rangle$ (msg3)
$l_6 = \langle \vartheta'_{UEB}(t^{UEA}_{UEA,UA,d1,Rand_a,(hash(UEA,UA,d1,Rand_a),SK_{UEA})})\rangle$
$l_7 = \langle \vartheta_{UEB}(\{UEB, UB, d1, Rand_a, Rand_b, Ts_6, (hash(UEB, UA, d1, Rand_a, Rand_b, Ts_6),$
$SK_{UEB})\}_{PK_{UEA}}), \vartheta\prime_{UEA}(u^{UEB}_{enc}\rangle$ (msg4)
$l_8 = \langle \vartheta'_{UEA}(v^{UEB}_{UEB,UB,d1,Rand_a,Rand_b,Ts_6,(hash(UEB,UA,d1,Rand_a,Rand_b,Ts_6),SK_{UEB}})\rangle$
$l_9 = \langle \vartheta_{UEA}(\{Ts_7, Rand_b, (hash(Ts_7, Rand_2), SK_{UEA})\}_{PK_{UEB}}), \vartheta'_{UEB}(w^{UEA}_{enc})\rangle$ (msg5)
$l_{10} = \langle \vartheta'_{UEB}(x^{UEA}_{Ts_7,Rand_b,(hash(Ts_7,Rand_2),SK_{UEA})})\rangle$
$l_{11} = \langle \vartheta_{UEB}(\{UA, d1, Ts_8, (hash(UA, d1, Ts_8), SK_{UEB})\}_{PK_{UEA}})), \vartheta_{UEA}(y^{UE}_{enc})\rangle$ (msg6)
$l_{12} = \langle \vartheta'_{UEA}(z^{UEB}_{UA,d1,Ts_8,(hash(UA,d1,Ts_8),SK_{UEB})})\rangle$
$l_{13} = \langle \vartheta_{UEA}(\{DATA, Kd1, ACap, (hash(DATA, Kd1, ACap), SK_{UEA})\}_{PK_{UEB}}), \vartheta_{UEB}(a^{UEA}_{enc})\rangle$
(msg7)
$l_{14} = \langle \vartheta'_{UEB}(b^{UEA}_{DATA,Kd1,Acap,(hash(DATA,Kd1,Acap),SK_{UEA})})\rangle$

H.5 DDACap Enhanced Labels

$l_1 = \langle \vartheta_{UEC}(UEC, DATANAME, PK_{UEC}), \vartheta_{Br}(p_{UEC,DATANAME,PK_{UEC}})\rangle$ (msg1),
$l_2 = \langle \vartheta'_{UEB}(\{UEB, UB, d1, Rand\_rb, PK_{UEB}, (hash(UB, d1, Rand\_rb, PK_{UEB}),$
$SK_{UEB})\}_{PK_{UEC}}), \vartheta'_{UEC}(s^{UEB}_{enc})\rangle$ (msg2)
$l_3 = \langle \vartheta_{UEC}(t^{UEB}_{UEB,UB,d1,Rand\_rb,PK_{UEB},(hash(UB,d1,Rand\_rb,PK_{UEB}),SK_{UEB})})\rangle$
$l_4 = \langle \vartheta'_{UEC}(\{UC, d1, Rand\_rb, Rand\_rc, Ts_6, (hash(UEC, UC, d1, Rand\_rb, Rand\_rc, Ts_6),$
$SK_{UEC})\}_{PK_{UEB}}), \vartheta'_{UEB}(u^{UEC}_{enc}\rangle$ (msg3)
$l_5 = \langle \vartheta_{UEB}(v^{UEC}_{UC,d1,Rand_{rb},Rand_{rc},Ts_6,(hash(UEC,UC,d1,Rand_rb,Rand_{rc},Ts_6),SK_{UEC})})\rangle$
$l_6 = \langle \vartheta'_{UEB}(\{Ts_7, Rand\_rc, (hash(Ts_7, Rand\_rc), SK_{UEB})\}_{PK_{UEC}}),$
$\vartheta'_{UEC}(w^{UEB}_{enc})\rangle$ (msg4)
$l_7 = \langle \vartheta_{UEC}(x^{UEB}_{Ts_7,Rand\_rc,(hash(Ts_7,Rand\_rc),SK_{UEB})})\rangle$
$l_8 = \langle \vartheta'_{UEC}(\{UC, d1, Ts_8, (hash(UC, d1, Ts_8), SK_{UEC})\}_{PK_{UEB}})), \vartheta'_{UEB}(y^{UEC}_{enc})\rangle$ (msg5)
$l_9 = \langle \vartheta_{UEB}(z^{UEC}_{UC,d1,Ts_8,(hash(UC,d1,T_8)),SK_{UEC})})\rangle$
$l_{10} = \langle \vartheta'_{UEB}(\{DATA, Acap, Kd1, (hash(DATA, ACap, Kd1), SK_{UEB})\}_{PK_{UEC}}), \vartheta'_{UEC}(a^{UEB}_{enc})\rangle$
(msg6)
$l_{11} = \langle \vartheta_{UEC}(b^{UEB}_{DATA,ACap,Kd1,(hash(DATA,ACap,Kd1),SK_{UEB})})\rangle$

Appendix I

The graphical presentation of the square matrix $\mathcal{Q}$ of the CTMC of process ($CTMC(P)$) for each protocol.

I.1 NS-FId Matrix

$$
\mathbf{Q2} = \begin{array}{c|cccccccccccccccccccc|}
 & \scriptstyle l1 & \scriptstyle l2 & \scriptstyle l3 & \scriptstyle l4 & \scriptstyle l5 & \scriptstyle l6 & \scriptstyle l7 & \scriptstyle l8 & \scriptstyle l9 & \scriptstyle l10 & \scriptstyle l11 & \scriptstyle l12 & \scriptstyle l13 & \scriptstyle l14 & \scriptstyle l15 & \scriptstyle l16 & \scriptstyle l17 & \scriptstyle l18 & \scriptstyle l19 & \scriptstyle l20 \\
\scriptstyle l1 & -b & b & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\scriptstyle l2 & 0 & -2d & 2d & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\scriptstyle l3 & 0 & 0 & -b & b & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\scriptstyle l4 & 0 & 0 & 0 & -2d & 2d & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\scriptstyle l5 & 0 & 0 & 0 & 0 & -c & c & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\scriptstyle l6 & 0 & 0 & 0 & 0 & 0 & -3d & 3d & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\scriptstyle l7 & 0 & 0 & 0 & 0 & 0 & 0 & -c & c & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\scriptstyle l8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -3d & 3d & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\scriptstyle l9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -c & c & 0. & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\scriptstyle l10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -3d & 3d & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\scriptstyle l11 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -g & g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\scriptstyle l12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -5d & 5d & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\scriptstyle l13 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -b & b & 0 & 0 & 0 & 0 & 0 & 0 \\
\scriptstyle l14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2d & 2d & 0 & 0 & 0 & 0 & 0 \\
\scriptstyle l15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -f & f & 0 & 0 & 0 & 0 \\
\scriptstyle l16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -4d & 4d & 0 & 0 & 0 \\
\scriptstyle l17 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -b & b & 0 & 0 \\
\scriptstyle l18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2d & 2d & 0 \\
\scriptstyle l19 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -b & b \\
\scriptstyle l20 & 2d & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2d
\end{array} \tag{I.1}
$$

I.2 DCSS Matrix

$$\mathbf{Q3} = \begin{array}{c|cccccccccccccccc|} & l1 & l2 & l3 & l4 & l5 & l6 & l7 & l8 & l9 & l10 & l11 & l12 & l13 & l14 & l15 & l16 \\ \hline l1 & -f & f & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ l2 & 0 & -4d & 4d & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ l3 & 0 & 0 & -b & b & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ l4 & 0 & 0 & 0 & -2d & 2d & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ l5 & 0 & 0 & 0 & 0 & b & b & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ l6 & 0 & 0 & 0 & 0 & 0 & -2d & 2d & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ l7 & 0 & 0 & 0 & 0 & 0 & 0 & -b & b & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ l8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2d & 2d & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ l9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -f & f & 0. & 0 & 0 & 0 & 0 & 0 \\ l10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -4d & 4d & 0 & 0 & 0 & 0 & 0 \\ l11 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -b & b & 0 & 0 & 0 & 0 \\ l12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2d & 2d & 0 & 0 & 0 \\ l13 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -b & b & 0 & 0 \\ l14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2d & 2d & 0 \\ l15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -b & b \\ l16 & 2d & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2d \end{array} \tag{I.2}$$

I.3 DDSec Matrix

$$\mathbf{Q4} = \begin{array}{c|cccccccccccccc|} & l1 & l2 & l3 & l4 & l5 & l6 & l7 & l8 & l9 & l10 & l11 & l12 & l13 & l14 \\ \hline l1 & -b & b & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ l2 & 0 & -2d & 2d & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ l3 & 0 & 0 & -b & b & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ l4 & 0 & 0 & 0 & -2d & 2d & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ l5 & 0 & 0 & 0 & 0 & -g & g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ l6 & 0 & 0 & 0 & 0 & 0 & -5d & 5d & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ l7 & 0 & 0 & 0 & 0 & 0 & 0 & -i & i & 0 & 0 & 0 & 0 & 0 & 0 \\ l8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -7d & 7d & 0 & 0 & 0 & 0 & 0 \\ l9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -c & c & 0 & 0 & 0 & 0 \\ l10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -3d & 3d & 0 & 0 & 0 \\ l11 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -f & f & 0 & 0 \\ l12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -4d & 4d & 0 \\ l13 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -f & f \\ l14 & 4d & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -4d \end{array} \tag{I.3}$$

I.4 DDACap Matrix

$$\mathbf{Q5} = \begin{array}{c|ccccccccccc|} & l1 & l2 & l3 & l4 & l5 & l6 & l7 & l8 & l9 & l10 & l11 \\ l1 & -3s & 3s & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ l2 & 0 & -h & h & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ l3 & 0 & 0 & -6d & 6d & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ l4 & 0 & 0 & 0 & -h & h & 0 & 0 & 0 & 0 & 0 & 0 \\ l5 & 0 & 0 & 0 & 0 & -6d & 6d & 0 & 0 & 0 & 0 & 0 \\ l6 & 0 & 0 & 0 & 0 & 0 & -c & c & 0 & 0 & 0 & 0 \\ l7 & 0 & 0 & 0 & 0 & 0 & 0 & -3d & 3d & 0 & 0 & 0 \\ l8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -f & f & 0 & 0 \\ l9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -4d & 4d & 0 \\ l10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -f & f \\ l11 & 4d & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -4d \end{array} \tag{I.4}$$

