UNIVERSITY OF CALIFORNIA

Los Angeles

Paving the Way for Secure and Available Mobile Networked Systems

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Computer Science

by

Muhammad Taqi Raza Husnain Mehdi

2019

ABSTRACT OF THE DISSERTATION

Paving the Way for Secure and Available Mobile Networked Systems

by

Muhammad Taqi Raza Husnain Mehdi
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2019
Professor Songwu Lu, Chair

Today 4G mobile networked systems provide anywhere and anytime Internet access to billions of mobile users. These systems have built-in security mechanisms that protect against disclosure of information exchanged between users and the network. Despite these existing security mechanisms, an attacker is still capable of impersonating a user by forging control-plane packets and causing the service outage. Our key finding is that the attacker breaks 4G LTE encryption and integrity protection without relying on the knowledge of security key. The root causes lie on the missing binding between different LTE protocol identities and the disjoint security establishment procedures. We have found that the LTE security association setup procedures, which establish security between the device and the network, are disconnected. The security keys are installed through one procedure, whereas their associated parameters (such as uplink and downlink counters) are reset through a different procedure. The adversary can thus exploit the disjoint security setup procedures, and launch the keystream reuse attacks. He can consequently break the message encryption, when he tricks the victim into using the same pair of key and counter value to encrypt multiple messages. This control-plane attack can hijack the location update procedure, thus rendering the device to be unreachable from the Internet. Moreover, it may also deregister the victim from the LTE network.

Motivated by these attacks, we advocate for an efficient and exhaustive vulnerability analysis on 4G LTE mobile networks to discover security loopholes previously unknown. In this effort, we design algorithms that can extract new vulnerabilities and enable exhaustive security analysis in polynomial time. Our idea is to introduce multi-protocols conformance testing for validating/invalidating the interaction between the device and the network. We find that validating such interactions that require us to check all possible device states in the device finite state machine is challenging as it leads to the state explosion problem. We solve this challenge by minimizing the device states in a finite state machine by using the LTE domain knowledge. Once we get the compact representation of finite state machine, then we traverse all device states to find valid interactions between the device and the network. These interactions are then checked against the LTE standard documents to discover new undefined device operational conditions and scenarios.

Our results show that the security weaknesses also arise due to accidental systems faults, design errors, and unexpected operating conditions hence compromising 4G network availability. The core of LTE network is being redesigned because it handles the devices' control-plane and data-plane traffic and becomes susceptible to network resource constraints. To ease these constraints, Network Function Virtualization (NFV) provides high scalability and flexibility by enabling dynamic allocation of LTE core network resources. NFV achieves this by decomposing LTE Network Functions (NF) into multiple instances. However, LTE core network architecture which is designed considering fewer NF boxes does not fit well where the decomposed NF instances incur delays while executing the device events (e.g., registration, mobility, service access, and other events). The delayed execution of time-critical control-plane events brings network service unavailability. To address LTE core network limitations on its virtualization, we propose Fat-Proxy which acts as a stand-alone execution engine of critical network events. Through space uncoupling, we execute several signaling messages in parallel while skipping unnecessary messages to reduce event execution time and signaling overhead. We build our system prototype of open source LTE core network over the virtualized platform. Our results show that we can reduce event execution time and signaling overhead up to 50% and 40%, respectively.

Looking forward, this dissertation provides a new dimension for jointly solving security and availability problems in 5G and various related fields including Internet of Things (IoT), multimedia subsystems, and network analytics.

The dissertation of Muhammad Taqi Raza Husnain Mehdi is approved.

Peter L Reiher

Lixia Zhang

Christina Panagio Fragouli

Songwu Lu, Committee Chair

University of California, Los Angeles

2019

To my mother Shagufta Parveen and my Father Khurshid Ahmed Malik whose dreams have led me to pursue Ph.D.,

To my wife Fatima and daughter Huda for their motivation and support

&

To my sister Mudassara and brothers Qaisar and Zeeshan for their counseling

TABLE OF CONTENTS

List of Figures . . . . **ix**

List of Tables . . . . **xii**

Acknowledgments . . . . **xiii**

Vita . . . . **xv**

1 Introduction **1**

1.1 Research Challenges 3

1.2 Limitations of Existing Solutions 4

1.3 Our Contributions 5

1.3.1 Analyzing Security Weaknesses on LTE Protocols Interaction 5

1.3.2 Analyzing Security Weaknesses on LTE Key Installation 6

1.3.3 Performing LTE Vulnerability Analysis through Testing 8

1.3.4 Achieving High Availability of LTE Network Functions Virtualization . . . 9

1.4 Dissertation Structure 10

2 Background and the State-of-the-Art **12**

2.1 LTE Architecture 12

2.2 Limitations of the State-of-the-Art 13

2.2.1 LTE Protocols Security 13

2.2.2 LTE Key Installation Security 14

2.2.3 LTE Testing based Analysis 14

2.2.4 LTE-NFV Availability 15

3 Analyzing Security Weaknesses on LTE Protocols Interaction 17

3.1 LTE Protocols Inter-layer Interaction 17

3.2 Experimental Methodology 19

3.3 Weak Authentication: Non-Authentic Messages are Accepted 21

3.3.1 Vulnerabilities 21

3.3.2 Attacks and Validation 25

3.4 Suggested Remedies 32

4 Analyzing Security Weaknesses on LTE Key Installation 34

4.1 Integrity and Confidentiality Procedures in LTE 34

4.2 System Settings and Threat Model 35

4.3 Attacking LTE Control Plane 37

4.3.1 LTE Location Hijacking Attack 38

4.3.2 Designing LTE Service Outage Attack 50

4.4 Proposed Remedies 51

4.4.1 Remedies 52

4.4.2 Security Analysis through Prototyping 53

5 Performing LTE Vulnerability Analysis through Testing 54

5.1 LTE Testing 54

5.2 The Need of Complete LTE Testing 57

5.2.1 Limitations and Challenges 57

5.2.2 Our approach 57

5.3 LTE Testing as Finite Automata 59

5.3.1 Reducing FSM States 60

5.3.2 Implementation 67

5.3.3 Analysis 68

6 Achieving High Availability of LTE Network Functions Virtualization 72

6.1 Motivation and Problem Scope 72

6.2 Challenges in Virtualizing LTE-EPC 75

6.2.1 On Data-Center Network Characteristics 75

6.2.2 On Inter-VNF Delay 76

6.3 System Design 79

6.3.1 Design Insights 79

6.3.2 Design Overview 81

6.3.3 Fat-Proxy Design 82

6.3.4 Robustness 89

6.4 System Implementation 91

6.5 Evaluation and Results 93

7 Conclusion and Future Work 96

7.1 Summary 96

7.2 Insights and Lessons Learned 97

7.3 Future Work 99

References 101

LIST OF FIGURES

2.1 LTE architecture: an overview 12

3.1 LTE protocol layering and interaction at device and network side 18

3.2 NV reader/writer tool that modifies non-volatile memory of device (left), service programmer that helps to launch an attack from the device (center), and our testbed consisting of commodity hardware and open source platform (right) that helps to validate vulnerabilities at the network side 19

3.3 Different identities are used at various network functions 23

3.4 a) The victim's identity can be obtained from broadcast paging message b) Detach message is created by using victim's identity 25

3.5 The RRC layer helps to deliver NAS message when RRC protocol interacts with NAS protocol 27

3.6 The device logs showing that the Detach procedure is invoked over unsecured channel 28

3.7 The victim device is detached from the network on receiving detach request from attacker. 29

3.8 The device detach procedure is invoked over insecure channel 29

3.9 The network and UE logs show that the paging message with victim's IMSI can detach the victim device from the network 31

4.1 Integrity and ciphering procedures. 35

4.2 Authentication procedure installs security `key` and enables integrity protection at the device and the network. The NAS Security Mode Command procedure activates ciphering at the device and the network sides after successful completion of the authentication procedure. 39

4.3 Control plain attack main steps. 47

4.4 Error margin (min, max, avg, and std) for different experiments. Time between (a) RACH Request to RRC Connection Request messages; (b) RRC Connection Request to Security Mode Command messages (c) Authentication Response to Security Mode Complete messages; (d) Security Mode Complete (5th try) to TAU Request (3rd try). . 48

5.1 LTE test execution scenarios between device and network 55

5.2 One-transition NFA states are converted into DFA 62

5.3 Two equivalent DFA states are combined 63

5.4 RRC integrity and ciphering is not applied 69

5.5 Data transmission starts without activating RRC security 69

5.6 Re-Attach Request is delayed due to Radio Link Failure 69

6.1 Number of events per user during busy hours 73

6.2 One legacy MME is decomposed into multiple virtualized MME instances
73

6.3 Event completion time (average) during busy hours in operational EPC network 77

6.4 RTT fluctuation (average) in virtualized network 77

6.5 Signaling messages expire pre-maturely when timer based on RTT value is too short . 79

6.6 Expiry of timer between two NFs has a domino effect that leads to an event failure . . 79

6.7 State transition diagram shows that only one event at a time can be executed 80

6.8 MME NF can support mutual exclusive operations 80

6.9 Design overview 82

6.10 Generating functional call graph through gcc instrument functions 83

6.11 Function decomposition: (a) Different functions chain differently based on the event logic. (b) The function call graph consists of those functions (i) which are part of an event (right nodes) and (ii) which are called within a particular functions (left nodes). (c) In other words, functions are nested and only runtime call graph can identify these functions (i.e. ii) 84

6.12 Identifying logic dependencies through Common Subgraph Isomorphism (ICS) and data dependencies through protocol analysis 88

6.13 Concurrent execution of messages 90

6.14 Total no of signaling messages per subscriber/hr 93

6.15 Signaling load vEPC is exposed during peak hours with and without Fat-Proxy 94

6.16 Event execution time during peak hours with and without Fat-Proxy concept 94

LIST OF TABLES

3.1 Classification of LTE identifications 22

4.1 Silently getting victim TMSI: The time margin the attacker has to hang-up the call by making sure that (1) paging message is broadcasted towards victim, and (2) victim device does not ring. 42

4.2 Forcing victim to establish CSFB call connection instead of VoLTE: The error margin in terms of time the attacker has to start UL jamming so that the victim device fails to establish the VoLTE call connection. As a result, the victim phone receives call through CSFB procedure. 44

5.1 Summary of test cases – our procedure finds new legitimate test cases 66

5.2 Summary of novel findings. 68

ACKNOWLEDGMENTS

First of all, I would like to thank Almighty Allah for giving me the knowledge, strength, ability and the opportunity to conduct my Ph.D. research and completing the dissertation. Without His Mercy and Blessings, this accomplishment was not possible.

I am thankful to my Ph.D. supervisor Professor Songwu Lu for his mentorship throughout my Ph.D. I have been fortune to have a supervisor like Professor Lu who cared so much about my research, and who responded to my questions and queries so promptly. He has guided my research towards excellence. He provided his deep insights and valuable suggestions throughout my Ph.D. studies. I am also indebted to Professor Lu for his selfless support and being patient during my learning process of doing scientific research. Many thanks to Professor Lu as without him I would have never been able to perform the high-quality research, and so this dissertation.

I am also thankful to Professor Mario Gerla for shepherding my research and acting as my co-advisor. Professor Mario mentored my research by sharing his valuable early research experiences in designing Internet protocols. He shared his experience of early Internet protocols design with me. The lessons he learned had helped me developing effective cloud computing protocols. It is with great sadness that he is not among us today and I will genuinely miss his mentorship.

I am grateful to the rest of my Ph.D. dissertation committee, Professor Lixia Zhang, Professor Peter Reiher, and Professor Christina Fragouli for their feedbacks and comments that have significantly improved the quality of my research.

I want to extend my appreciation to the Computer Science admission committee, especially Professor Miodrag Potkonjak, and the Computer Science Department for providing me lifetime opportunity to pursue my Ph.D. at UCLA. I am grateful to the UCLA Computer Science Department and the Graduate Division for supporting my research through several awards: Computer Science Fellowship, Teaching Fellowship, and the Dissertation Year Award Fellowship.

I am grateful to Professor Seung-Wha Yoo and Professor Ki-Hyung Kim from Ajou University for cultivating my interests in doing scientific research and motivating me to do Ph.D.

I want to acknowledge the support from the Qualcomm Research and the Hewlett-Packard Labs. I thank Dongho Kim and Kyu-Han Kim for mentoring my research on NFV.

I thank my colleagues and friends of the WiNG Research Lab for their companionship and support. I am also thankful for their help in my research, whenever I needed it.

I would also like to thank my friends (too many to list but you know who you are!) for their motivation, support and friendship.

This dissertation was not possible without the family support that I have received throughout my Ph.D. My wife, Fatima has been incredibly supportive of me throughout my Ph.D. She has made countless sacrifices to help me get to this point. My daughter, Huda, has continually provided the requisite breaks from my work and motivated me to finish my degree on time. My parents, Shagufta and Khurshid, for their sacrifice, unconditional support, encouragement and love, and without which I would not have come this far. My sister, Mudassara, and my brothers, Qaisar and Zeeshan, I can't thank you enough for your encouragement and motivation throughout my Ph.D. journey.