Appendix J

J.1 5G-AKA Attack Derivation and Trace

```
Starting query event(endSEAF(x1_80)) ==> event(beginSEAF(x1_80))
goal reachable: attacker(x1_4218) -> end(endSEAF(x1_4218))

1. The attacker has some term suci_4228.
attacker(suci_4228).
3. The attacker has some term rand_4227.
attacker(rand_4227).
The event endSEAF(a) is executed.
4. By 3, the attacker may know rand_4227.
By 2, the attacker may know x1_4230.
Using the function SHA256 the attacker may obtain SHA256(rand_4227,x1_4230).
attacker(SHA256(rand_4227,x1_4230)).

TRACE 1
event endSEAF(a) at {57} in copy a_4234 (goal)
The event endSEAF(a) is executed.
A trace has been found.
RESULT event(endSEAF(x1_80)) ==> event(beginSEAF(x1_80)) is false.

TRACE 2
event endSEAF(a_5801) at {57} in copy a_5800 (goal)
The event endSEAF(a_5801) is executed in session a_5800.
A trace has been found.
RESULT inj-event(endSEAF(x1_84)) ==> (inj-event(beginSEAF(x1_84)) &&
(inj-event(e3(x1_84)) ==> (inj-event(e2(x1_84)) ==> inj-event(e1(x1_84,x2_85)))))
is false.
```

J.2 SAP-AKA Attack Derivation and Trace

```
Starting query event(endAAA(u_121,a_122,r,k_124)) ==>
```

```
event(beginUE(u_121,a_122,r,k_124))
goal reachable: attacker(gpsi_16360) && attacker(servname_16361)
&& attacker(sid_16362) && attacker(hostU_16363) ->
end(endAAA(hostU[],hostA[],at_rand_53[k_51 = kue3a_27[],u = hostU[]

1. The message pk(sksp[]) may be sent to the attacker at output {7}.
attacker(pk(sksp[])).
2. We assume as hypothesis that attacker(hostU_16401).

new sksp: skey creating sksp_16418 at {1}
new kue3a_27: key creating kue3a_16417 at {3}
out(c, ~M_16501) with ~M_16501 = pk(sksp_16418) at {7}

new at_any_id_45: bitstring creating at_any_id_16529 at {54} in copy a_16416
in(c, (ServReq,aencrypt((a_16413,a_16414,a_16415),~M_16501))) with
aencrypt((a_16413,a_16414,a_16415),~M_16501) =
aencrypt((a_16413,a_16414,a_16415),pk(sksp_16418)) at {55} in copy a_16416
out(c, (ReqId,~M_16589)) with ~M_16589 = at_any_id_16529 at {57} in copy a_16416
in(c, (ResId,a_16412,hostU)) at {58} in copy a_16416
get keys(hostU,hostA,kue3a_16417) at {92} in copy a_16416

Trace 1
A trace has been found.
RESULT event(endAAA(u_121,a_122,r,k_124)) ==>
event(beginUE(u_121,a_122,r,k_124)) is false.

Trace 2
A trace has been found.
RESULT inj-event(endAAA(u_130,a_131,r_132,k_134)) ==>
inj-event(beginUE(u_130,a_131,r_132,k_134)) is false.
```

J.3 NS-FId Attack Derivation Trace

```
Starting query event(endUE(U,I,K)) ==> event(beginIDP(U,I,K))
goal reachable: end(endUE(hostU[],hostI[],pk(skidp[])))

1. The attacker has some term A'_17951.
attacker(A'_17951).
2. The message A'_17951 that the attacker may have by 1 may be received at
input {25}.
So event endUE(hostU[],hostI[],pk(skidp[])) may be executed at {34}.
end(endUE(hostU[],hostI[],pk(skidp[]))).

new skue: skey creating skue_17958 at {1}
new sksidp: sskey creating sksidp_17964 at {8}
```

```
out(c, ~M) with ~M = pk(skue_17958) at {11}
in(c, a) at {25} in copy a_17956

TRACE 1
event endUE(hostU,hostI,pk(skidp_17957)) at {34} in copy a_17956 (goal)
The event endUE(hostU,hostI,pk(skidp_17957)) is executed.
A trace has been found.
RESULT event(endUE(U,I,K)) ==> event(beginIDP(U,I,K)) is false.

TRACE 2
event endUE(hostU,hostI,pk(skidp_22772)) at {34} in copy a_22770 (goal)
The event endUE(hostU,hostI,pk(skidp_22772)) is executed in session a_22770.
A trace has been found.
RESULT inj-event(endUE(U_119,I_120,K_121)) ==>
inj-event(beginIDP(U_119,I_120,K_121)) is false.
RESULT (even event(endUE(U_22760,I_22761,K_22762)) ==>
event(beginIDP(U_22760,I_22761,K_22762)) is false.)
```

J.4 DCSS Attack Derivation and Trace

```
Starting query event(endUE(U_110,SS_111,K_112)) ==>
event(beginSS(U_110,SS_111,K_112))
goal reachable: end(endUE(hostU[],hostSS[],kuess[]))

1. The attacker has some term A'_15173.
attacker(A'_15173).
2. The attacker has some term kue3a_15172.
attacker(kue3a_15172).
3. The attacker initially knows hostU[].
attacker(hostU[]).
Using the function 3-tuple the attacker may obtain (hostU[],hostU[],kue3a_15172).
attacker((hostU[],hostU[],kue3a_15172)).

out(c, ~M) with ~M = pk(skue_15181) at {6}
out(c, ~M_15346) with ~M_15346 = spk(skss_15182) at {8}
out(c, ~M_15427) with ~M_15427 = spk(skaaa_15183) at {10}
in(c, a_15179) at {12} in copy a_15180
in(c, (hostU,hostU,a)) at {124} in copy a_15178
get keys(hostU,hostSS,kuess) at {56} in copy a_15180

TRACE 1
event endUE(hostU,hostSS,kuess) at {25} in copy a_15180 (goal)
The event endUE(hostU,hostSS,kuess) is executed.
A trace has been found.
RESULT event(endUE(U_110,SS_111,K_112)) ==>
event(beginSS(U_110,SS_111,K_112)) is false.
```

```
TRACE 2:
event endUE(hostU,hostSS,kuess) at {25} in copy a_19431 (goal)
The event endUE(hostU,hostSS,kuess) is executed in session a_19431.
A trace has been found.
RESULT inj-event(endUE(U_116,SS_117,K_118)) ==>
inj-event(beginSS(U_116,SS_117,K_118)) is false.
```

J.5 DDSec Attack Derivation and Trace

```
TRACE 1:
Starting query not attacker(rand_a[])
goal reachable: attacker(rand_a[])

1. The attacker initially knows rand_a[].
attacker(rand_a[]).
new skuea_36: skey creating skuea_2040 at {1}
new skub: sskey creating skub_2043 at {4}
new kgnbb: key creating kgnbb_2045 at {6}
out(c, ~M) with ~M = pk(skuea_2040) at {8}
out(c, ~M_2190) with ~M_2190 = spk(skua_2041) at {10}
out(c, ~M_2262) with ~M_2262 = pk(skueb_2042) at {12}
insert keys(A,pk(skuea_2040)) at {15}
insert keys(B,pk(skueb_2042)) at {16}
The attacker has the message rand_a.

A trace has been found.
RESULT not attacker(rand_a[]) is false.

TRACE 2:
Starting query event(endUEB(A_107,B_108,K,rand_a_109,rand_b_110))
==> event(beginUEA(A_107,B_108,K,rand_a_109,rand_b_110))
goal reachable: end(endUEB(A[],B[],pk(skueb_37[]),rand_a[],
rand_b[]))

1. The attacker has some term hm3_7794.
attacker(hm3_7794).
2. The attacker has some term m3_7793.
attacker(m3_7793).
3. Using the function PublMsg the attacker may obtain PublMsg.
attacker(PublMsg).

Using the function 3-tuple the attacker may obtain (PublMsg,m3_7793,hm3_7794).
attacker((PublMsg,m3_7793,hm3_7794)).
5. The message (PublMsg,m3_7793,hm3_7794) that the attacker may have by 4 may be
received at input {63}.
```

```
new ua_66: id creating ua_8130 at {57} in copy a_7802
new ub_67: id creating ub_8131 at {58} in copy a_7802
new dataname_68: id creating dataname_8132 at {59} in copy a_7802
out(c, ~M_8189) with ~M_8189 = sencrypt((ub_8131,dataname_8132,
pk(skueb_7803)),kgnbb_7808) at {62} in copy a_7802
in(c, (PublMsg,a_7800,a_7801)) at {63} in copy a_7802

event endUEB(A,B,pk(skueb_7803),rand_a,rand_b) at {64} in copy a_7802 (goal)
The event endUEB(A,B,pk(skueb_7803),rand_a,rand_b) is executed.
A trace has been found.
RESULT event(endUEB(A_107,B_108,K,rand_a_109,rand_b_110)) ==>
event(beginUEA(A_107,B_108,K,rand_a_109,rand_b_110)) is false.

TRACE 3:
event endUEB(A,B,pk(skueb_9980),rand_a,rand_b) at {64}
in copy a_9977 (goal)
The event endUEB(A,B,pk(skueb_9980),rand_a,rand_b) is
executed in session a_9977.
A trace has been found.
RESULT inj-event(endUEB(A_116,B_117,K_118,rand_a_119,rand_b_120))
==>inj-event(beginUEA(A_116,B_117,K_118,rand_a_119,rand_b_120))
is false.
```

Appendix K

K.1 Protocols Transition

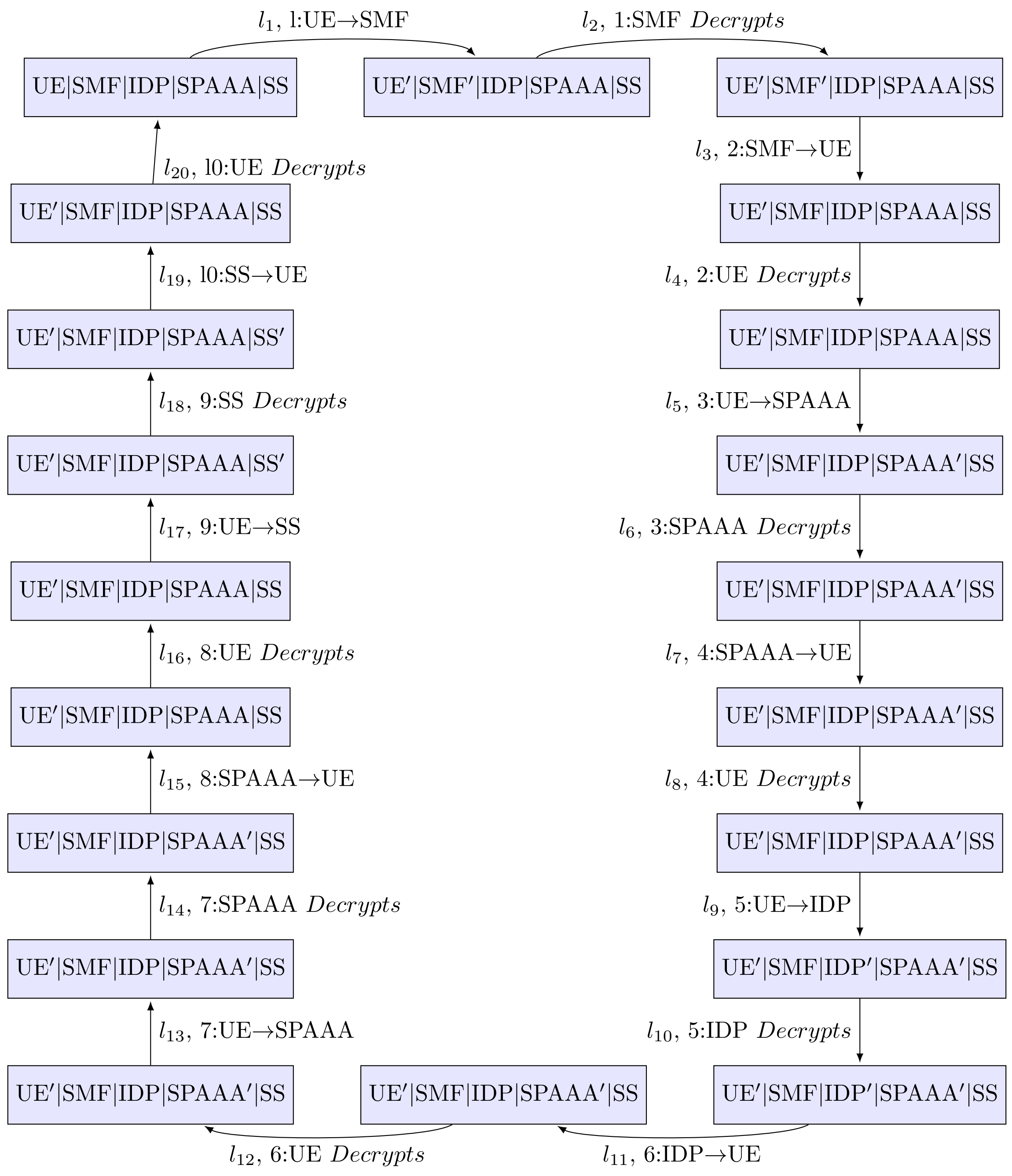


Figure K.1: NS-FId State Transition System

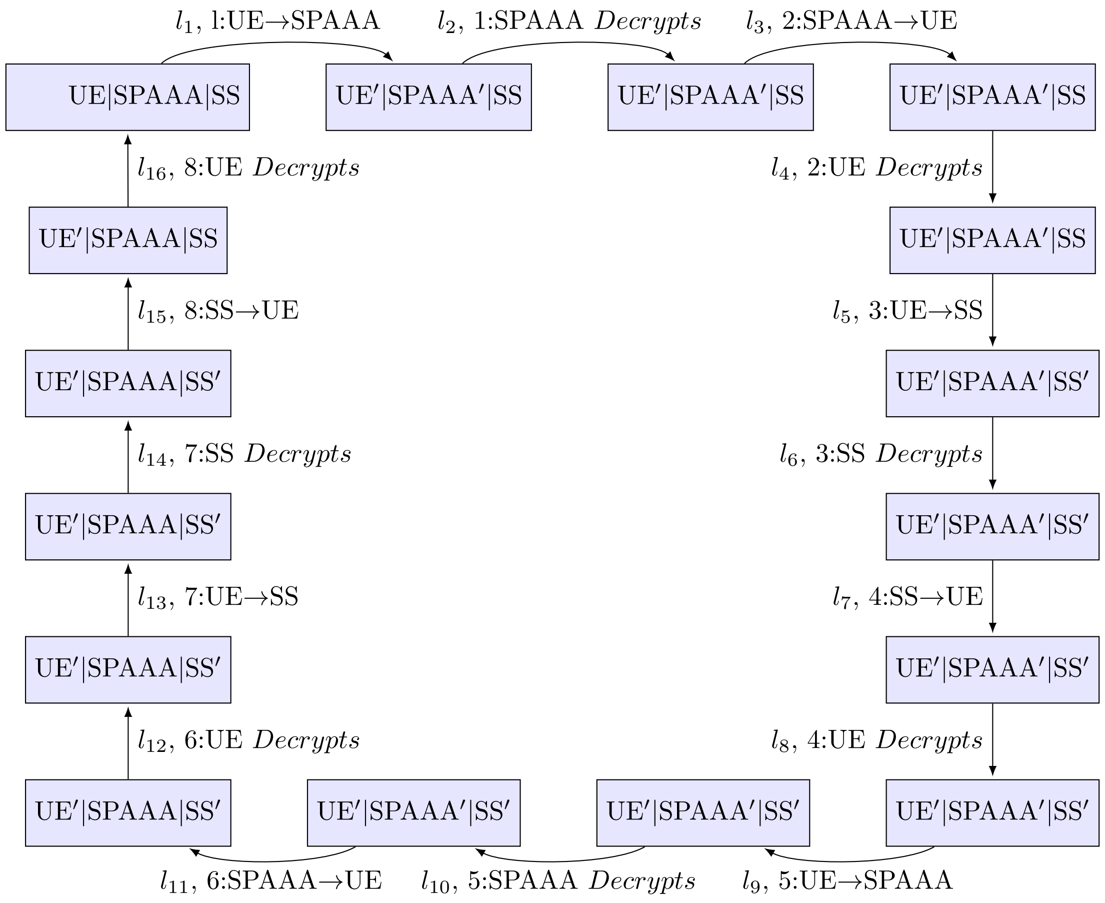


Figure K.2: DCSS State Transition System

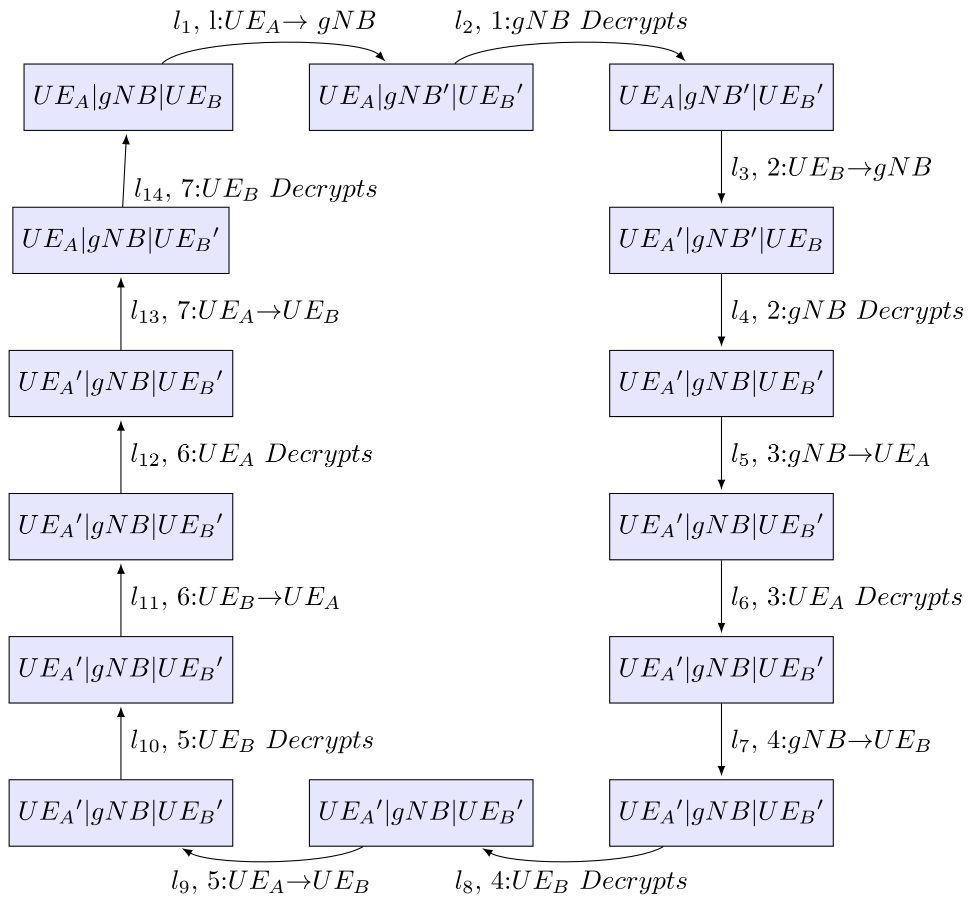

l_1, 1:$UE_A \rightarrow gNB$
l_2, 1:gNB $Decrypts$
$UE_A|gNB|UE_B$
$UE_A|gNB'|UE_B'$
$UE_A|gNB'|UE_B'$
l_{14}, 7:UE_B $Decrypts$
l_3, 2:$UE_B \rightarrow gNB$
$UE_A|gNB|UE_B'$
$UE_A'|gNB'|UE_B$
l_{13}, 7:$UE_A \rightarrow UE_B$
l_4, 2:gNB $Decrypts$
$UE_A'|gNB|UE_B'$
$UE_A'|gNB|UE_B'$
l_{12}, 6:UE_A $Decrypts$
l_5, 3:$gNB \rightarrow UE_A$
$UE_A'|gNB|UE_B'$
$UE_A'|gNB|UE_B'$
l_{11}, 6:$UE_B \rightarrow UE_A$
l_6, 3:UE_A $Decrypts$
$UE_A'|gNB|UE_B'$
$UE_A'|gNB|UE_B'$
l_{10}, 5:UE_B $Decrypts$
l_7, 4:$gNB \rightarrow UE_B$
$UE_A'|gNB|UE_B'$
$UE_A'|gNB|UE_B'$
$UE_A'|gNB|UE_B'$
l_9, 5:$UE_A \rightarrow UE_B$
l_8, 4:UE_B $Decrypts$