VITA

2002–2006	B.S., Information Technology, NUST, Pakistan
2006–2008	M.S., Information and Communication Engineering, Ajou University, South Korea
2008–2010	Member of Engineering Staff, ETRI, South Korea
2010–2013	Senior Research Engineering, LG Electronics, South Korea
2013–2017	M.S., Computer Science, UCLA
2014–2016	Teaching Assistant, Computer Science, UCLA
2016–2019	Teaching Fellow, Computer Science, UCLA

PUBLICATIONS

Muhammad Taqi Raza, and Songwu Lu, “Systematic Way to LTE Testing”, In *25th ACM International Conference on Mobile Computing and Networking (ACM MobiCom), 2019.*

Muhammad Taqi Raza, Songwu Lu, and Mario Gerla, “vEPC-sec: Securing LTE Network Functions Virtualization on Public Cloud”, In *IEEE Transactions on Information Forensics and Security (IEEE TIFS), 2019.*

Muhammad Taqi Raza, Songwu Lu, Mario Gerla, and Xi Li, “Refactoring Network Functions Modules to Reduce Latencies and Improve Fault Tolerance in NFV”, In *IEEE Journal on Selected*

Areas in Communications (IEEE JSAC), 2018.

Muhammad Taqi Raza, Dongho Kim, Kyu-Han Kim, Songwu Lu, and Mario Gerla, "Rethinking LTE Network Functions Virtualization", In *25th IEEE International Conference on Network Protocols (IEEE ICNP), 2017.*

Muhammad Taqi Raza, and Songwu Lu, "Enabling Low Latency and High Reliability for IMS-NFV", In *13th ACM/IEEE International Conference on Network and Service Management (ACM/IEEE CNSM), 2017.*

Muhammad Taqi Raza, Fatima Muhammad Anwar, and Songwu Lu, "Exposing LTE Security Weaknesses at Protocol Inter-Layer, and Inter-Radio Interactions", In *13th International Conference on Security and Privacy in Communication Networks (SecureComm), 2017.*

Muhammad Taqi Raza, Hsiao-Yun Tseng, ChangLong Li, and Songwu Lu, "Modular Redundancy for Cloud based IMS Robustness", In *15th International Symposium on Mobility Management and Wireless Access (ACM MobiWac), 2017.*

CHAPTER 1

Introduction

The fourth-generation (4G) Long Term Evolution (LTE) network is the latest mobile network technology to offer wide-area mobile and wireless access to smartphones and tablet devices. LTE is a complex network technology that consists of numerous subsystems – designed to provide uninterrupted network connectivity, and backward compatibility to legacy cellular networks. The operations of these subsystems are standardized in more than 200 documents [TOT]. These standards guarantee interoperability between the device and the network. Like any other network, LTE employs mechanisms to ensure authentication, authorization, access control, and user data confidentiality between the device and the network. These security measures promise the protection of mobile user activity over the wireless network. LTE standard has defined device operations and their security mechanisms for normal usage scenarios. These device operations follow specific patterns during normal interaction between the device and the network. However, corner use cases arise when the device operations deviate from their defined patterns, with their security mechanisms no longer intact. These deviations have not been envisioned beforehand by the standard body.

In this dissertation, we aim to find those abnormal device usage scenarios that make LTE protocols vulnerable. First, we study the abnormal device usage scenarios on LTE protocols interactions. Much like the Internet and WiFi communication design, LTE protocol layers are functionally independent, yet these layers communicate with each other to facilitate device operation. Device operations are carried out by transferring the data-plane packets between different layers in the LTE protocol. Potential loopholes arise when LTE security mechanisms do not guard such inter-layer traffic flow. There are certain device control-plane messages that can escape authentication and authorization checks at these layers in the network mainly due to missing cross-layer binding.

As a result, an adversary can deregister the victim device from the LTE network. Second, we study the abnormal device usage scenarios for different LTE procedures. Specifically, we study LTE security key installation method and counters handling process as part of the security establishment procedure. In the security establishment procedure, the device first installs a new key through an authentication procedure. Once the key is installed, the network runs the security mode command procedure to reset the counter values for encryption. In reality, the signaling message may be lost or dropped. In case, the device response to the security mode command request is dropped, the network reinitiates the security mode command procedure. On receiving the replayed security mode command request from the network, the device resets the counter values again before generating the response message. By intentionally forcing count resets, the confidentiality protocol can be attacked, e.g., packets can be replayed, decrypted, and/or forged. The attacker can launch attacks on device location update and deregistration procedures. These attacks render the victim device to be unreachable from the outside world (e.g., it cannot receive voice calls), or even leaves the device without LTE service (i.e., no service scenario).

The vulnerabilities at protocol interactions and key installation procedure motivate us to conduct testing based vulnerability analysis. We test interactions between two LTE protocols and identify abnormal device usage scenarios. Our key idea is to leverage the message exchanges between the device and the network for LTE protocols testing. By examining the output messages of the device, we can infer whether the device has properly traversed the particular states of its Finite State Machines (FSMs) or not. All those output messages which are not LTE standard complaint are marked as vulnerable/untested.

The key lesson we have learned from the testing based vulnerability analysis is that a poorly designed system leads to accidental faults, design errors, and unexpected operating conditions that bring cyber attacks. Indeed, LTE mobile networks are being redesigned to support exponential data growth. The traditionally approaches by deploying additional network functions may not be suitable in evolving data industry because of two major reasons. First, the dedicated network functions in mobile operations add significant capital expenditure (CAPEX) challenge. Second, the nature of mobile broadband data traffic is dictated by new services including new Multimedia Broadcast Multicast Services (MBMS), Internet of Things (IoT) applications, all-IP based solutions, and others. This rapid capacity growth and increasing traffic diversity in LTE networks stress to revisit the existing network function deployment and operational paradigms. The Network Function Virtualization (NFV) aims to address these dynamic market requirements by leveraging commodity servers, switches, and storage in a large scale data center environment. In this effort, the core network operations (known as evolved packet core or EPC) is virtualized at the data center by providing network operators commercial off-the-shelf distributed platform for the delivery of end-to-end services. Every EPC network function is virtualized as a stand-alone virtual instance. However, EPC implementation as a virtualized network over traditional data-center network incurs higher signaling failure rate because of packet transmission time-out at the sender. Moreover, any glitches at software, virtual machine, or hardware may result in total virtual network function failure and can cause subscribers' service outage. To address these issues, we take a holistic approach to design a robust and scalable EPC virtualization architecture. Our design theme is to perform logic based network functions segregation for an event (such as mobility event, service access event, and others), rather than the instance based network functions segregation being currently done in vEPC. In our design, we extract an event's logic from each network function in the form of a module, and then assemble the extracted modules of the same event from several network functions, into a Fat-Proxy. This Fat-Proxy acts as a standalone execution engine for that event. In this way, our approach can bound packet transmission latencies and react quickly against the virtual network function failure.

1.1 Research Challenges

LTE secures its communication between the device and the network. Separate security procedures are designed to secure the device communication with the core network and the base station. In our research, we ask: whether these security mechanisms are enough to safeguard all types of device traffic? do corner use cases arise in practice that weakens LTE security? what practical attacks we can launch? and can we design backward compatible security solutions? Finding answers to these research questions were challenging that requires us to look LTE security from different angles: interactions among LTE protocols, and interactions among LTE procedures. The research challenges lie in designing new security attacks that circumvent individual protocol's security checks. We are required to identify the types of signaling packets that compromise LTE security. We need to determine the exact signaling timings, and their timely execution with high confidence to make LTE security attacks practical. The attacker is also required to identify the victim device in the wild to launch his attacks. At the solution side, the challenges lie in providing a security solution that is not only LTE standard compliant but also requires minimum changes at the device/network.

We design a comprehensive vulnerability analysis technique that validates the device and network interactions for abnormal usage scenario. This is challenging in terms of our solution to be complete and efficient. Our solution needs to be complete ensuring that all the device and the network side interactions (both normal and abnormal usage scenarios) are tested; our solution needs to be efficient ensuring that all such interactions are verified in polynomial time.

LTE network, when virtualized to address exponential data traffic demands, does not meet LTE service availability requirements, and hence it brings the denial of service attacks. This is mainly because neither the LTE network function nor the virtualization techniques are designed for each other. The retransmission of signaling packets due to network congestion causes timeout domino effect among all LTE core network functions. The virtualization approaches that provide flow-level guarantees do not provide packets level recovery. We are required to design a solution that takes both LTE domain knowledge and data center network characteristics in ensuring LTE NFV availability. We need to measure the service execution time-bounds that define the high availability of network services.

1.2 Limitations of Existing Solutions

Existing works on LTE security mainly focus on user privacy [SS16, Hon18], access control [PLT, PLW14], data charging [KKK15, LTP15], and infrastructure side vulnerabilities [QM12]. These works mainly discuss security vulnerabilities arising due to misconfigurations or operator side bugs. They do not study the seemingly working security protocols and procedures have security flaws when certain signaling messages bypass established security checks. Researchers have studied the key installation vulnerabilities in WiFi [Van17, Bor01, Stu04] but LTE key installation procedure was mainly assumed to be secure due to separate ciphering and authentication procedures and their unique counter values for both uplink and downlink traffic. Other prior works [SSK16] [ZJ16] [GPG12] have discussed the performance and vulnerability related tests for LTE. Their problem scope is limited to a certain extent and does not provide an exhaustive analysis of LTE protocols interaction. These works do not discuss abnormal yet practical use cases.

Existing literature on LTE-NFV discusses network function placement [BH14], service chaining [SP15], and network function decomposition approaches [CX15]. However, LTE-NFV is not studied on its requirements on availability and fault tolerance. The fundamental question of whether legacy LTE design will work in a virtualized environment was never studied before.

1.3 Our Contributions

We first unveil the security threats on LTE protocols interaction and LTE key installation procedure. We then perform testing based vulnerability analysis that unearths a number of vulnerabilities through a systematic approach. Finally, we made LTE service access highly available in a cloud setting by addressing the denial of service issue of LTE virtualization.

1.3.1 Analyzing Security Weaknesses on LTE Protocols Interaction

Our study reveals that some messages sent from the LTE network to the device, soon after the device recovers from its idle mode, are executed without any authentication. This gives an adversary a chance to kick the victim out of the network. This security weakness arises when different LTE protocol layers communicate with each other. In the end-to-end protocol interactions, intermediate protocol layers (either at the local device or the remote network) act as forwarding layers. They forward the packets to the layer above or below without inspecting the contents of the forwarded packets. Hence, *packet forwarding blindly facilitates such protocol interactions*. Furthermore, LTE protocol layers perform atomic network operations to interact with one another. These interactions happen without any integrity check between these layers. This signifies that *the trust among these protocol layers is unconditional*. We also find that certain control messages are accepted at the network before the device security mechanisms kick in. *LTE network assumes that certain control messages after the device's idle state are legitimate*. These messages specify the device's intent for different types of services, e.g., voice or data service, and set up the network resources accordingly. The device can misuse network resources by generating fake control messages. The potential impacts from such vulnerabilities are quite high where an adversary can kick the victim out of the network. To make things worse, the attacker does not need to interact with the victim device to launch these attacks, (i.e., no Trojan or malware is required).

1.3.2 Analyzing Security Weaknesses on LTE Key Installation

We study LTE security key installation method and counters handling process for a number of LTE procedures (such as device registration, deregistration, location update, and others). 4G (like 3G) employs Authentication and Key Agreement (AKA) protocol to install the security keys and enables the integrity protection of its signaling messages. After that, it runs a Security Mode Command procedure to activate ciphering of messages at LTE subscriber. LTE employs stream ciphers which have been a popular method of encryption for the confidentiality of its signaling and data packets. The ciphering algorithm takes the key (installed through AKA procedure), counter value and a couple of others as an input and generates keystream block. The keystream block, k is exclusive-ored (xored) with the plaintext message, m, to produce the encrypted message, $k \oplus m$ = e. In practice, the keystream is truly random that generates the ciphertext known as a one-time pad, proved unbreakable by Shannon [Sha48]. It is an established fact that the security of stream ciphers rests on never reusing the keystream block k [Kah96]. In case k is reused to cipher two different plaintext messages, m and n, then the encrypted texts $k \oplus m$ and $k \oplus n$ can be xored together to recover $m \oplus n$. By using chosen-text attack, one can further break $m \oplus n$, and gets the messages m and n.

The scenario in which LTE ciphering algorithm gives same keystream block over multiple rounds is the one in which the ciphering key remains constant and the counter value (responsible for generating random keystream block) is reset. We call this *"key reinstallation"* vulnerability. We look LTE control-plane and data-plane procedures that lead to key reinstallation attacks.

The idea behind our control-plane attacks can be summarized as follows. In the security establishment procedure, the device first installs a new key through the authentication procedure. Once the key is installed, the network runs the security mode command procedure to reset the counter values for encryption. The network re-initiates the procedure if it did not receive the desired response as an acknowledgment. Once the device receives the security mode command request again from the network, the device resets the counter values to zero and generates a new response message towards the network. This means two signaling messages sent after two security mode command responses are encrypted with the same keystream block at the device. We show that an attacker can force count resets by blocking the response to security mode command request message. In this way, he attacks LTE control-plane location update and device deregistration procedures and brings denial of service attacks at the device.

In our root cause analysis, the key questions we ask: despite all existing 4G security measures why 4G LTE security can be compromised? why formal LTE analysis that checks security properties fail to identify the security vulnerabilities? We find that unlike legacy cellular technologies such as UMTS and GSM (refer to Figure 4.7.7/1 3GPP TS 24.008 in [3GP12a] where ciphering and authentication request is made through a single signaling message), LTE security setup procedures are disjoint (authentication and ciphering are two procedures). We argue that although such design choice makes security procedures distributed and fault-tolerant, it can compromise 4G security. The security keys are installed through one procedure whereas the security count values (also known as nonces) are reset through another procedure. In this way, the attacker can launch the key reinstallation attack after constantly resetting the counters. Further, the security issues discussed in this dissertation do not violate the security properties proven in formal LTE analysis work, such as [Hus18]. The formal analysis requires that the LTE key should not be shared over the air and all protocols (and their interactions) should behave as desired by the 3GPP standard. Our attacks do not leak ciphering or integrity keys and strictly follow LTE standards. Further, although the attacker can launch the attacks by reseting the counts, he cannot repeatedly do so for more than one signaling message as the integrity protection becomes mandatory thereafter. However, this is sufficient for an attacker to launch as serious an attack as deregistering the victim subscriber from the LTE network.

1.3.3 Performing LTE Vulnerability Analysis through Testing

The service requirements, architecture and protocol functionalities of LTE are standardized by 3GPP. The 3GPP standard provides a number of LTE protocols conformance test cases to ensure that the device and network elements comply to established procedures for their control and user plane functionalities. These test cases validate the device implementation of LTE protocol standards. We find that conformance testing mainly focuses on validating/invalidating single protocol test case, mostly ignoring interactions between two protocols. This leads to incomplete LTE testing. Because a number of valid test cases related to inter-protocol interactions are not tested, the potential vulnerabilities also remain unearthed.

We argue that multiple protocol interactions in a system should be defined for testing to get a complete set of test cases. In this regard, we formulate the problem of finding multiple protocol interaction related test cases. To give a bit of background, a device and network exchange a number of messages to execute a test case. To generate one such message, the device traverses through different states of one or more LTE protocols FSMs. Simply looking at device output messages, we can tell that the device has properly transitioned different states of FSM(s). This observation significantly reduces our effort to generate device test cases. A test case is represented as an output messages combination that a device can generate. To provide a complete list of test cases, we are required to generate *all* possible combinations of these output messages. For n output messages, there are 2^n possible test cases, which are practically infeasible to analyze. We are interested in finding all those output message combinations that the device will *never* produce, which maps into a problem of finding all don't cares output values for output message combinations. We traverse device protocol FSMs in reverse (from output state towards input state) to find these don't care outputs. This is challenging especially when each protocol FSM has too many states to traverse (consider all states related to configurations, timings, and functionalities). This motivates us to reduce the number of states at device protocol FSMs. We refer to FSM reduction and minimization algorithms in a finite automaton. We propose two novel algorithms that minimize only deterministic finite automaton (DFA) states through LTE domain knowledge and skip non-deterministic finite automaton (NFA) states minimization (which is an NP complete problem). Once we get a compact representation of device protocol FSMs, we can quickly find the don't cares from output message combinations and provide a complete set of test cases without enumerating all possible device output message combinations. In this effort, we discover a number of LTE vulnerabilities: integrity and ciphering enforcement can be skipped, the user data can be sent without the success radio access security, the device re-registration procedure can be delayed for an extended amount of time, and many others.

1.3.4 Achieving High Availability of LTE Network Functions Virtualization

We analyze the impact of virtualization on EPC functionality and service provisioning. We find that current LTE EPC architecture which is designed for fewer powerful dedicated Network Functions (NFs) does not fit well when thousands or even hundreds of NFs are chained within one EPC. We discover the following two major challenges on virtualizing EPC.

Virtualized network is not designed for LTE: LTE EPC is virtualized over data center network which suffers from long queueing delays in switches [AGM11] [BAMa], packet losses [CMZ] [AYK12], timed out retransmissions [XWS14] [ZIK13], and out of order packets delivery [Far]. Because of these characteristics, virtualized NFs (VNFs) implemented over commodity data center network only provides flow level guarantees [mic], whereas LTE standard requires packet level guarantees (100ms and 300ms delays for voice and data packets, respectively) [3GP13a]. This significantly degrades user quality of service (QoS) and even causes temporary service unavailability when active sessions drop.

EPC is not designed for virtualized network: In legacy EPC, there are fewer NF boxes, which are connected through dedicated fiber links. The Round Trip Time (RTT) over a one-hop link is stable and determines NF reachability and packet retransmission counters [3GP13c] [Net14]. In virtualized EPC implementation, some network signaling packets take longer and congested path triggering unnecessary packet retransmission at the sender. Further, it causes the domino effect by triggering time out at other chained NFs. This higher signaling failure rate while executing certain network events [Luc14] have a direct impact on user traffic (e.g., voice and data) continuity. We regard these events as mission-critical events.

Our goal is to ensure that mission-critical events do not suffer delays and failures. We categorize these events being *handover* event during device mobility, *paging* event during device idle mode, and *service request* for gaining network resources. Second, because these three events cause 50% of all network signaling [NSN12], therefore, we aim to isolate them to reduce networking signaling load at EPC.

To achieve our goal, we first decompose these events from EPC and implement them separately as a Fat-Proxy. As the name suggests, the Fat-Proxy acts as an execution engine to an event. When an EPC receives event request, it forwards the request to particular Fat-Proxy – that takes responsibility of executing the event and finally flushes the updated event status and device session information to EPC. In other words, all event execution logic being local to one virtual machine does not only address above mentioned challenges but also keeps a greater number of signaling messages flow away from EPC.
In our design we also address several challenges such as identifying event specify logic from chained NFs, and resolving functional dependencies while implementing Fat-Proxy, etc. Our result shows that (1) Fat-Proxy reduces more than 40% signaling load, (2) reduces up to 50% event execution time, and (3) generates up to 40% less signaling messages by skipping and parallelizing these messages.

1.4 Dissertation Structure

We organize our dissertation as follow. Chapter 2 introduces the background on mobile network architecture, and network function virtualization, as well as provides state-of-the-art work in comparison to the dissertation.

Chapter 3 discusses security issues at LTE protocol inter-layer interaction. Chapter 3.1 describes preliminaries on LTE protocol inter-layer interaction followed by experimental methodology in Chapter 3.2. Chapter 3.3 discusses vulnerabilities on inter-protocol interaction that trick the device to perform a weaker form of authentication. It discusses device deregistration attacks followed by impacts and limitation of these attacks. Chapter 3.4 suggests some of the quick fixes in order to address the vulnerabilities.

Chapter 4 highlights security weaknesses on LTE key installation procedure. Chapter 4.1 discusses how ciphering and integrity are enforced in LTE. Chapter 4.2 describes the system settings and the threat model. Chapter describes detail LTE control plane attack procedure followed by the impact. Chapter 4.4 proposes the remedies of the attack.

Chapter 5 describes testing based vulnerability analysis. Chapter 5.1 highlights the importance of LTE testing and the limitations in the existing approaches. Chapter 5.2 makes a case for complete LTE testing and the challenges in designing a complete testing technique. Chapter 5.3 discusses our approach based on finite automata. In Chapter 5.3.2, we discuss our implementation efforts, and provide an analysis in Chapter 5.3.3.

Chapter 6 describes a new way of LTE NFV over commodity data center network architecture. Chapter 6.1 motivates the problem and limits the problem scope. Chapter 6.2 describes the challenges of LTE NFV. Chapter 6.3 proposes our Fat-Proxy system design. Chapter 6.4 discusses our implementation efforts followed by the evaluation results in Chapter 6.5.

Chapter 7 summarizes the dissertation and discusses our future directions.

CHAPTER 2

Background and the State-of-the-Art

In this chapter, we first introduce the background of mobile network infrastructure, i.e., LTE mobile networks. We also present the related state-of-the-art mobile network studies.

2.1 LTE Architecture

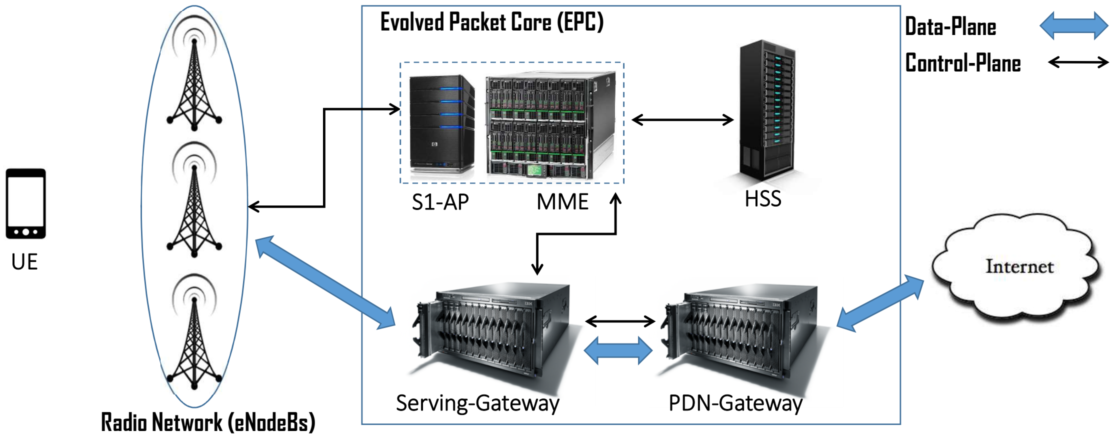


Figure 2.1: LTE architecture: an overview

LTE mobile network consists of three main elements, which are User Agent (UE), Evolved Node Base-station (eNodeB), and Evolved Packet Core (EPC), as shown by Figure 2.1. The eNodeB anchors as a radio interface between UE and EPC. EPC communicates with packet data networks in the outside world such as the Internet, private corporate networks or the IP Multimedia Subsystem (IMS) and facilitates user communication. LTE EPC comprises over a number of LTE Network Functions (NFs), that includes Mobility Management Entity (MME), HSS (Home Subscriber Server), Serving Gateway (SGW), Packet Data Network Gateway (PGW), Policy and

Charging Rules Function (PCRF), and few others. These NFs handle control-plane and data-plane traffic through separate network interfaces. As shown in Figure 2.1, control-plane traffic from radio network is sent to MME, whereas data-plane traffic is forwarded to SGW. MME acts as a central management entity that authenticates and authorizes UE, handles network events (such as *device Attach*, *Handover*, *Service provisioning*, and *Paging* events), and maintains SGW and PGW connections for data-plane traffic.

EPC NFs are static in nature and are connected, or chained, in a certain way that achieves desired overall functionality or service that LTE network is designed to provide. These NFs exchange a number of control messages to execute a specific network event. For example, during *device Attach* event, MME obtains device security keys from HSS, authenticates the device, creates device session information at SGW and PGW. Then SGW and PGW establish data bearer connection with the device and configure specific QoS profile. Thereafter, the device is said to be registered with LTE network. The delay or failure in one control-message results into complete event failure [3GP13b]. Therefore, NFs which are implemented over vendor specific software and hardware guarantee per signaling message level reliability and NFs high availability support.

2.2 Limitations of the State-of-the-Art

We present the state-of-the-art of mobile network research in comparison with the dissertation.

2.2.1 LTE Protocols Security

Closest to our work are [TLP] and [SS16]. [TLP] discloses performance issues on inter-protocol communication in operational LTE network. However, we discover security vulnerabilities that are rooted in LTE standard and do not discuss any performance bottlenecks. [SS16] discusses privacy attacks in which signaling information is leveraged to infer user privacy information. Moreover, such attacks are only possible if the network operator disables integrity and ciphering protection. For LTE DoS attacks, [SS16] assumes the attacker can change the message contents (such as device capabilities in *Attach Request*) for non-integrity protected *Attach Request* message. In contrast, our work discloses security weaknesses of common device operations even if all LTE security mechanisms are well in place. Security on mobile devices and their applications focus on permission control [BKO10], inter-application communication [CFG11], [MRF12], plagiarizing applications [PNN12] and leaking privacy information [SZZ11] by smartphones. Our attack models do not depend on any given mobile data application.

2.2.2 LTE Key Installation Security

Closest to our work is key reinstallation attack in WiFi [Van17, Bor01, Stu04]. Mathy and et al. [Van17] has recently shown a variant of key reinstallation attack in WiFi. Their work exposes design and implementation issues in WiFi security protocols that reinstall an already-in-use key. [Bor01] discusses passive and active attacks due to keystream reuses in WEP. [Stu04] shows key recovery attack on WEP. In contrast, our work although in the similar direction is different than all above works. We show key reinstallation attacks in LTE, even though LTE never reuses the same key (all keys are chained in forward direction), employs separate keys and counters for encryption and integrity protection.