Figure K.3: DDSec State Transition System

$$\begin{tikzcd}
UE_B|UE_C \arrow[r, bend left, "{l_1,\ 1{:}UE_B \to UE_C}"] & UE_B'|UE_C' \arrow[r, bend left, "{l_2,\ 2{:}UE_C \to UE_B}"] & UE_B'|UE_C' \arrow[d, "{l_3,\ 2{:}UE_B\ \textit{Decrypts}}"] \\
 & & UE_B'|UE_C' \arrow[d, "{l_4,\ 3{:}UE_B \to UE_C}"] \\
UE_B|\mathrm{UEC}' \arrow[uu, "{l_9,\ 5{:}UE_C\ \textit{Decrypts}}"'] & & UE_B'|UE_C' \arrow[d, "{l_5,\ 3{:}UE_C\ \textit{Decrypts}}"] \\
UE_B'|UE_C' \arrow[u, "{l_8,\ 5{:}UE_B \to UE_C}"'] & UE_B'|UE_C' \arrow[l, bend left, "{l_7,\ 4{:}UE_B\ \textit{Decrypts}}"] & UE_B'|UE_C' \arrow[l, bend left, "{l_6,\ 4{:}UE_C \to UE_B}"]
\end{tikzcd}$$

Figure K.4: DDACap State Transition System

Appendix L

L.1 5G-AKA NS-3 Source Code

```
/*This file is part of a PhD NS-3 simulation experiment*/
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/* See GNU General Public License for more details.
*
* Author: Ed Kamya Kiyemba Edris
(Adapted from mmwave module.cc and authentictaion scheme)
* This program is distributed WITHOUT ANY WARRANTY.
*/
#include "ns3/core-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/mobility-module.h"
#include "ns3/internet-module.h"
#include "ns3/netanim-module.h"
#include "ns3/flow-monitor-helper.h"
#include "ns3/mmwave-helper.h"
#include "ns3/point-to-point-helper.h"
#include "ns3/config-store-module.h"
#include "ns3/command-line.h"
#include "ns3/mmwave-point-to-point-epc-helper.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/config-store.h"
#include "ns3/output-stream-wrapper.h"

using namespace ns3;
using namespace std;
using namespace mmwave;

static bool verbose = 0;
uint32_t M1 = 384, M2=448, M3 = 448, M4 = 1738, M5 = 928, M6=672, M7 = 256,
M8 = 256, M9 = 192, M10=576, M11 = 512, M12 = 640;

char * stringbuilder( char* prefix,  char* sufix){
char* buf = (char*)malloc(50);
snprintf(buf, 50, "%s%s", prefix, sufix);
return  buf;
}

ApplicationContainer sendMessage(ApplicationContainer apps, double time,
Ptr<Node>source,Ptr<Node>sink, uint32_t packetSize){
Ipv4Address remoteAddress =sink->GetObject<Ipv4> ()->GetAddress (1, 0).GetLocal();

uint16_t port = 9;  // well-known echo port number
uint32_t maxPacketCount = 1;
Time interPacketInterval = Seconds (20.);
UdpClientHelper client (remoteAddress, port);

client.SetAttribute ("MaxPackets", UintegerValue (maxPacketCount));
client.SetAttribute ("Interval", TimeValue (interPacketInterval));
client.SetAttribute ("PacketSize", UintegerValue (packetSize+12));
client.SetAttribute ("StartTime", TimeValue (Seconds (time)));
apps.Add(client.Install (source));
return apps;
}

ApplicationContainer authenticateA(ApplicationContainer appContainer, double time,
Ptr<Node> user, Ptr<Node> gateway ){

if (verbose){
std::cout<<"user:"<< user->GetObject<Ipv4> ()->GetAddress (1, 0).GetLocal ();
std::cout<<" gateway: "<< gateway->GetObject<Ipv4>()->GetAddress(1, 0).GetLocal();
}
return appContainer;
}

ApplicationContainer authenticateB(ApplicationContainer appContainer, double time,
Ptr<Node> user, Ptr<Node> gateway , Ptr<Node> gateway2 ){

if (verbose){
std::cout<<"user: "<< user->GetObject<Ipv4> ()->GetAddress(1, 0).GetLocal ();
std::cout<<" gateway:"<< gateway->GetObject<Ipv4> ()->GetAddress (1, 0).GetLocal();
std::cout<<"gateway2:"<< gateway2->GetObject<Ipv4> ()->GetAddress (1, 0).GetLocal
()<<std::endl;
}
return appContainer;
}
```

```
ApplicationContainer authenticateC(ApplicationContainer appContainer, double time,
Ptr<Node> UE, Ptr<Node> SEAF, Ptr<Node> AUSF, Ptr<Node> ARPF ){

if (verbose){
std::cout<<"UE:"<< UE->GetObject<Ipv4> ()->GetAddress(1, 0).GetLocal();
std::cout<<"SEAF: "<< SEAF->GetObject<Ipv4> ()->GetAddress(1, 0).GetLocal ();
std::cout<<" AUSF: "<< AUSF->GetObject<Ipv4> ()->GetAddress (1, 0).GetLocal
()<<std::endl;
std::cout<<"    ARPF : "<< ARPF->GetObject<Ipv4>()->GetAddress (1, 0).GetLocal
()<<std::endl;
}
uint32_t M1 = 384, M2=448, M3 = 448, M4=1738, M5 = 928, M6=672, M7=256, M8=256,
M9 = 192, M10=576, M11 = 512, M12=640;
appContainer = sendMessage(appContainer, time, UE, SEAF, M1);
appContainer = sendMessage(appContainer, time, SEAF, AUSF, M2);
appContainer = sendMessage(appContainer, time, AUSF, ARPF, M3);
appContainer = sendMessage(appContainer, time, ARPF, AUSF, M4);
appContainer = sendMessage(appContainer, time, AUSF, SEAF, M5);
appContainer = sendMessage(appContainer, time, SEAF, UE, M6);
appContainer = sendMessage(appContainer, time, UE, SEAF, M7);
appContainer = sendMessage(appContainer, time, SEAF, AUSF, M8);
appContainer = sendMessage(appContainer, time, AUSF, SEAF, M9);
appContainer = sendMessage(appContainer, time, UE, SEAF, M10);
appContainer = sendMessage(appContainer, time, SEAF, AUSF, M11);
appContainer = sendMessage(appContainer, time, AUSF, ARPF, M12);
return appContainer;
}
NS_LOG_COMPONENT_DEFINE ("mmwaveMultipleHosts");
int
main (int argc, char *argv[])
{
LogComponentEnable ("mmwaveMultipleHosts", LOG_LEVEL_ALL);
LogComponentEnable ("UdpClient", LOG_LEVEL_INFO);
LogComponentEnable ("PacketSink", LOG_LEVEL_INFO);
LogComponentEnable ("BulkSendApplication", LOG_LEVEL_INFO);

uint16_t serverNode1 = 1;
uint16_t serverNode2 = 1;
uint16_t numberOfNodes = 2;
double interPacketInterval = 100;
double minDistance = 10.0; // eNB-UE distance in meters
double maxDistance = 150.0; // eNB-UE distance in meters
bool harqEnabled = true;
bool rlcAmEnabled = false;
uint32_t stopTime = 2400;
bool verbose = 0;
bool enablePcap = 0;
bool enableAnim = 0;
bool verifyResults = 0; //used for regression
char saveFilePrefix[50] ;
bool tracing = true; // change to true if want pcap capture

// Command line arguments
CommandLine cmd;
cmd.AddValue("numberOfNodes", "Number of enbNodes + UE pairs", numberOfNodes);
cmd.AddValue
("interPacketInterval", "Inter-packet interval [us])", interPacketInterval);
cmd.AddValue ("harq", "Enable Hybrid ARQ", harqEnabled);
cmd.AddValue ("rlcAm", "Enable RLC-AM", rlcAmEnabled);
cmd.AddValue ("AUSF", "server node", serverNode1);
cmd.AddValue ("ARPF", "server node", serverNode2);
cmd.AddValue ("p", "Enable/disable pcap file generation", enablePcap);
cmd.AddValue ("a","Enable/disable xml gneration for netanim-module",enableAnim);
cmd.AddValue ("o", "Show output end of the siUElation", verifyResults);
cmd.AddValue ("v", "Verbose mode.", verbose);
cmd.AddValue ("s", "Define the prefix for .pcap anf .xml files. Default: 5GAKA ",
 saveFilePrefix);
cmd.AddValue ("tracing", "Enable pcap tracing", tracing);

cmd.Parse(argc, argv);

if (stopTime < 2)
{
std::cout << "Use a simulation stop time >= 2 seconds" << std::endl;
exit (1);
}
std::cout  <<  "UE="<< numberOfNodes <<", SEAF="<< numberOfNodes <<", AUSF="<<
serverNode1 << ", ARPF="<< serverNode2 <<std::endl;

if (verbose)
{
LogComponentEnable ("mmwaveMultipleHosts", LOG_LEVEL_ALL);
LogComponentEnable("UdpClient", LOG_LEVEL_INFO);
LogComponentEnable("UdpServer", LOG_LEVEL_INFO);
LogComponentEnable ("PacketSink", LOG_LEVEL_INFO);

Config::SetDefault ("ns3::MmWaveHelper::RlcAmEnabled",BooleanValue(rlcAmEnabled));
Config::SetDefault ("ns3::MmWaveHelper::HarqEnabled",BooleanValue (harqEnabled));
Config::SetDefault ("ns3::MmWaveFlexTtiMacScheduler::HarqEnabled", BooleanValue
(harqEnabled));
Config::SetDefault ("ns3::LteRlcAm::ReportBufferStatusTimer", TimeValue
(MicroSeconds (100.0)));
```

```
Config::SetDefault ("ns3::LteRlcUmLowLat::ReportBufferStatusTimer", TimeValue
(MicroSeconds (100.0)));

Ptr<MmWaveHelper> mmwaveHelper = CreateObject<MmWaveHelper> ();
mmwaveHelper->SetSchedulerType ("ns3::MmWaveFlexTtiMacScheduler");
Ptr<MmWavePointToPointEpcHelper>epcHelper = CreateObject
<MmWavePointToPointEpcHelper> ();
mmwaveHelper->SetEpcHelper (epcHelper);
mmwaveHelper->SetHarqEnabled (harqEnabled);

ConfigStore inputConfig;
inputConfig.ConfigureDefaults ();

// parse again so you can override default values from the command line
cmd.Parse(argc, argv);

// Create remote hosts and install Internet Stack
NodeContainer remoteHosts;
remoteHosts.Create (numberOfNodes);
InternetStackHelper internetStack;
internetStack.Install (remoteHosts);
NodeContainer ueNodes;
NodeContainer enbNodes;
enbNodes.Create(1);
ueNodes.Create(numberOfNodes);

//Create a router
Ptr<Node> router = CreateObject<Node> ();
internetStack.Install (router);

// Create the PGW
Ptr<Node> pgw = epcHelper->GetPgwNode ();

// Create p2p links
PointToPointHelper p2ph;
p2ph.SetDeviceAttribute ("DataRate", DataRateValue (DataRate ("100Gb/s")));
p2ph.SetDeviceAttribute ("Mtu", UintegerValue (1500));
p2ph.SetChannelAttribute ("Delay", TimeValue (Seconds (0.010)));

//Install link between PGW and Router
NetDeviceContainer pgwRouterDevices = p2ph.Install (pgw, router);

NetDeviceContainer remoteHostsDevices;
NetDeviceContainer routerDevices;
for (uint16_t u = 0; u < numberOfNodes; u++)
{
```

```
NetDeviceContainer c = p2ph.Install (router, remoteHosts.Get (u));
routerDevices.Add (c.Get(0));
remoteHostsDevices.Add (c.Get(1));
}

// Assigning Ipv4 Addresses
Ipv4InterfaceContainer routerInterfaces;
Ipv4InterfaceContainer remoteHostsInterfaces;
Ipv4AddressHelper pgwRouterIpv4 ("192.168.0.0", "255.255.0.0");
Ipv4InterfaceContainer pgwRouterInterfaces=pgwRouterIpv4.Assign(pgwRouterDevices);
Ipv4AddressHelper internetIpv4 ("1.0.0.0", "255.0.0.0");
for (uint16_t r = 0; r < remoteHosts.GetN(); ++r)
{
NetDeviceContainer ndc;
ndc.Add (remoteHostsDevices.Get (r));
ndc.Add (routerDevices.Get (r));
//ndc.Add (pgwDevices.Get (r));
Ipv4InterfaceContainer ifc = internetIpv4.Assign (ndc);
remoteHostsInterfaces.Add (ifc.Get (0));
routerInterfaces.Add (ifc.Get (1));
}
Ipv4StaticRoutingHelper ipv4RoutingHelper;
Ptr<OutputStreamWrapper> stream =Create<OutputStreamWrapper> (&std::clog);

uint16_t j = 2;
for (uint16_t u = 0; u < numberOfNodes; u++)
{
stringstream ss;
ss << u + j;
string addr = "1.0.0." + ss.str();
Ptr<Ipv4StaticRouting>remoteHostStaticRouting = ipv4RoutingHelper.GetStaticRouting
(remoteHosts.Get(u)->GetObject<Ipv4>());
remoteHostStaticRouting->AddNetworkRouteTo (Ipv4Address ("7.0.0.0"),Ipv4Mask
("255.0.0.0"), Ipv4Address(addr.c_str()), 1);
remoteHostStaticRouting->PrintRoutingTable(stream);
j++;
}

Ptr<Ipv4StaticRouting> routerStaticRouting = ipv4RoutingHelper.GetStaticRouting
(router->GetObject<Ipv4> ());
routerStaticRouting->AddNetworkRouteTo (Ipv4Address ("7.0.0.0"), Ipv4Mask
("255.0.0.0"), Ipv4Address("192.168.0.1"), 1);
routerStaticRouting->PrintRoutingTable(stream);
Ptr<Ipv4StaticRouting> pgwStaticRouting = ipv4RoutingHelper.GetStaticRouting
(pgw->GetObject<Ipv4> ());
pgwStaticRouting->AddNetworkRouteTo(Ipv4Address ("1.0.0.0"),Ipv4Mask("255.0.0.0"),
Ipv4Address("192.168.0.2"), 2);
pgwStaticRouting->PrintRoutingTable(stream);
Ptr<Ipv4> ipv4 = pgw->GetObject<Ipv4>();
uint32_t ifaces = ipv4->GetNInterfaces();
for (uint32_t u = 0; u < remoteHosts.GetN (); ++u)
{
ipv4 = remoteHosts.Get (u)->GetObject<Ipv4>();
ifaces = ipv4->GetNInterfaces();
for (uint32_t i = 0; i < ifaces; i++)
{
Ipv4InterfaceAddress iaddr = ipv4->GetAddress(i,0);
NS_LOG_INFO("AUSF"<< "\t iface:" <<i << "\tip:" <<iaddr.GetLocal());
NS_LOG_INFO("ARPF"<< "\t iface:" <<i << "\tip:" <<iaddr.GetLocal());
}
}
ipv4 = router->GetObject<Ipv4>();
ifaces = ipv4->GetNInterfaces();
for (uint32_t i = 0; i < ifaces; i++)
{
Ipv4InterfaceAddress iaddr = ipv4->GetAddress(i,0);
NS_LOG_INFO("Router \t iface:" << i << "\tip:" << iaddr.GetLocal());
}
ipv4 = pgw->GetObject<Ipv4>();
ifaces = ipv4->GetNInterfaces();
for (uint32_t i = 0; i < ifaces; i++)
{
Ipv4InterfaceAddress iaddr = ipv4->GetAddress(i,0);
NS_LOG_INFO("SEAF \t iface:" << i << "\tip:" << iaddr.GetLocal());
}
// Install Mobility Model
Ptr<ListPositionAllocator> enbPositionAlloc=CreateObject<ListPositionAllocator>();
enbPositionAlloc->Add (Vector (0.0, 0.0, 0.0));
MobilityHelper enbmobility;
enbmobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
enbmobility.SetPositionAllocator (enbPositionAlloc);
enbmobility.Install (enbNodes);
MobilityHelper uemobility;
Ptr<ListPositionAllocator> uePositionAlloc=CreateObject<ListPositionAllocator>();
Ptr<UniformRandomVariable> distRv=CreateObject<UniformRandomVariable>();
for (unsigned i = 0; i < numberOfNodes; i++)
{
double dist = distRv->GetValue (minDistance, maxDistance);
uePositionAlloc->Add (Vector (dist, 0.0, 0.0));
}
uemobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
uemobility.SetPositionAllocator (uePositionAlloc);
uemobility.Install (ueNodes);

// Install mmWave Devices to the nodes
NetDeviceContainer enbmmWaveDevs = mmwaveHelper->InstallEnbDevice (enbNodes);
NetDeviceContainer uemmWaveDevs = mmwaveHelper->InstallUeDevice (ueNodes);

// Install the IP stack on the UEs
internetStack.Install (ueNodes);
Ipv4InterfaceContainer ueIpIface;
ueIpIface = epcHelper->AssignUeIpv4Address (NetDeviceContainer (uemmWaveDevs));
for (uint32_t u = 0; u < ueNodes.GetN (); ++u)
{
Ptr<Ipv4StaticRouting> ueStaticRouting = ipv4RoutingHelper.GetStaticRouting
(ueNodes.Get (u)->GetObject<Ipv4> ());
ueStaticRouting->SetDefaultRoute (epcHelper->GetUeDefaultGatewayAddress (), 1);
ueStaticRouting->PrintRoutingTable(stream);
}
for (uint32_t u = 0; u < ueNodes.GetN (); ++u)
{
Ptr<Ipv4> ipv4 = ueNodes.Get (u)->GetObject<Ipv4>();
uint32_t ifaces = ipv4->GetNInterfaces();
for (uint32_t i = 0; i < ifaces; i++)
{
Ipv4InterfaceAddress iaddr = ipv4->GetAddress(i,0);
NS_LOG_INFO("UE " << "\t iface:" << i << "\tip:" << iaddr.GetLocal());
}
}
for (uint16_t i = 0; i < numberOfNodes; i++)
{
//mmwaveHelper->AttachToClosestEnb (uemmWaveDevs.Get(i), enbmmWaveDevs.Get(0));
mmwaveHelper->AttachToClosestEnb (uemmWaveDevs, enbmmWaveDevs);
}
uint16_t port = 9;
ApplicationContainer clientApps;
ApplicationContainer serverApps;
UdpServerHelper dlClientHelper(port);
for (uint16_t u = 0; u < remoteHosts.GetN (); ++u){
serverApps.Add(dlClientHelper .Install (remoteHosts.Get (u)));
}
for (uint16_t u = 0; u < remoteHosts.GetN (); ++u){
serverApps.Add(dlClientHelper .Install (remoteHosts.Get (u)));
}
for (uint16_t u = 0; u < enbNodes.GetN (); ++u){
serverApps.Add(dlClientHelper.Install (enbNodes.Get (u)));
}
double time = 1;
```

```
for (uint16_t i = 0; i < ueNodes.GetN (); ++i){
Ptr<Node> user = ueNodes.Get (i);

for (uint16_t u = i; u < serverNode1; u+=numberOfNodes
){
Ptr<Node> UE = ueNodes.Get (u);
Ptr<Node> ARPF = remoteHosts.Get (u);
Ptr<Node> SEAF = enbNodes.Get (u);
Ptr<Node> AUSF = remoteHosts.Get (u);

if(u == 0){
clientApps = authenticateA(clientApps, time , user, SEAF);
}else{
clientApps = authenticateB(clientApps, time , user, enbNodes.Get (0), SEAF);
}
clientApps = authenticateC(clientApps, time, UE, SEAF, ARPF, AUSF);
}
serverApps.Add( dlClientHelper.Install (user));
}
serverApps.Start (Seconds (0.0));
serverApps.Stop (Seconds (stopTime+1));

 //started induvugualy
clientApps.Stop (Seconds (stopTime+1));

std::cout <<"Setup Complete."<<std::endl;

if (verbose){
std::cout <<"servers stops at  "<<stopTime+1<<std::endl;
std::cout <<"final transmission  scheduled at  "<<(time-.33)<<std::endl;
std::cout << "server apps installed till now :"<<serverApps.GetN ()<< std::endl;
std::cout << "client apps installed till now :"<<clientApps.GetN ()<< std::endl;
}
snprintf(saveFilePrefix, 50, "5GAKA_%dx%dx%dx%d_", numberOfNodes, numberOfNodes,
serverNode1, serverNode2);

mmwaveHelper->EnableTraces ();
if (tracing == true)
{
p2ph.EnablePcap ("5gaka", uemmWaveDevs.Get (0));
p2ph.EnablePcap ("5gaka", enbmmWaveDevs.Get (0));
p2ph.EnablePcap ("5gaka", remoteHostsDevices.Get (0),true);
}
//Trace file
AsciiTraceHelper ascii;
p2ph.EnableAsciiAll(ascii.CreateFileStream("5gaka.tr"));
```

```
//Animation
if(enableAnim) {
AnimationInterface anim (stringbuilder(saveFilePrefix,(char*)"-animation.xml"));
// Mandatory
for (uint16_t i = 0; i < ueNodes.GetN (); ++i)
{
anim.UpdateNodeDescription (ueNodes.Get (i), "UE"); // Optional
anim.UpdateNodeColor (ueNodes.Get (i), 255, 0, 0); // Optional
}
for (uint16_t i = 0; i < remoteHosts.GetN (); ++i)
{
anim.UpdateNodeDescription (remoteHosts.Get (i), "ARPF"); // Optional
anim.UpdateNodeColor (remoteHosts.Get (i), 255, 255, 0); // Optional
}
for (uint16_t i = 0; i < enbNodes.GetN (); ++i)
{
anim.UpdateNodeDescription (enbNodes.Get (i), "Gateway"); // Optional
anim.UpdateNodeColor (enbNodes.Get (i), 0, 255, 0); // Optional
}
anim.EnablePacketMetadata (); // Optional/
anim.EnableWifiMacCounters (Seconds (0), Seconds (10)); //Optional
anim.EnableWifiPhyCounters (Seconds (0), Seconds (10)); //Optional
}
Ptr<FlowMonitor> flowMonitor;
FlowMonitorHelper flowHelper;
flowMonitor = flowHelper.InstallAll();

Simulator::Stop (Seconds (stopTime+1));
NS_LOG_INFO("Starting...");
Simulator::Run();
Simulator::Destroy();
flowMonitor->SerializeToXmlFile(stringbuilder(saveFilePrefix,(char*)"
_flowMonitor.xml"), true, true); //true, true

uint32_t bytes_received = 0, totalPacketsThrough;
for (uint32_t i = 0; i < serverApps.GetN(); ++i){
totalPacketsThrough=DynamicCast<UdpServer> (serverApps.Get(i))->GetReceived();;
bytes_received += totalPacketsThrough;
}
std::cout <<"Total packets received ("<< "UE="<< numberOfNodes <<", SEAF="<<
numberOfNodes <<", AUSF="<< serverNode1 << ", ARPF="<< serverNode2 << ") : "<<
bytes_received << std::endl;
NS_LOG_INFO("\ndone!");
return 0;
}
```