Other works related to key reinstallation attacks are count reset due to power failure [Zen09]; use of static counter due to implementation bugs [B16]; faulty state machine transitions leading to count resets [Beu, De 15]; count resets through routing protocols [Aum11]; and side channel attacks on CBC mode with a block cipher [Vau02]. Contrary to these works, our work studies LTE design flaws that resets counter values welcoming key-reinstallation attacks. Our attacks are neither implementation bugs nor brought due to careless design choices. We show that seemingly working security protocols have security loopholes when certain signaling messages are re-transmitted.

2.2.3 LTE Testing based Analysis

The LTE testing is still a relatively unaddressed topic in the research community, particularly for its completeness. Some early studies [TLP] [HQG13] [CL10] have looked into cellular protocol interactions from the performance standpoint, whereas others [Hus18] [SS16] have examined practical attacks over LTE. Other prior works [SSK16] [ZJ16] [GPG12] have discussed the importance of performance or vulnerability related tests for LTE. In contrast, we focus on the LTE testing in terms of complete test cases execution. Therefore, we address a different and bigger problem scope to certain extent.

There are also prior efforts on wireless or network related testing. They include network protocol testing [Shi16] [Fay16] [Lee97], testing via model checkers [God97] [Khu03] [Fit16], and test cases generation by learning queries [Dan87] and finite state machines [TS78] [Pet17]. Specifically, [Shi16] tackles runtime wireless protocol validations by sniffing wireless transmission first and adding nondeterministic transitions later to incorporate uncertainty. The presented technique does not address the NP completeness in search and instead uses heuristics to limit the search. [Fay16] and [Lee97] discuss the model-based approach to NFV testing and network fault detection, respectively. Both model the network nodes as FSMs and generate test traffic for FSM executions. However, [Fay16] and [Lee97] do not provide complete list of test cases. [God97] and [Khu03] verify the state-space exploration. They require either constrained metrics as the input or going through all system states. In our work, we show that finding all possible inputs is practically not feasible for LTE testing.

2.2.4 LTE-NFV Availability

We elaborate on related efforts, which are categorized as follows.

Industry and academic efforts: The ETSI has provided several documents discussing guidelines and requirements for LTE-NFV. The open source NFV platform (OPNFV) [OPN] is designed to accelerate deployment efforts of LTE-NFV. There are several white papers provided by technology giants [ETSa] [ATT] [ERIb] [WIN], but none of them has demonstrated any (1) system design of LTE-NFV that solves LTE specific issues in virtualized environment (2) prototype that clearly shows the merit of LTE-NFV over legacy LTE EPC design. Our work stimulates discussion on system design that clearly shows the merit over naïve NF decomposition approach discussed in industrial white papers.

Indeed, there exists recent academic literature that discusses LTE-NFV design choices, but none of them have justified their merits in terms of performance, fault tolerance, and availability when compared to the legacy LTE network. Such as [CX15] decomposes those functions which are performed by a traditional edge router and are assigned to different elements in the system. The [HG15] discusses two implementation options for telecommunication network Gateway (GW) functions: one running a complete GW in a virtual machine and the other decomposing the GW into a control and user plane virtual machine. While [SP15] provides an algorithm to map network service chaining composed of NFs to the network infrastructure while taking possible decompositions of NFs into account.

End-to-end NF management: Closest to our work is [BH14], which discusses the NF placement problem in LTE core. While both, our work and [BH14], address NF decomposition issues but our work significantly differ from their design goals and problem statement. The [BH14] aims to use SDN for control-plane traffic whereas enhances data-plane performance through NFV decomposition. In contrast, we try to solve issues that mainly stem from the control-plane traffic explosion where time-critical events miss their tight execution deadline and effect data-plane traffic. For example, the delay in lengthy handover event (that consists of up to 35 signaling message exchanges between various NFs) cause call or data-packets drop or even causes the deregistration of the device from the network.

CHAPTER 3

Analyzing Security Weaknesses on LTE Protocols Interaction

In this chapter, we discuss the vulnerabilities arising from LTE protocols interactions. We first introduce how different LTE protocols interact, then we explain our experimental methodology to validate the vulnerabilities. Thereafter, we discuss the vulnerabilities and our attacks that exploit the weaknesses on LTE protocols interactions. Lastly, we discuss the remedies to fix the discussed vulnerabilities.

3.1 LTE Protocols Inter-layer Interaction

LTE protocols' functionality is divided across different layers, where each layer is designed to carry out a specific function [lte]. Figure 3.1 shows layered LTE protocol at the mobile device (known as User Agent - UE), LTE base-station (known as evolved NodeB - eNodeB), and LTE core-network entity (known as Mobility Management Entity - MME). The design goal of layered LTE protocol is: a) to simplify communication design by dividing it into functional layers, and b) assigning independent tasks to each protocol layer. Although the layers execute their independent tasks, the successful execution of operations lies in frequent interactions among the protocol layers. Such protocol layer interactions take place within the device, and across the device with the network. For example, two procedures known as Hybrid Automatic Repeat Request (HARQ), and Automatic Repeat Request (ARQ) are proposed at Medium Access Control (MAC) layer and Radio Link Control (RLC) layer of LTE protocol stack, respectively [Ahm13]. The combination of these two protocol layers (i.e. MAC and RLC) can be viewed as inter-layer protocol interaction. MAC and RLC protocols coordinate back and forth in a feedback channel loop to achieve reliable data transmission, (as shown in Figure 3.1).

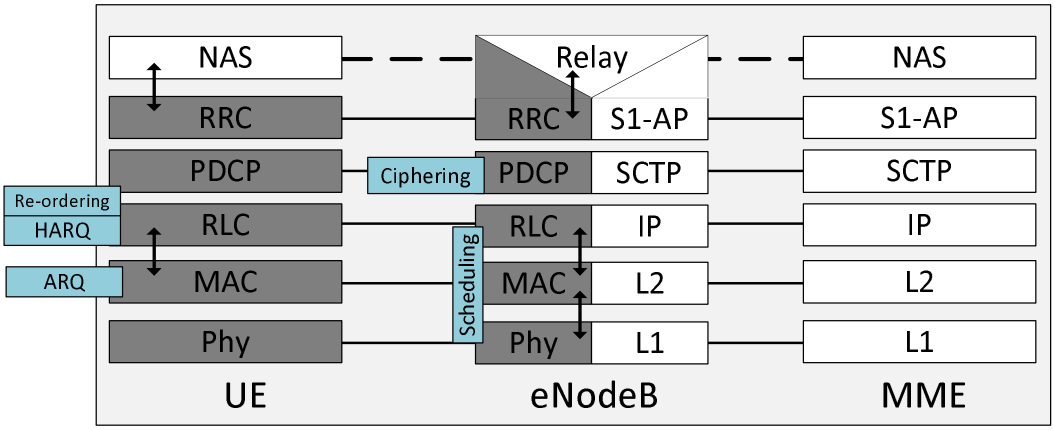


Figure 3.1: LTE protocol layering and interaction at device and network side

Another example of LTE protocol's inter-layer interaction is shown in Figure 3.1, when Radio Resource Control (RRC[1]) layer at UE is communicating with Non-Access Spectrum (NAS[2]) protocol at MME. The RRC layer is responsible for securing radio connection between UE and eNodeB, whereas the NAS ensures secure data connection between UE and MME. Although, RRC and NAS function independently, these two layers coordinate frequently in order to perform certain device/network level operations. One such operation is *device registration procedure (i.e. Attach Request message)* with the network. In this, RRC layer at UE first establishes the radio connection with eNodeB, and then NAS layer at UE registers it with MME. Since NAS operation immediately follows the successful RRC connection, NAS message piggybacks the last successful RRC message [Ahm13], to reduce the signaling overhead and, speeds up the device registration procedure [Ste11].

We show that LTE protocol's inter-layer interaction is the culprit of bypassing security setup. For example, LTE core network processes *Attach Request* message, without even authenticating the device. Similarly, *device Power-off*, *Location Update procedure*, *device Idle to Connected Mode operation*, and many other messages can be executed without authentication due to inter-layer communication.

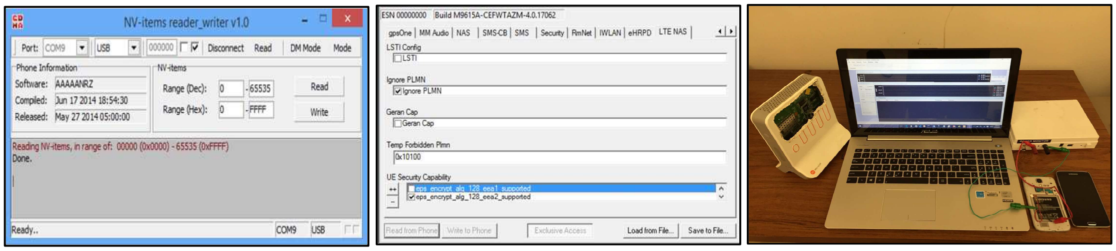
[1] The communication between UE and eNodeB is performed by RRC

[2] The communication between UE and MME is performed by NAS

We show how seemingly innocuous protocol's interaction can cause serious security threats to users' activity in the network. We have found that the vulnerabilities arise when different layers (1) accept the messages from each other without inquiring the true identity of the sender and network functions, (2) execute the message without establishing the authenticity of the message, and (3) do not validate the packets that were sent before the authentication was established.

3.2 Experimental Methodology

To validate each vulnerability, we are required to log complete device traces. LTE modem vendors (e.g., Qualcomm or Mediatek) let developers collect LTE protocol traces. Tools such Qualcomm eXtensible Diagnostic Monitor (QXDM) [Qua] and MobileInsight [mob] help to collect LTE protocol traces in operational LTE network. The real challenge is the modification of control message contents for LTE modem. The current modem implementation is hidden, and the programmer does not get any interface to inject his commands. Although AT commands [AT] are provided to activate/deactivate the device session with the network, the modem does not allow us to change the contents of these messages (such as security capabilities). We found that LTE modem's functionalities are controlled by non-volatile memory items / NV items. There are around 65535 NV items, holding values from device capabilities to its functioning parameters. In fact, the mobile phone vendors change these NV items to restore phone configurations. Figure 3.2 (left) shows freeware tool that allows us to read/write phone's NV items.



Port: COM9 USB 000000 □ ☑

Software: AAAAANRZ
Compiled: Jun 17 2014 18:54:30
Released: May 27 2014 05:00:00

Range (Dec): 0 - 65535
Range (Hex): 0 - FFFF

Reading NV-items, in range of: 00000 (0x0000) - 65535 (0xFFFF)
Done.

Ready.. COM9 USB

ESN 00000000 Build M9615A-CEFWTAZM-4.0.17062

LSTI Config
□ LSTI

Ignore PLMN
☑ Ignore PLMN

Geran Cap
□ Geran Cap

Temp Forbidden Plmn
0x10100

UE Security Capability
□ eps_encrypt_alg_128_eea1_supported
☑ eps_encrypt_alg_128_eea2_supported

Figure 3.2: NV reader/writer tool that modifies non-volatile memory of device (left), service programmer that helps to launch an attack from the device (center), and our testbed consisting of commodity hardware and open source platform (right) that helps to validate vulnerabilities at the network side

We validated the existence of vulnerabilities by modifying the Non-Volatile Memory of the LTE modem. Then we used Qualcomm's service-programmer tool (QPST Service Programmer) [ser], and AT-command tool (TeraTerm) [ter] to communicate with the device chipset. For example, we first let the device enter into sleep mode and then issued *"Detach Request (power-off)"* message using AT-command.

In order to understand how different protocol layers communicate in a feedback loop, we parse the traces and analyze to confirm LTE standard vulnerabilities.

Last, we assess the practical implication of vulnerabilities by converting them into attacks. We launched the attacks either using Qualcomm service programmer [ser] or deploying our testbed. The Qualcomm service programmer helps modify device parameters. By changing these parameters, the adversary can impersonate the victim device. Since certain messages are accepted without integrity check, the network believes as if it is talking to the actual device. For some other type of attacks, we provide proof of concept model using our testbed. It consists of OpenEPC [ope] setup that includes gateways (Serving-GW and PDN-GW), LTE core-network entity (MME), subscriber information database (HSS), and external network proxy – all implemented in software, as well as closed source eNodeB (nanoLTE Access Point [nan]). We have used two Android phones (i.e. Samsung S4 (with Qualcomm's LTE modem MDM-9215 chipset), and S5 (with Qualcomm's LTE modem MDM-9635 chipset)) with USIM cards programmed with the appropriate identification name and secret code to connect with the base-station. Figure 3.2 (right) gives a snapshot of our testbed that consists of commodity hardware devices including two smart-phones, 3G femto-cell, power monitor tool, and a laptop.

The following section dig deep into the root causes of major exposed vulnerabilities, reveal how these security loopholes arise, and what special attacks can be launched to exploit the LTE protocol's weaknesses.