L.2 SAP-AKA NS-3 Code Excerpt

```
static bool verbose = 0;
uint32_t M1 = 224, M2=512, M3 = 352, M4 = 32, M5 = 64, M6=1104, M7 = 592, M8=224,
M9 = 32;

ApplicationContainer authenticateC(ApplicationContainer appContainer, double time,
Ptr<Node> UE, Ptr<Node> SMF, Ptr<Node> SPAAA, Ptr<Node> SS ){

if (verbose){
std::cout<<"UE: "<< UE->GetObject<Ipv4> ()->GetAddress (1, 0).GetLocal ();
std::cout<<"SMF: "<< SMF->GetObject<Ipv4> ()->GetAddress (1, 0).GetLocal ();
std::cout<<"  SPAAA: "<< SPAAA->GetObject<Ipv4> ()->GetAddress (1, 0).GetLocal
()<<std::endl;
std::cout<<" SS: "<< SS->GetObject<Ipv4> ()->GetAddress (1, 0).GetLocal
()<<std::endl;
}
uint32_t M1 = 224, M2=512, M3 = 352, M4 = 32, M5 = 64, M6=1104, M7 = 592, M8=224,
M9 = 32;
appContainer = sendMessage(appContainer, time, UE, SMF, M1);
appContainer = sendMessage(appContainer, time, SMF, UE, M2);
appContainer = sendMessage(appContainer, time, UE, SPAAA, M3);
appContainer = sendMessage(appContainer, time, SPAAA, UE, M4);
appContainer = sendMessage(appContainer, time, UE, SPAAA, M5);
appContainer = sendMessage(appContainer, time, SPAAA, UE, M6);
appContainer = sendMessage(appContainer, time, UE, SPAAA, M7);
appContainer = sendMessage(appContainer, time, SPAAA, UE, M8);
appContainer = sendMessage(appContainer, time, SPAAA, SMF, M9);
return appContainer;
}
int
main (int argc, char *argv[])
{
uint32_t numUe = 1;
uint32_t numEnb = 1;
uint32_t serverNodes = 4;

CommandLine cmd;
cmd.AddValue ("UE", "number of UEs", numUE);
cmd.AddValue ("SMF", " number of functions", numEnb);
cmd.AddValue ("SPAAA", "number of server nodes", serverNodes);
cmd.AddValue ("s", "Define the prefix for .pcap anf .xml files. Default:SAPAKA ",
saveFilePrefix);

snprintf(saveFilePrefix, 50, "SAPAKA_%dx%dx%d_", numUe, numEnb, serverNodes);
//Enable Pcap File
if (tracing == true)
{
phy.EnablePcap ("sapaka", uemmwaveDevs.Get (0));
phy.EnablePcap ("sapaka", enbmmwaveDevs.Get (0));
phy.EnablePcap ("sapaka", remoteHostsDevices.Get (0),true);
}
   //Creating trace file
AsciiTraceHelper ascii;
phy.EnableAsciiAll(ascii.CreateFileStream("sapaka.tr"));

std::cout <<"Total packets received ("<<    "UE="<< mnumUe <<", SMF="<< numEnb <<",
 SPAAA="<< serverNodes << ") : "<< bytes_received << std::endl;
return 0;
}
```

L.3 NS-FId NS-3 Code Excerpt

```
static bool verbose = 0;
uint32_t M1 = 224, M2=448, M3 = 416, M4 = 480, M5 = 480, M6= 1088, M7=512, M8=896,
M9 = 512, M10 = 416;

ApplicationContainer authenticateC(ApplicationContainer appContainer, double time,
Ptr<Node> UE, Ptr<Node> SMF, Ptr<Node> SPAAA, Ptr<Node> IDP, Ptr<Node> SS ){

if (verbose){
std::cout<<"UE: "<< UE->GetObject<Ipv4> ()->GetAddress (1, 0).GetLocal ();
std::cout<<"SMF: "<< SMF->GetObject<Ipv4> ()->GetAddress (1, 0).GetLocal ();
std::cout<<" IDP: "<< IDP->GetObject<Ipv4> ()->GetAddress (1, 0).GetLocal
()<<std::endl;
std::cout<<" SPAAA: "<< SPAAA->GetObject<Ipv4> ()->GetAddress (1, 0).GetLocal
()<<std::endl;
}
uint32_t  M1 = 224, M2=448, M3 = 416, M4=480, M5 = 480, M6= 1088, M7=512, M8=896,
M9 = 512, M10 = 416;
appContainer = sendMessage(appContainer, time, UE, SMF, M1);
appContainer = sendMessage(appContainer, time, SMF, UE, M2);
appContainer = sendMessage(appContainer, time, UE, SPAAA, M3);
appContainer = sendMessage(appContainer, time, SPAAA, UE, M4);
appContainer = sendMessage(appContainer, time, UE, IDP, M5);
appContainer = sendMessage(appContainer, time, IDP, UE, M6);
appContainer = sendMessage(appContainer, time, SPAAA, UE, M7);
appContainer = sendMessage(appContainer, time, SPAAA, UE, M8);
appContainer = sendMessage(appContainer, time, UE, SS, M9);
appContainer = sendMessage(appContainer, time, SS, UE, M10);
return appContainer;
}
int
main (int argc, char *argv[])
{
uint32_t numUe = 1;
uint32_t numEnb = 1;
uint32_t serverNodes = 3;

CommandLine cmd;
cmd.AddValue ("UE", "number of UE", numUe);
cmd.AddValue ("SMF", "number of functions", numEnb);
cmd.AddValue ("IDP", "number of servers nodes", serverNodes);
cmd.AddValue ("SPAAA", "number of server nodes", serverNodes);
cmd.AddValue ("SS", "number of server nodes", serverNodes);

std::cout  <<  "UE="<< numUe <<", SMF="<< numEnb <<", IDP="<< serverNodes << ",
SPAAA="<< serverNodes << ", SS="<< serverNodes <<std::endl;

snprintf(saveFilePrefix, 50, "NSFID_%dx%dx%d_", numUe, numEnb, serverNodes);

//Enable pcap file
if (tracing == true)
{
phy.EnablePcap ("nsfid", uemmwaveDevs.Get (0));
phy.EnablePcap ("nsfid", remoteHostDevs_SPAAA.Get (0));
phy.EnablePcap ("nsfid", remoteHostsDevices_IDP.Get (0));
phy.EnablePcap ("nsfid", remoteHostsDevicess_SS.Get (0));
phy.EnablePcap ("nsfid", enbmmwaveDevs.Get (0),true);
}
      //Trace file
AsciiTraceHelper ascii;
phy.EnableAsciiAll(ascii.CreateFileStream("nsfid.tr"));

std::cout <<"Total packets received ("<< "UE="<< numUe <<", SMF="<< numEnb <<",
IDP="<< serverNodes << ", SPAAA="<< serverNodes << ",SS="<< serverNodes << ") :
"<< bytes_received << std::endl;
return 0;
}
```