3.3 Weak Authentication: Non-Authentic Messages are Accepted

LTE employs power saving mechanisms in which device enters into *RRC Idle state* when it has nothing to send/receive any data (CS or PS). In *RRC Idle state*, the UE releases its radio connection and deactivates the security connection with eNodeB. When UE has some data to send/receive, the UE establishes its radio connection with eNodeB and switches to *RRC Connected state*. After moving to *RRC Connected state*, the device renews its RRC security with eNodeB. However, a threat exists when the UE is able to communicate with the network before activating its radio security procedure. In fact, it is allowed by the network to boost device performance by preparing network resources for the UE beforehand.

3.3.1 Vulnerabilities

When the device enters into the connected state, the protocol layers interact to facilitate each other's functions to improve the response time from the network. Issues arise when these protocol functions are used to carry unauthorized traffic.

In the following subsections, we discuss how such protocols' interaction can be vulnerable when the security shield is not yet in place.

3.3.1.1 Blind Forwarding

The logical division of protocols into different layers provide distributed functionality for complex LTE operation. A single protocol cannot perform any functionality without communicating with layers above and below. Such interaction is divided into two different parts where 1) one layer communicates with the layer immediately above or below, and 2) a layer communicates with another layer which is either significantly far in the protocol stack or located at a remote host. In the case of 2), the intermediate layers simply relay anonymous packets. For example, a mobile device establishes RRC layer connection with eNodeB while the device forms NAS layer connection with MME through the eNodeB (refer to Figure 3.1). The eNodeB relays NAS messages to MME without looking into the message contents [3GP13b]. Such an implementation removes security

Table 3.1: Classification of LTE identifications

Group	LTE ID Name	Usage
UE ID	IMSI, GUTI, S-TMSI, IP address, C-RNTI, eNodeB UE S1AP ID, MME UE S1AP ID, Old UE X2AP ID, UE X2AP ID	UE, eNodeB and MME
Mobile Hardware ID	IMEI	UE and MME
Location ID	TAI, TAC	UE and MME
Session ID	PDN ID (APN), EPS Bearer ID, E-RAB ID, DRAB ID, TEID, LBI	UE and MME

threats between the device and core-network communication, in case the eNodeB is compromised. Hence, message forwarding without any inspection across different layers of protocols is rooted in the design.

3.3.1.2 Disjoint Identifications

There are a number of different identities used in LTE, grouped based on their function and usage scenarios. For example, IMSI (International Mobile Subscriber Identity) is a permanent subscriber identity used by mobile operators to identify mobile subscribers. Leakage of such identity can lead to a number of user privacy issues. Therefore, a Temporary Mobile Subscriber Identity (TMSI) is used instead to ensure the privacy of the mobile subscriber. The network provides a mapping between IMSI and TMSI to establish on-demand network resources for the device.

LTE network further maintains other identities and group them according to their usage in different network functions. Some of these identities are commissioned upon equipment installation, others are provisioned by the operator before or during service operation, and some are created when user accesses the network for its services. Table 3.1 sums up all LTE identities as per their classification. We find that some of the identities are not mapped with any other identity in their group. That is, these identities do not hold any identity relation and remain disjoint. This introduces the potential threat where one part of user traffic is communicated with its true identity, whereas the rest of communication is allowed to be carried out by fake identity.

When the device attaches with the network it receives a number of identities. The MME assigns TMSI to UE based on which the UE can be uniquely identified at MME. Similarly, the eNodeB assigns C-RNTI[3] to distinguish the devices within the radio network. The S1AP[4] layer handles the control messages between an eNodeB and an MME. In order to tell which control message is for which UE, an eNodeB allocates an ID (eNodeB UE S1AP ID) to each UE when it sends the message for a UE to an MME. Similarly, in order to tell which control message is for which UE in which eNodeB, the MME allocates an ID (MME UE S1AP ID) to each UE when it sends the first message for a UE to an eNodeB. Both eNodeB UE S1AP ID and MME UE S1AP ID have one to one mapping that distinguishes a UE across MME and eNodeB.

When the eNodeB receives the message, it maps the UE C-RNTI with eNodeB UE S1AP ID and forwards the packet to MME. The S1AP layer of MME receives the message and forwards it to the MME core function. The MME recognizes UE based on IMSI/TMSI and performs the desired action.

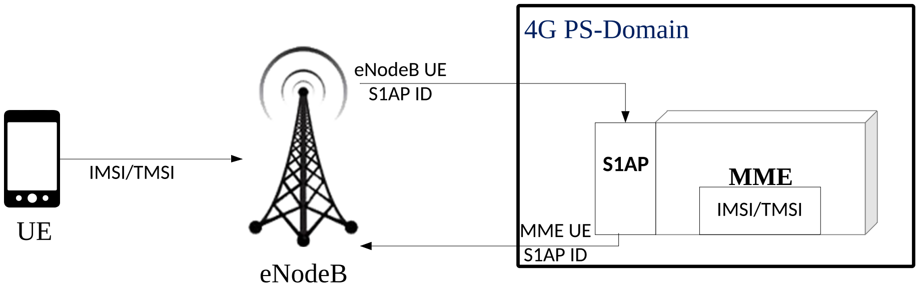


Figure 3.3: Different identities are used at various network functions

A potential vulnerability occurs due to the missing mapping between MME UE S1AP ID and IMSI. As shown in Figure 3.3, the device generates the NAS message by putting the victim's IMSI and sends this to eNodeB. When the eNodeB receives the message from the device, it correctly maps the device C-RNTI and its associated S1AP ID pair, and forwards the message to MME.

[3] Cell Radio Network Temporary Identifier (C-RNTI) identifies UE over the air.

[4] S1AP facilitates control-plane traffic between eNodeB and MME.

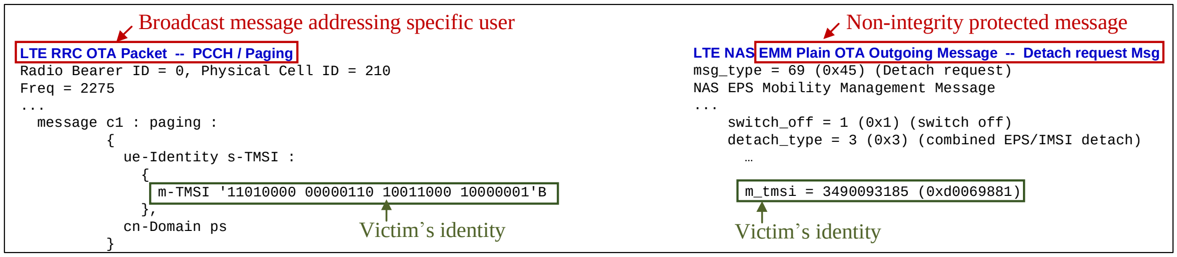
The MME S1AP layer removes the S1AP header and forwards the actual message to MME core function. The MME core function does not have any mapping between S1AP ID and associated IMSI; therefore, it takes action based on provided IMSI without checking whether the originator of the message is a genuine subscriber or not.

3.3.1.3 Blind Execution of Messages

As stated earlier, when the device switches from idle state to connected state, it is required to establish radio security. Before such security messages exchange takes place, certain messages need to be executed first. These messages are (1) type of operation the device has requested (2) the network resources that the device operation may need, etc. Such messages are exchanged between the device and the network, which are executed at both sides in order to establish the type of activity to be performed next.

To take an example, NAS *Service Request* message informs MME about the type of service (such as PS data or CS call etc.) the UE needs imminently. To prepare the resources that the UE requires, eNodeB forwards such request to MME before initiating RRC security procedure[5]. When MME receives the NAS message, it executes the message even if message authentication code included in the message fails the integrity check or cannot be verified (Section 4.4.4.3 *Integrity checking of NAS signaling messages* in LTE NAS specification [3GP13b]). Such actions help the network to quickly prepare network resources for device but come at the cost of security risks where an attacker can get unauthenticated messages executed at MME. There exists a vulnerability when the attacker makes MME processes non-integrity protected message. For example, the attacker sends a non-integrity protected *Service Request* message to MME and puts victim's TMSI in the message. MME first receives and then processes the NAS *Service Request* message where it finds the message to be non-integrity protected. The MME generates *Service Reject* message by rejecting the request with cause "*UE identity cannot be derived by the network*" and sends this message to victim UE. On

[5] Section 5.3.3 *RRC connection establishment* procedure and Section 5.3.4 *Initial security activation* in LTE RRC specification [3GP12b]. Note that initial NAS message (such as *Service Request*) is sent as a piggybacked message with *RRCConnectionSetupComplete* message that eNodeB forwards to MME. However, *SecurityModeCommand* message is sent thereafter.


Broadcast message addressing specific user
Non-integrity protected message
Victim’s identity
Victim’s identity

```
LTE RRC OTA Packet  --  PCCH / Paging
Radio Bearer ID = 0, Physical Cell ID = 210
Freq = 2275
...
  message c1 : paging :
          {
            ue-Identity s-TMSI :
              {
                m-TMSI '11010000 00000110 10011000 10000001'B
              },
            cn-Domain ps
          }
```

```
LTE NAS EMM Plain OTA Outgoing Message  --  Detach request Msg
msg_type = 69 (0x45) (Detach request)
NAS EPS Mobility Management Message
...
    switch_off = 1 (0x1) (switch off)
    detach_type = 3 (0x3) (combined EPS/IMSI detach)
      …

    m_tmsi = 3490093185 (0xd0069881)
```

Figure 3.4: a) The victim’s identity can be obtained from broadcast paging message b) Detach message is created by using victim’s identity

receiving *Service Reject* message, victim device enters into deregistered state and initiates the attach procedure. In short, an attacker can exploit those NAS messages which are processed by MME even if these messages are not integrity protected.

3.3.2 Attacks and Validation

The three vulnerabilities explained above are rooted in the LTE protocol design and can be exploited even when LTE security shields are well in place. We assume that all components function normally without any misconfiguration, malware, or intrusion. We further assume that all other mechanisms in cellular networks and at other mobile clients work properly. Irrespective of such measures, the attacker can still leverage improper operations at network function to launch attacks against the victim.

The attacker connects to radio network as a legitimate user. Once the radio connection has been setup, it announces the victim’s identity in the NAS message and requests radio layer (RRC) to forward it to MME. The MME receives the message from eNodeB and assumes that the message is part of the chain of steps needed for specific device operation. The MME then executes the message and sends back an acknowledgement to the victim.

This threat becomes more powerful when the attacker is able to execute the message on behalf of victim without asking for an acknowledgement.

3.3.2.1 Detach a Victim from the Network through Spoofed Message

In this exploit, the attacker can detach any device from the network. This attack is launched when RRC layer at device communicates with the NAS layer at MME. When the device switches from idle state to connected state, it first establishes the RRC connection. The device is allowed to send piggybacked NAS message with the acknowledgement of radio connection setup (i.e., *RRC Setup Complete* message). The attacker takes advantage of this and sends UE *Detach Request* message with an action of *power-off* to MME by putting victim's identity in the message. Once the MME receives the message, it first verifies the integrity of the message by the checking message authentication code of the message. Because this message is not originated from the legal subscriber, the integrity check fails at MME. However, LTE standard mandates the *Detach Request* message with *power-off* type should be processed by MME even if its integrity check fails or even the message does not include message authentication code (Section 4.4.4.3 and Section 5.5.2.2.2 in [3GP13b]). Once the MME receives the message, it takes an action for *power-off* request by releasing the victim's network resources. Note that the *device power-off* reason does not trigger acknowledgement from the network to the victim device (Figure 5.5.2.2.1.1: UE initiated detach procedure in LTE NAS specification [3GP13b]) that makes victim device wrongly believe that MME is out of service. The victim device remains out-of-service until the victim performs hard-reboot on device or uses the airplane mode feature to initiate the device attach procedure.

In order to launch this attack, the adversary needs to expose the victim's identity, which can be obtained from the following procedure.

Exposing victim's identity When the device attaches with the network, it is assigned with TMSI. All the communication between the device and the network is based on TMSI. The TMSI is valid until the UE remains within the reach of serving MME – which typically handles all the devices within a large metropolitan city [mmea].

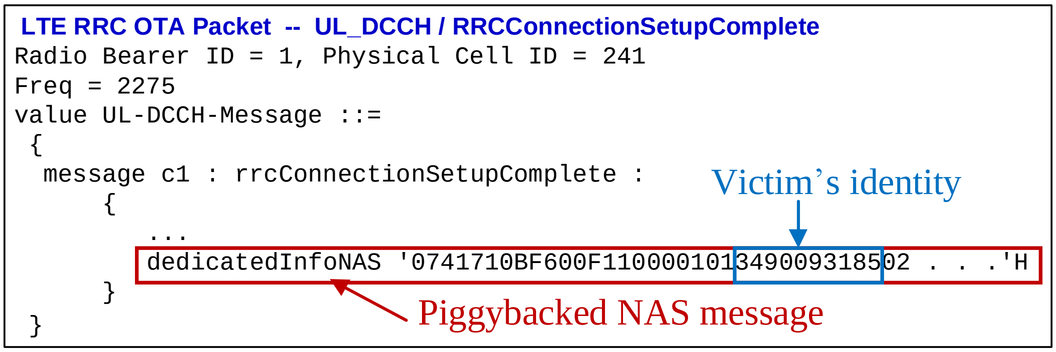
The device enters into idle state when it has nothing to send or receive. If a PS data or CS call is destined for the device during idle state, the MME sends *paging-message*[6] to that device. On

[6] Paging message is a control beacon sent from LTE network to a device, when packet switched (PS) data, or circuit switched (CS) call is impending at LTE core network. These paging messages are sent when device is in *RRC Idle*

receiving this *paging message*, the device enters into connected state and receives the traffic. Since the device has no active connection with the network during the idle period, the *paging-messages* are broadcast in nature. All the neighboring devices receive the paging message and discard it if their identity is not listed in the message. Note that the attacker is a legitimate device connected with LTE network which also receives the paging messages destined for other devices. The attacker can simply get the TMSI of the victim out of the paging message.

The attacker can also originate a *paging message* towards the victim device. It should be recalled that whenever the device receives an incoming voice call during idle state, it is paged by the core-network. Therefore, simply calling the victim's phone number and then hanging up even before the phone rings, triggers a *paging message*. The attacker gets hold of this paging message (because paging messages are broadcasted within MME tracking area[7]) and maps the victim's TMSI value with its phone number.

We run device traces and get victims identity through *paging message* (as shown in Figure 3.4a). Then the adversary generates *Detach request message* (Figure 3.4b) piggybacked over RRC (Figure 3.5).



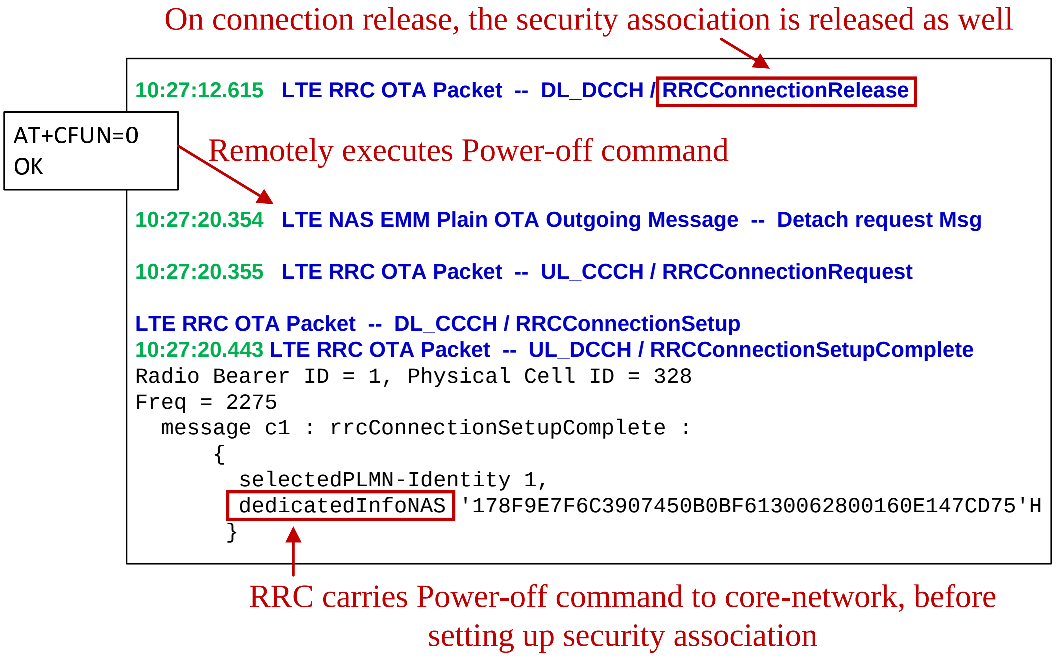
```
LTE RRC OTA Packet  --  UL_DCCH / RRCConnectionSetupComplete
Radio Bearer ID = 1, Physical Cell ID = 241
Freq = 2275
value UL-DCCH-Message ::=
 {
  message c1 : rrcConnectionSetupComplete :
      {
          ...
         dedicatedInfoNAS '0741710BF600F110000101349009318502 . . .'H
      }
 }
```

Figure 3.5: The RRC layer helps to deliver NAS message when RRC protocol interacts with NAS protocol

To launch this attack, we first register the victim device (Samsung Galaxy S4 smartphone), and the attacker device (Samsung Galaxy S5 smartphone) with our LTE testbed platform. Once

state.

[7] The tracking area is a logical concept of an area where a user can move around without updating the MME. In operational network, one tracking area spans to a number of eNodeBs.



On connection release, the security association is released as well

Remotely executes Power-off command

RRC carries Power-off command to core-network, before setting up security association

```
AT+CFUN=0
OK
```

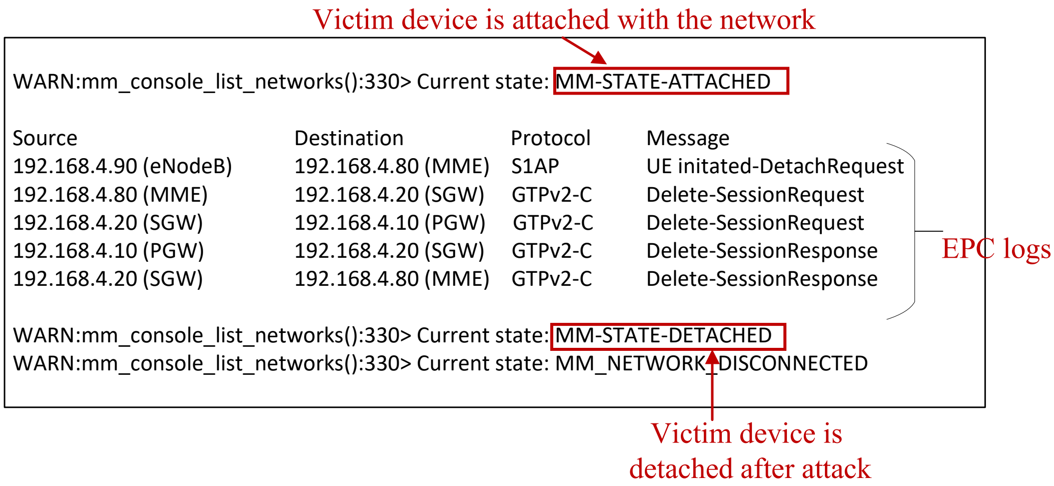
```
10:27:12.615  LTE RRC OTA Packet  --  DL_DCCH / RRCConnectionRelease

10:27:20.354  LTE NAS EMM Plain OTA Outgoing Message  --  Detach request Msg

10:27:20.355  LTE RRC OTA Packet  --  UL_CCCH / RRCConnectionRequest

LTE RRC OTA Packet  --  DL_CCCH / RRCConnectionSetup
10:27:20.443 LTE RRC OTA Packet  --  UL_DCCH / RRCConnectionSetupComplete
Radio Bearer ID = 1, Physical Cell ID = 328
Freq = 2275
  message c1 : rrcConnectionSetupComplete :
      {
        selectedPLMN-Identity 1,
        dedicatedInfoNAS '178F9E7F6C3907450B0BF6130062800160E147CD75'H
       }
```

Figure 3.6: The device logs showing that the Detach procedure is invoked over unsecured channel

both victim device and attacker are registered, the attacker sends *Detach Request message* (i.e. AT+CFUN=0) in device *RRC idle mode*, as shown in Figure 3.6. Note that in this detach request message, the attacker can masquerade victim device identity (TMSI). On receiving the detach request message, the MME finds the detach-request type as *Power-off* and immediately releases the associated device connection with Serving GW and PDN GW. We captured wireshark logs (as shown in Figure 3.7) that reveal on receiving the detach-request, the UE connection is cleared by MME, serving GW and PDN GW. The associated device is said to be "detached" and "deregistered" from core-network's view.

3.3.2.2 Detach Multiple Victims from the Network through Broadcast Message

The UE monitors a paging channel during *RRC idle* state to detect its pending notification. The UE can be paged through either of its identities, i.e. TMSI or IMSI. The LTE standard makes a distinction between paging messages generated with TMSI and with IMSI. Paging using IMSI is defined as abnormal procedure used for error recovery in the network (Section 5.6.2.2.2 *Paging for EPS services through E-UTRAN using IMSI* in LTE NAS specification [3GP13b]). The net-



```
WARN:mm_console_list_networks():330> Current state: MM-STATE-ATTACHED

Source                  Destination          Protocol     Message
192.168.4.90 (eNodeB)   192.168.4.80 (MME)   S1AP         UE initated-DetachRequest
192.168.4.80 (MME)      192.168.4.20 (SGW)   GTPv2-C      Delete-SessionRequest
192.168.4.20 (SGW)      192.168.4.10 (PGW)   GTPv2-C      Delete-SessionRequest
192.168.4.10 (PGW)      192.168.4.20 (SGW)   GTPv2-C      Delete-SessionResponse
192.168.4.20 (SGW)      192.168.4.80 (MME)   GTPv2-C      Delete-SessionResponse

WARN:mm_console_list_networks():330> Current state: MM-STATE-DETACHED
WARN:mm_console_list_networks():330> Current state: MM_NETWORK_DISCONNECTED
```

Figure 3.7: The victim device is detached from the network on receiving detach request from attacker.

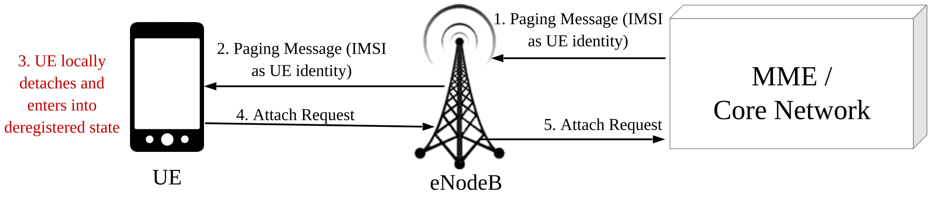


Figure 3.8: The device detach procedure is invoked over insecure channel

work may initiate paging using IMSI (as shown in Figure 3.8) if the TMSI is not available due to a network failure. Upon reception of a paging using IMSI, the UE locally deactivates any bearer context(s), detaches itself locally from LTE network and changes the state to *Network DEREGISTERED*. After performing the local detach, the UE then performs an attach procedure.

In our attack model, the attacker uses this abnormal condition to its advantage and kicks the victim out of the network. Because the paging messages are in plain text and broadcast in nature, these messages cannot be secured. Furthermore, the device executes such messages while it has not maintained any connection with the network (as it has torn down secure connection with the network before entering into idle mode). This fact brings security vulnerability where an attacker can detach the device by simply generating paging messages using IMSI as device identity. The impact of such vulnerability is enormous where an attacker can take down all of the devices connected to one eNodeB [3GP13b].

Exposing victim's IMSI identity through side channel The network operator allocates a unique IMSI to each subscriber and embeds it to customer USIM card. In order to support the subscriber identity confidentiality, the MME allocates TMSI to mobile subscribers, when the mobile device establishes a new connection with MME. Thereafter, TMSI is used as UE identity for all subsequent messages exchange between UE and MME.

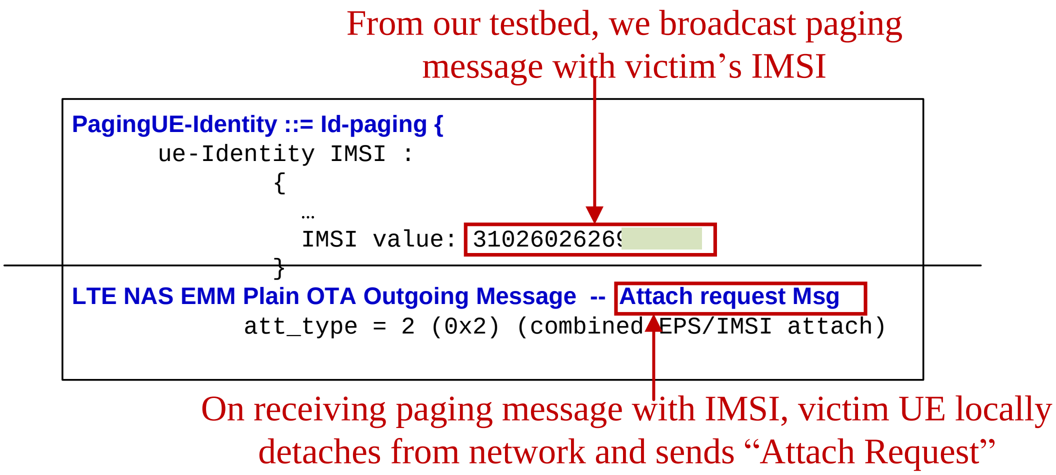
Therefore, finding the IMSI of the victim is a challenging task. Although previous studies [BU14] [GN16] have used special hardware [Str07], to expose the IMSI of a device, we discovered a new method to obtain the device IMSI using commodity hardware, i.e. 3G femto-cell.