L.4 DCSS NS-3 Code Excerpt

```
static bool verbose = 0;
uint32_t M1 = 736, M2=576, M3 = 576, M4 = 480, M5=288, M6= 512, M7 = 512, M8=416;

ApplicationContainer authorize(ApplicationContainer appContainer, double time,
Ptr<Node> UE, Ptr<Node> SMF, Ptr<Node> SPAAA, Ptr<Node> SS ){
```

```
if (verbose){
std::cout<<"UE : "<< UE->GetObject<Ipv4> ()->GetAddress (1, 0).GetLocal ();
std::cout<<"SMF : "<< SMF->GetObject<Ipv4> ()->GetAddress (1, 0).GetLocal ();
std::cout<<"    SPAAA : "<< SPAAA->GetObject<Ipv4> ()->GetAddress (1, 0).GetLocal
()<<std::endl;
std::cout<<"    SS : "<< SS->GetObject<Ipv4> ()->GetAddress (1, 0).GetLocal
()<<std::endl;
}
uint32_t M1 = 736, M2=576, M3 = 576, M4=480, M5 = 288, M6= 512, M7 = 512, M8=416;
appContainer = sendMessage(appContainer, time, UE, SPAAA, M1);
appContainer = sendMessage(appContainer, time, SPAAA, UE, M2);
appContainer = sendMessage(appContainer, time, UE, SS, M3);
appContainer = sendMessage(appContainer, time, SS, UE, M4);
appContainer = sendMessage(appContainer, time, UE, SPAAA, M5);
appContainer = sendMessage(appContainer, time, SPAAA, UE, M6);
appContainer = sendMessage(appContainer, time, UE, SS, M7);
appContainer = sendMessage(appContainer, time, SS, UE, M8);
return appContainer;
}

int
main (int argc, char *argv[])
{
uint16_t numEnb = 1;
uint16_t numUe = 1;
uint32_t serverNode1 = 1;
uint32_t serverNode2 = 1;

// Command line arguments
CommandLine cmd;
cmd.AddValue ("SEAF", "Number of functions", numEnb);
cmd.AddValue ("UE", "Number of UEs", numUe);
cmd.AddValue ("SPAAA", "number of server nodes", remoteHostServer);
cmd.AddValue ("SS", "number of server nodes", remoteHostServer1);
cmd.AddValue ("s", "Define the prefix for .pcap anf .xml files. Default: DCSS ",
saveFilePrefix);

std::cout << "UE="<< numUe <<", SPAAA="<< serverNode1 << ", SS="<< serverNode2
<<std::endl;

// Enable PCAP tracing
if (tracing == true)
{
p2ph.EnablePcap ("dcss", uemmWaveDevs.Get (0));
p2ph.EnablePcap ("dcss", enbmmWaveDevs.Get (0));
p2ph.EnablePcap ("dcss", remoteHostsDevices.Get (0), true);
}

// Trace file
AsciiTraceHelper ascii;
p2ph.EnableAsciiAll(ascii.CreateFileStream("dcss.tr"));

std::cout <<"Total packets received ("<< "UE="<< numUe <<", SEAF="<< numEnb <<",
SPAAA="<< serverNode1<< ", SS="<< serverNode2<< "): "<< bytes_received<<std::endl;
return 0;
}
```

L.5 DDSec NS-3 Code Excerpt

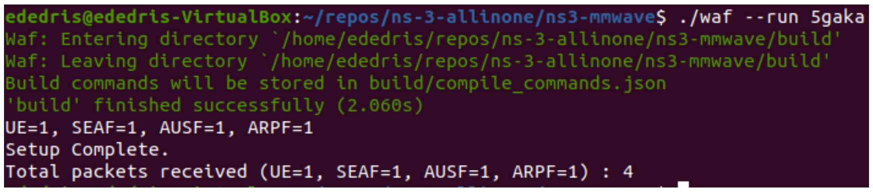
```
static bool verbose = 0;
uint32_t M1 = 1312, M2=1056, M3 = 800, M4 = 1056, M5 = 1200, M6=1296, M7 = 1184;

ApplicationContainer authenticateC(ApplicationContainer appContainer, double time,
Ptr<Node> UEA, Ptr<Node> gNB, Ptr<Node> UEB){

if (verbose){
std::cout<<"UE: "<< UEA->GetObject<Ipv4> ()->GetAddress (1, 0).GetLocal ();
std::cout<<"gNB: "<< gNB->GetObject<Ipv4> ()->GetAddress (1, 0).GetLocal ();
std::cout<<" UEB: "<< UEB->GetObject<Ipv4> ()->GetAddress (1, 0).GetLocal
()<<std::endl;
}
uint32_t M1 = 1312, M2=1056, M3 = 800, M4 = 1056, M5 = 1200, M6=1296, M7 = 1184;
appContainer = sendMessage(appContainer, time, UEA, gNB, M1);
appContainer = sendMessage(appContainer, time, UEB, gNB, M2);
appContainer = sendMessage(appContainer, time, gNB, UEA, M3);
appContainer = sendMessage(appContainer, time, gNB, UEB, M4);
appContainer = sendMessage(appContainer, time, UEA, UEB, M5);
appContainer = sendMessage(appContainer, time, UEB, UEA, M6);
appContainer = sendMessage(appContainer, time, UEA, UEB, M7);
return appContainer;
}
int
main (int argc, char *argv[])
{
uint32_t numUe = 2;
uint32_t numEnb = 1;
uint32_t serverNodes = 1;

CommandLine cmd;
cmd.AddValue ("UEA", "number of UE", numUe);
cmd.AddValue ("UEB", "number of UE", numUe);
cmd.AddValue ("gNB", "number of eNB", numEnb);
cmd.AddValue ("SN1", "number of server nodes", serverNodes);

std::cout  <<  "UEA="<< numUe <<", gNB="<< numEnb <<", UEB="<< numUe <<std::endl;

snprintf(saveFilePrefix, 50, "DDSEC_%dx%dx%d_", numUE, numEnb, serverNodes);

// Enable Pcap capture
if (tracing == true)
{
phy.EnablePcap ("ddsec", uemmmwaveDevs.Get (0));
phy.EnablePcap ("ddsec", enbmmwaveDevs.Get (0));
phy.EnablePcap ("ddsec", remoteHostsDevices.Get (0),true);
}
// Trace file
AsciiTraceHelper ascii;
phy.EnableAsciiAll(ascii.CreateFileStream("ddsec.tr"));
}
std::cout <<"Total packets received ("<< "UEA="<< numUe <<", gNB="<< numEnb <<",
UEB="<< serverNodes << ") : "<< bytes_received << std::endl;
return 0;
}
```

L.6 DDACap NS-3 Code Excerpt

```
static bool verbose = 0;
uint32_t M1 = 1056, M2=1056, M3 = 1168, M4 = 1296, M5 = 1184;

ApplicationContainer authenticateC(ApplicationContainer appContainer,double time,
Ptr<Node> UEB, Ptr<Node> gNB, Ptr<Node> UEC){

if (verbose){
std::cout<<"UE: "<< UEB->GetObject<Ipv4> ()->GetAddress (1, 0).GetLocal ();
std::cout<<"gNB: "<< gNB->GetObject<Ipv4> ()->GetAddress (1, 0).GetLocal ();
std::cout<<" UEC: "<< UEC->GetObject<Ipv4> ()->GetAddress (1, 0).GetLocal
()<<std::endl;

}
uint32_t M1 = 1056, M2=1056, M3 = 1168, M4 = 1296, M5 = 1184;
appContainer = sendMessage(appContainer, time, UEB, UEC, M1);
appContainer = sendMessage(appContainer, time, UEC, UEB, M2);
appContainer = sendMessage(appContainer, time, UEB, UEC, M3);
appContainer = sendMessage(appContainer, time, UEC, UEB, M4);
appContainer = sendMessage(appContainer, time, UEB, UEC, M5);
return appContainer;
}
int
main (int argc, char *argv[])
{
uint32_t numUe = 2;
uint32_t numEnb = 1;
uint32_t serverNodes = 1;

CommandLine cmd;
cmd.AddValue ("UE", "number of UEs", numUE);
cmd.AddValue ("gNB", "number of eNB", numEnb);
cmd.AddValue ("SN1", "number of server nodes", serverNodes);
cmd.AddValue ("s", "Define the prefix for .pcap anf .xml files. Default: DDACAP ",
saveFilePrefix);

snprintf(saveFilePrefix, 50, "DDACAP_%dx%dx%d_", numUe, numEnb, serverNodes);

//Enable Pcap capture
if (tracing == true)
{
phy.EnablePcap ("ddacap", uemmWaveDevs.Get (0));
phy.EnablePcap ("ddacap", enbmmWaveDevs.Get (0));
phy.EnablePcap ("ddacap", remoteHostsDevices.Get (0),true);
}

std::cout <<"Total packets received ("<< "UEB="<< numUe <<",
UEC="<< serverNodes << ") : "<< bytes_received << std::endl;
return 0;
}
```