We discover whenever the 3G femto-cell is brought within the proximity of a UE, this UE detaches from its LTE eNodeB and camps with 3G femto-cell. This is because the UE finds femto-cell signal strength higher than the serving LTE eNodeB and performs handover to femto-cell. We noticed that during this handover messages exchange, the 3G core-network sends an *identity request* message to the device, where UE responds with its IMSI. We observe this behavior because femto-cell and the eNodeB do not have any direct link with each other. As a consequence, the LTE MME does not send device security keys to 3G core-network, and let the 3G network re-authenticate the user. In order to derive the security keys, the 3G core-network needs to expose IMSI of the device and generate challenge/response messages as part of UE authentication procedure.

Note that *identity request/response* message exchange occurs prior to the establishment of device security. This makes these message exchange non-encrypted and can be logged at femto-cell. Since the femto-cell is a closed 3G base-station, *we hacked the femto-cell and defeated its in-place hardware and software security mechanisms*[8].

Once we espied victim (connected to operational LTE network carrier) IMSI through side channel, we now require the victim device to perform cell reselection to our testbed eNodeB. LTE defines priority-based cell reselection in which the device in Idle state periodically monitors its

[8] Because femtocells are part of operator network, therefore, operators take both hardware and software security measures to secure it. Therefore, as shown in Figure 3.2 (right) , we only broke small part of femtocell cover, just to access the debugging pins (JP1, JP2, JP5, JP6, PL2, etc). We used *screen command* to dump femtocell memory image. Then uncompressed it, reversed the kernel image, and looked for user information in */etc/passwd* file. We then applied brute force technique to decode the password string within 7 days.



From our testbed, we broadcast paging message with victim's IMSI

```
PagingUE-Identity ::= Id-paging {
      ue-Identity IMSI :
              {
                …
                IMSI value: 3102602626[redacted]
              }
LTE NAS EMM Plain OTA Outgoing Message  -- Attach request Msg
            att_type = 2 (0x2) (combined EPS/IMSI attach)
```

On receiving paging message with IMSI, victim UE locally detaches from network and sends "Attach Request"

Figure 3.9: The network and UE logs show that the paging message with victim's IMSI can detach the victim device from the network

neighboring cells. The priority based cell reselection ensures that the device always stay connected with higher priority cell [3GP13e]. The operational LTE eNodeB informs its associated devices about cell priorities through broadcast SIB messages. We sniff SIB4 and SIB5 parameters that define Intra-frequency and Inter-frequency LTE neighboring cells priorities [3GP12b] and configure our testbed eNodeB accordingly. We configure our eNodeB's cell as of higher cell priorities as compared to operational LTE eNodeB. This tricks the victim device to camp over our testbed eNodeB cell. Once the victim device is camped with our eNodeB cell, we generate a paging message (where we put UE identity as IMSI) towards the victim device. The victim device treats forged paging message as if it is coming from legitimate eNodeB. Soon after sending paging message, we turn-off our configured eNodeB. This is an important step that makes victim device to camp on operational eNodeB cell that forwards device attach message to operational MME. It is possible that the victim device goes through Radio Link Failure (RLF) as it was disconnected from our testbed eNodeB cell when it initiated the Attach Request message (after detaching locally). On re-establishing the radio connection (RRCConnectionRestablishment procedure), the victim device re-sends the Attach Request message (when it does not receive the reply to its first Attach Request message). We show this in Figure 3.9, on receiving the paging message with IMSI, the victim device detaches and sends a new *Attach Request* message to LTE network operator.

3.3.2.3 Impact and Limitation

In first variant of UE detach attack, the attacker can kick victim device out of the network without raising any alarm at victim device. The victim will observe out-of-network-service symbol until reboot. We believe that the victim will not reboot his device thinking that his mobile device will recover from network outage automatically. We must point out that any implementation that binds the device across all its identities (such as binding of eNodeB UE S1APID, MME UE S1AP ID, and device IMSI/TMSI) can restrain the attack.

In our second variant of the attack, we can generate one paging control message, and can potentially take down all the devices connected within the tracking area (e.g. a shopping mall or an office space etc.). The paging message allows the network to address multiple recipients by putting their identities (IMSI/TMSI) in one paging message body. Such paging message is sent to all eNodeBs defined within one tracking area. This can potentially cause network outage to all the UEs connected to these eNodeBs. The impact of this attack is limited because the device automatically reconnects with the network after detaching. Nevertheless, an attacker can keep generating paging messages with IMSI as UE identity that will keep UE barred from accessing network services.

3.4 Suggested Remedies

We suggest some remedies to address the vulnerabilities arising from inter-layer protocol interactions. Our proposal seeks to mitigate the impact from the attacks, within the current LTE standard (i.e. 3GPP standard). We should point out that the device, eNodeB, and core-network entities are 3GPP compliant and any vendor-specific implementation, conflicting with the LTE standard, may fail inter-operability between devices and the network functions. Therefore, these vulnerabilities need to be discussed in the 3GPP standard for a permanent solution. Once the operator receives the non-integrity protected “power-off” request message from the device, it should consult its database to resolve device identity (IMSI or TMSI) to eNodeB-S1AP-ID. If the received and look-up eNodeB-S1AP-IDs do not match, the network should discard the “power-off” message.

In order to address device detach using paging message, the device vendor should keep the counter value for “paging using IMSI” request messages. If the counter value exceeds a threshold defined by the vendor, the device should discard any follow-up *paging request messages*. Note that, in this attack, the adversary needs to periodically send “paging using IMSI” request messages to refrain UE from gaining network resources.

CHAPTER 4

Analyzing Security Weaknesses on LTE Key Installation

In this section, we describe the security weaknesses on LTE key installation. We first describe how integrity and ciphering procedures are implemented in LTE. Then we describe the vulnerabilities on key installation. We provide integrity and ciphering procedures in LTE, as well as on key reinstallation vulnerabilities.

4.1 Integrity and Confidentiality Procedures in LTE

LTE employs integrity and confidentiality procedures which are applied at both device and network side. Figures 4.1(a) and 4.1(b) show integrity and ciphering procedures, respectively. LTE uses two separate algorithms for integrity and ciphering of messages. Both algorithms take a number of input parameters and output the Message Authentication Code (MAC), if integrity algorithm is used, or `keystream block`, if the ciphering algorithm is used. As shown in Figure 4.1, the input parameters are 28-bit integrity/ciphering `key` named KEY, a 32-bit `count` named Count, a 5-bit bearer identity, i.e. Bearer, the 1-bit direction of the transmission i.e. Direction (0 for uplink, and 1 for downlink transmission). The integrity algorithm takes the message itself, i.e. Message, as input as well, and outputs MAC; whereas, the ciphering algorithm inputs the length of the keystream required i.e. Length, to generate the output `keystream block`. This *Length* parameter affects only the length of the `keystream block`, not the actual bits in it [3GP13d]. The `keystream block` is xored with characters in the plaintext to produce the ciphertext at sender side. Xoring the ciphertext with the same `keystream block` produces the plaintext at receiver.

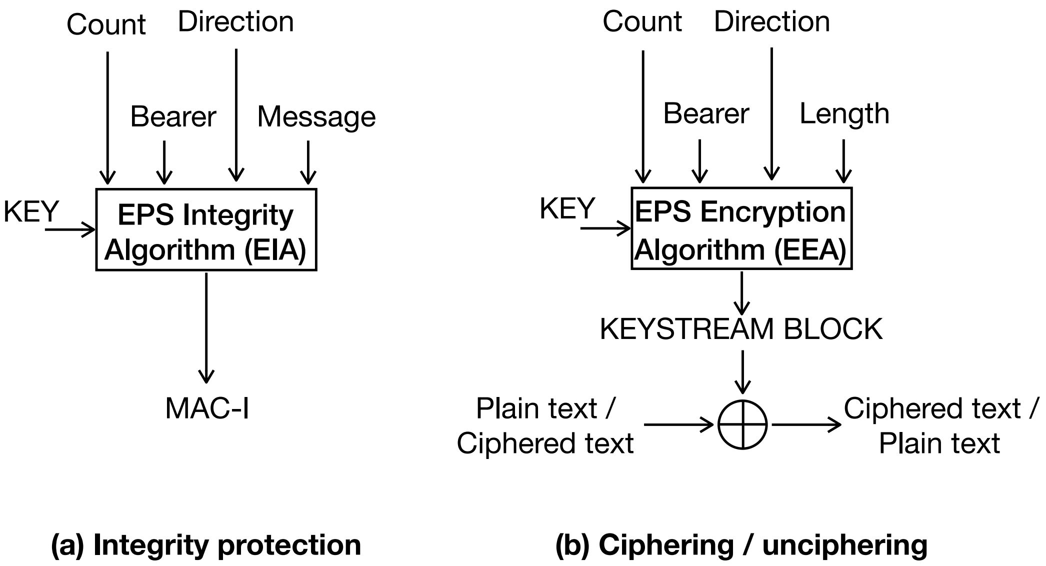

Count
Direction
Bearer
Message
KEY
EPS Integrity Algorithm (EIA)
MAC-I
(a) Integrity protection
Count
Direction
Bearer
Length
KEY
EPS Encryption Algorithm (EEA)
KEYSTREAM BLOCK
Plain text / Ciphered text
Ciphered text / Plain text
(b) Ciphering / unciphering

Figure 4.1: Integrity and ciphering procedures.

4.2 System Settings and Threat Model

System settings The attacker controls LTE device (i.e. attacker device) that is associated with the same LTE network operator as that of victim subscriber. Both attacker and victim are located in an area where the network operator supports both 3G and 4G LTE services. The attacker knows the phone number of victim device, and can dial Circuit Switched Fall Back (CSFB) call towards victim device[1]. The victim device can receive the call either through CSFB or Voice over LTE (VoLTE). Lastly, we consider both the attacker and victim devices are static during the attack period. That is, we do not evaluate the mobility scenarios.

Threat model Similar to threat models as discussed in [Rup18, Van17, Hus18], our attacker has the capability to act as a passive and an active attacker. Being a passive attacker, he can sniff the radio channel with which the victim is associated. He can do so by sniffing Physical Downlink Shared Channel (PDSCH). PDSCH is used to transport both broadcast system information for all devices and signaling/data payload for particular mobile devices. The attacker identifies different subscribers through their unique radio identity, C-RNTI.

Being an active attacker, he has the capability to modify the contents of the messages (after de-

[1] The attacker selects CSFB option (which is voice call option over 3G) in android/iOS phone call settings.

cryption) that he has sniffed over the air. There exists a number of commercial LTE signal messages sniffers, such as WaveJudge [wav], ThinkRF [thi], and others that the attacker can use to sniff both broadcast and device specific signaling messages. Contrary to attack models discussed in [Rup18, Van17], our attack model is more practical in which the attacker does not need to act as Man-in-the-Middle (MitM) to forward modified messages towards the network. To impersonate the victim device, if required, the attacker spoofs the victim's C-RNTI and TMSI values when he creates his own RRC and NAS messages and sends his signaling messages to LTE base station. The spoofing is essential to trick LTE base station to use victim context (not the attacker's context) while forwarding message to the core network.
In order to ensure that failure of certain signaling messages results in resetting the `count` values, the attacker has capability to block UL (from device to network) signaling messages. He achieves this by jamming UL signaling. There are a number of techniques [Lic16, Nas14, Lic14] to jam the signaling messages. We consider Asynchronous Off-Tone Jamming (AOTJ) approach to jam only UL signaling messages between the victim device and the network. The core idea of jamming is to introduce the inter-channel interference (ICI) among orthogonal OFDM subchannels. The interference brings loss of subchannel orthogonality, and as a result, the network cannot recover the original OFDM data symbols over its subchannels which are spectrally overlapping. In our AOTJ technique, the jammer is off-tone or not synchronous with the target signal. It transmits asynchronous off-tones which are not perfectly periodic or have an offset at the sampling frequencies that brings ICI at the receiver.

Evaluation of attacks We evaluated our attacks in terms of their feasibility and practicality over real operational LTE networks. We use Google Pixel 2 as an attacker device, and Google Pixel 1 as victim device. We consider two U.S LTE operators, i.e., AT&T (OPI) and T-mobile (OPII) to run our experiments. The attacker and victim devices use AT&T and T-mobile pre-paid sim cards to register with these two network operators. We use LTE signaling messages analyzer, MobileInsight, to capture LTE signaling messages at both attacker and victim devices. We run a total of 200 experiments on each network operator to access the practicality of attack for each attack step. We run experiments in a lab setting with LTE radio signal strength range -90 to -100 dBm for both operators.

To evaluate the practicality of the attacks, we use low-cost commodity SDR hardware (HACKRF One) of the value of $299 to jam LTE signaling messages. HACKRF One has the capability to block UL and DL LTE signaling messages by generating ICI signals towards LTE frequency band. To calibrate start and stop of jamming with respect to LTE signaling messages, we use QXDM [Qua] which is a real-time LTE signaling messages sniffing/capturing tool from Qualcomm. The victim device is connected to QXDM (running on a PC) through USB.