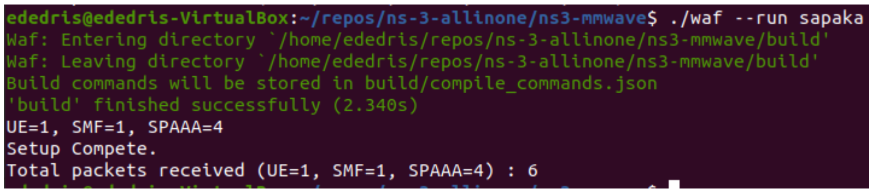
Appendix M

M.1 NS-3 Simulation Output



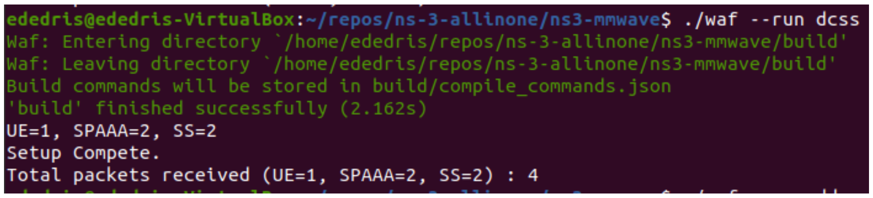
```
ededris@ededris-VirtualBox:~/repos/ns-3-allinone/ns3-mmwave$ ./waf --run 5gaka
Waf: Entering directory `/home/ededris/repos/ns-3-allinone/ns3-mmwave/build'
Waf: Leaving directory `/home/ededris/repos/ns-3-allinone/ns3-mmwave/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (2.060s)
UE=1, SEAF=1, AUSF=1, ARPF=1
Setup Complete.
Total packets received (UE=1, SEAF=1, AUSF=1, ARPF=1) : 4
```

Figure M.1: 5G-AKA simulation results



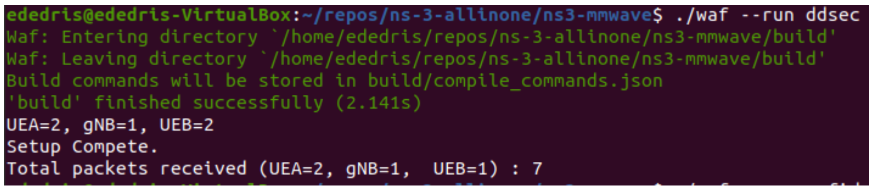
```
ededris@ededris-VirtualBox:~/repos/ns-3-allinone/ns3-mmwave$ ./waf --run sapaka
Waf: Entering directory `/home/ededris/repos/ns-3-allinone/ns3-mmwave/build'
Waf: Leaving directory `/home/ededris/repos/ns-3-allinone/ns3-mmwave/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (2.340s)
UE=1, SMF=1, SPAAA=4
Setup Compete.
Total packets received (UE=1, SMF=1, SPAAA=4) : 6
```

Figure M.2: SAP-AKA simulation results



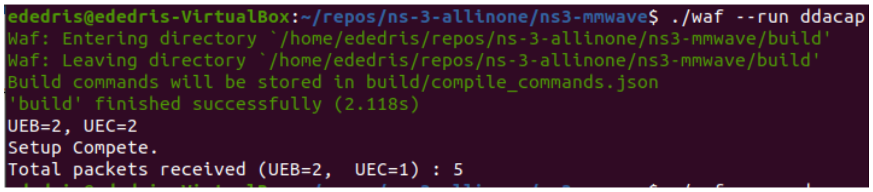
```
ededris@ededris-VirtualBox:~/repos/ns-3-allinone/ns3-mmwave$ ./waf --run dcss
Waf: Entering directory `/home/ededris/repos/ns-3-allinone/ns3-mmwave/build'
Waf: Leaving directory `/home/ededris/repos/ns-3-allinone/ns3-mmwave/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (2.162s)
UE=1, SPAAA=2, SS=2
Setup Compete.
Total packets received (UE=1, SPAAA=2, SS=2) : 4
```

Figure M.3: DCSS simulation results



```
ededris@ededris-VirtualBox:~/repos/ns-3-allinone/ns3-mmwave$ ./waf --run ddsec
Waf: Entering directory `/home/ededris/repos/ns-3-allinone/ns3-mmwave/build'
Waf: Leaving directory `/home/ededris/repos/ns-3-allinone/ns3-mmwave/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (2.141s)
UEA=2, gNB=1, UEB=2
Setup Compete.
Total packets received (UEA=2, gNB=1,  UEB=1) : 7
```

Figure M.4: DDSec simulation results

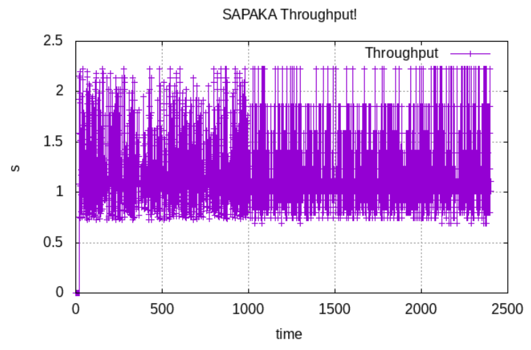


```
ededris@ededris-VirtualBox:~/repos/ns-3-allinone/ns3-mmwave$ ./waf --run ddacap
Waf: Entering directory `/home/ededris/repos/ns-3-allinone/ns3-mmwave/build'
Waf: Leaving directory `/home/ededris/repos/ns-3-allinone/ns3-mmwave/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (2.118s)
UEB=2, UEC=2
Setup Compete.
Total packets received (UEB=2,  UEC=1) : 5
```

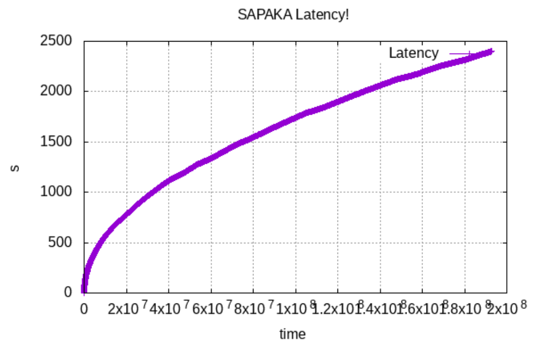
Figure M.5: DDACAp simulation results

Appendix N

N.1 NS-3 Simulation Performance Results

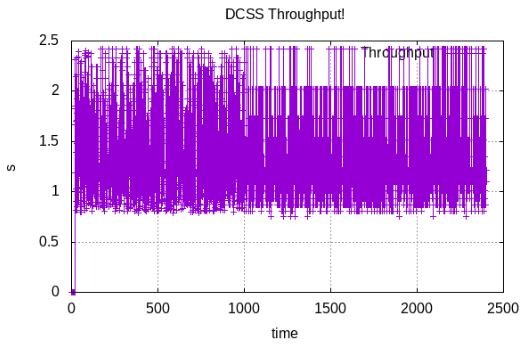


(a) Throughput

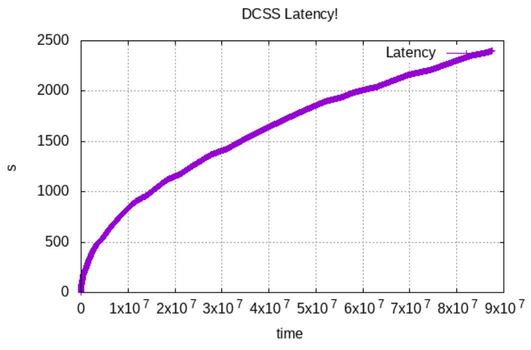


(b) Latency

Figure N.1: Communication Cost for SAP-AKA Protocol

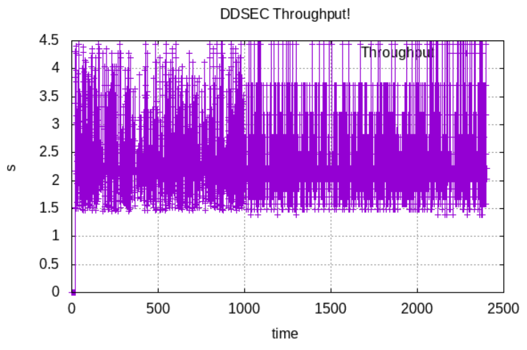


(a) Throughput

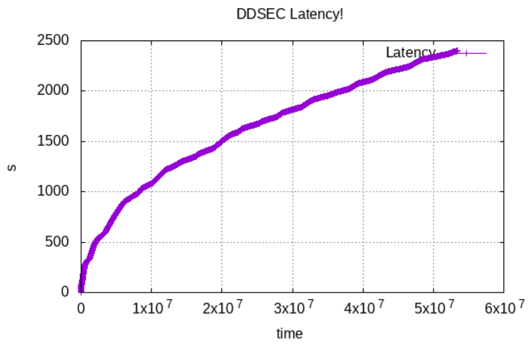


(b) Latency

Figure N.2: Communication Cost for DCSS Protocol

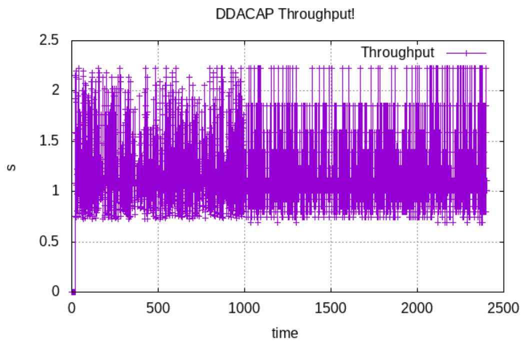


(a) Throughput

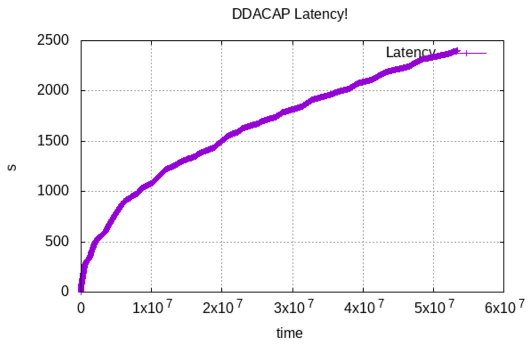


(b) Latency

Figure N.3: Communication Cost for DDSec Protocol



(a) Throughput



(b) Latency

Figure N.4: Communication Cost for DDACap Protocol