4.3 Attacking LTE Control Plane

Overview We demonstrate the feasibility and practicality of key reinstallation attacks in LTE control-plane. The adversary adopts the fact into his advantage that on the inter-system switch from LTE → 3G→ LTE, the location update procedure is triggered that installs the `key` and resets the `count` values. The attacker silently[2] brings an inter-system switch at victim device through CSFB procedure. He lets the device to complete the `key` installation procedure but strategically blocks the victim device UL signaling messages to bring `count` reset procedure failure. The network re-initiates the failed procedure that resets the `count` values at the device again. This results in `keystream block` reuse for those signaling messages that the victim device sends after resetting the `count` values. The attacker stops jamming, encrypts his spoofed message by using victim's `keystream block`[3], and dispatches it to the network without being MitM. The network receives two messages, the one originated from the device and the other from the attacker. The network executes the latest received message, according to 3GPP standard [3GP13b], and discards the message received earlier. This makes our attack realistic as the attacker message and the victim messages are not racing with each other. Because the message was modified by the attacker, it fails the integrity check at the network. However, instead of dropping the packet, the network re-authenticates the victim device and accepts the received spoofed message.

[2] Attacker terminates the call before the victim device starts ringing, hence making it silent inter-system switch.

[3] Attacker gets the `keystream block` by *xoring* location update message with the encrypted message.

Roadmap We first show LTE location hijacking attack due to key reinstallation vulnerabilities. We provide the feasibility analysis from LTE standard followed by the detailed attack procedure. We show step by step attack procedure and access the practicality of each step through experiments. Lastly, we extend this attack and design another type of attack, i.e. LTE service outage attack.

4.3.1 LTE Location Hijacking Attack

4.3.1.1 Feasibly Analysis from LTE Standard

Following we discuss two vulnerabilities that we exploit in attacking LTE confidentiality and integrity protocols.

LTE Integrity and confidentiality are enforced through two disjoint procedures LTE security is enforced through two separate procedures. In the first procedure, the LTE core network invokes mutual authentication procedure, i.e. AKA procedure, with the subscriber device. In LTE AKA procedure, as shown in Figure 4.2 (upper rectangular part), the core network element sends an Authentication Request message to the device. The device authenticates the LTE core network element, installs the `key` and sends the Authentication Response message to the network. LTE core network verifies the response message and installs the `key` at its end. After the authentication procedure, the network triggers NAS Security Mode Command (SMC) procedure. The network sends SMC message to device, as shown in Figure 4.2 (lower rectangular part) that includes NAS security algorithms to derive integrity and ciphering keys[4], as well as NAS-MAC (NAS Message Authentication Code). As the device does not know the selected encryption algorithm yet, this message is integrity protected only but not ciphered. On receiving the SMC message the device resets the pair of `count` (one for UL and one for DL transmission) values to *zero* after NAS-MAC verification for integrity protection. The LTE security specification (3GPP TS 33.401 [3GP13d]) states:

"*Only after EPS AKA, the NAS security mode command message shall reset NAS uplink and downlink COUNT values. Both the NAS security mode command and NAS security mode complete*

[4] For simplicity, we name all types of keys (i.e. integrity, ciphering) as `key`.

messages are protected based on reset COUNT values (zero)."

Thereafter, the device generates NAS Security Mode Complete message to the network which is both ciphered and integrity protected. The network successfully verifies the integrity of the received NAS Security Mode Complete message and resets the `counts`. Now the NAS security setup procedure is said to be completed.

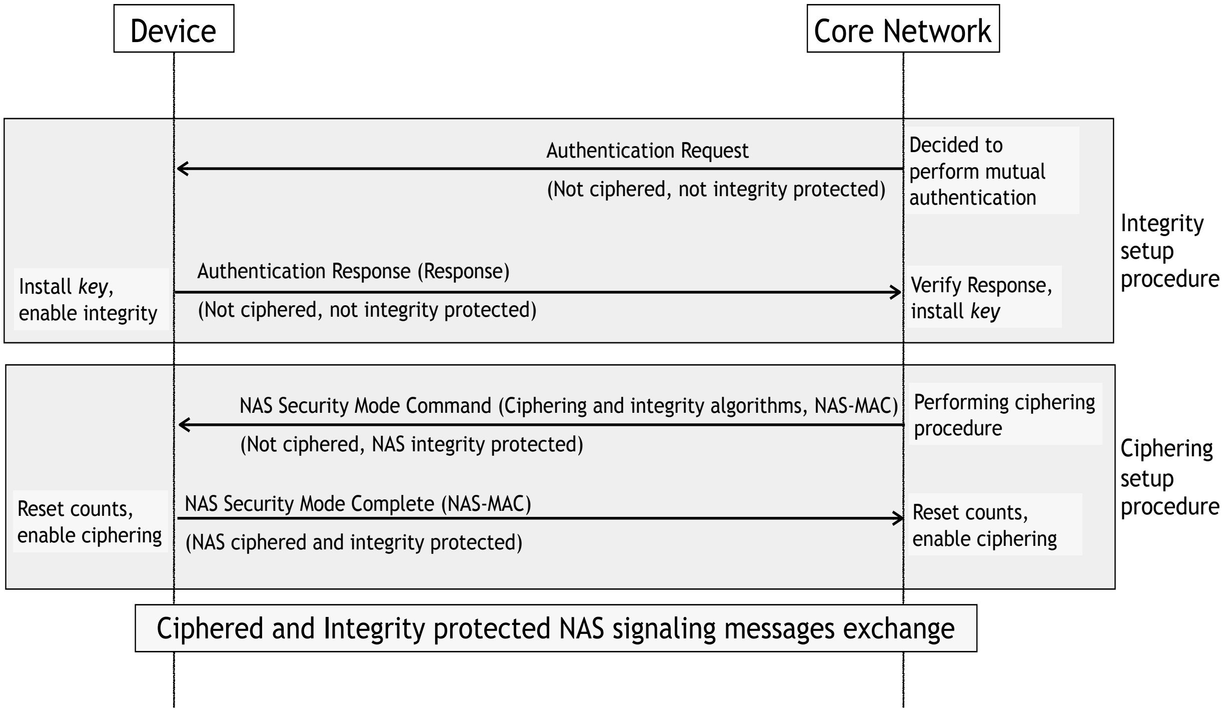

Device
Core Network
Authentication Request
(Not ciphered, not integrity protected)
Decided to perform mutual authentication
Integrity setup procedure
Install *key*, enable integrity
Authentication Response (Response)
(Not ciphered, not integrity protected)
Verify Response, install *key*
NAS Security Mode Command (Ciphering and integrity algorithms, NAS-MAC)
(Not ciphered, NAS integrity protected)
Performing ciphering procedure
Ciphering setup procedure
Reset counts, enable ciphering
NAS Security Mode Complete (NAS-MAC)
(NAS ciphered and integrity protected)
Reset counts, enable ciphering
Ciphered and Integrity protected NAS signaling messages exchange

Figure 4.2: Authentication procedure installs security `key` and enables integrity protection at the device and the network. The NAS Security Mode Command procedure activates ciphering at the device and the network sides after successful completion of the authentication procedure.

Now it is easy to see the vulnerability in which the attacker exploits the fact that the device resets the `count` values after installing the `key`. The attacker can block the transmission of NAS Security Mode Complete message and lets the network to re-initiate the SMC procedure; causing the device to reset the `counts` again. In this way, the signaling messages sent by device between subsequent SMC procedures use same the `keystream block` for their encryption.

Vulnerability 1: Failure of SMC procedure does not renew the security `key`.

Network accepts certain NAS messages that fail the integrity check It is understandable that a number of NAS signaling messages can be exchanged between the device and the network before

the activation of NAS security. These signaling messages include Attach Request, Authentication Request/Response/Failure, Security Mode Reject, Identity Response, and few others. However, there are a number of other messages (that include TAU Request and Detach Request/Accept messages) that are "conditionally" accepted when they fail the integrity check.
LTE NAS specification (3GPP TS 24.301 [3GP13b]) states:

"*These messages are processed by the MME even when the MAC that fails the integrity check or cannot be verified, as in certain situations they can be sent by the UE protected with an EPS security context that is no longer available in the network.*"

However, LTE core network re-authenticates the device before finally accepting the message. As stated in LTE NAS specification (3GPP TS 24.301 [3GP13b]):

"*If a TRACKING AREA UPDATE REQUEST message fails the integrity check, the MME shall initiate a security mode control procedure to take a new mapped EPS security context into use.*"

Such a 3GPP standard approach is vulnerable in which the network accepts the spoofed message, failing the integrity check, after re-authenticating the device.
Vulnerability 2: The network re-authenticates the device instead of rejecting certain messages failing the integrity check.

4.3.1.2 Detail Attack Procedure

We describe step by step attack procedure as follow.

Pre-condition Before launching the attack, the attacker needs to know the TMSI of the the victim subscriber for identification purpose. The attacker gets the TMSI through broadcast paging message addressing the victim device. He can easily generate the paging message for victim device by simply calling the victim. If the victim phone is in idle state, the core network sends a paging broadcast message that includes the victim's temporary identity (TMSI). On receiving the paging message, the victim device switches to the connected state and prepares to receive its call. Because the paging is a broadcast message within the tracking area, the attacker device also receives the paging message [Hus18]. By repeating this procedure, the attacker can ensure that the TMSI maps to the victim device (subscriber's phone number).

Experiment results A clever attacker would hang-up the call before the victim device starts ringing. To access the practicality of hanging-up the call so that the victim device does not start ringing, we run several experiments. We record the signaling messages and the time between call initialization and call ringing events. In our experiments, both caller and callee phones are time synchronized through which we accurately correlate the events between two phones. In total, we collected 200 logs with 2 major US operators. We consider the cases when the victim device receives the call through CSFB, and VoLTE. The attacker always makes a CSFB call (by turning off VoLTE option at its phone).

After initiating the call, the attacker must wait for paging message to be delivered to the victim device before hanging up the call. We can see from Table 4.1 that it takes around 3.5 seconds and 4.6 seconds (on average) for paging message to be received at victim device for OPI and OPII, respectively. The attacker can terminate the call afterward where he has the error margin of 3.3 seconds and 5.3 seconds (on average) to hang-up the call so that victim device does not ring for OPI and OPII, respectively. Table 4.1 also shows the results when the victim device receives the call through VoLTE instead of CSFB.

There is a possibility that the call from the attacker does not trigger any paging message towards the victim device. This is the case when the victim device is in a connected state. From Table 4.1, we can see that the attacker can easily determine whether the victim device is in idle or connected state. He first waits from call init to paging message triggering time. If he does not sniff the broadcast paging message during this period then he assumes that the victim device is in connected state. The attacker then backs-off for tens of seconds (the device's default inactivity timer – time to transition from connected to idle state – is 10s) and retries the call.

We now discuss our attack procedure in 3 main steps as shown in Figure 4.3.

① **Triggering `key` update through inter-system switch** The attacker's goal is to install fresh `key` and reset `count` values at victim device. To achieve this, he dials a phone call towards the victim to get CSFB call connection established with victim device and then hangs-up the call. The CSFB call forces victim device to perform inter-system switch (from LTE to 3G/2G). Once

Table 4.1: Silently getting victim TMSI: The time margin the attacker has to hang-up the call by making sure that (1) paging message is broadcasted towards victim, and (2) victim device does not ring.

Victim receives call through CSFB								
Operator	**Call init to paging msg**				**Paging msg to call ringing event**			
	Min	**Max**	**Avg**	**STD**	**Min**	**Max**	**Avg**	**STD**
OPI	3.2s	6.1s	4.6s	0.5s	2.4s	4.4s	3.3s	0.4s
OPII	2.5s	4.8s	3.5s	0.6s	3.5s	6.6s	5.3s	0.9s

Victim receives call through VoLTE								
Operator	**Call init to paging msg**				**Paging msg to call ringing event**			
	Min	**Max**	**Avg**	**STD**	**Min**	**Max**	**Avg**	**STD**
OPI	2.3s	4.4s	3.3s	0.5s	0.7s	2.0s	1.3s	0.3s
OPII	2.2s	4.6s	3.3s	0.6s	1.6s	2.6s	2.2s	0.3s

the attacker hangs-up the call, the victim device moves back to LTE (from 2G/3G) and performs random-access channel (RACH) procedure to synchronize with LTE base station. Through RACH procedure, the device receives a temporary radio identity (C-RNTI) mapped with its TMSI from the base station. The attacker sniffs RACH messages to associate victim subscriber's TMSI with its C-RNTI. After RACH procedure, the device setups its radio connection and sends unciphered TAU Request message as initial NAS message. The device also starts timer T3430 (default value of 15s) to retransmit the TAU Request message if a timeout occurs. On receiving the TAU Request message, the network performs the Authentication procedure through which both victim device and the network authenticate each other and install the `key`.

Experiment results We run more than 200 experiments to access the practicality of the attack. At first, we access how successfully an attacker can trigger the inter-system switch by dialing a phone call. We find that there are two cases: (1) either victim device *or* the network does not support VoLTE feature; or (2) both the victim device *and* its associated network support VoLTE.

In case of (1), the victim device automatically switches to circuit switched network, i.e. 2G or 3G, to receive the call. However, in the case of (2) the victim device does not perform the automatic inter-system switch, and the attacker needs to enforce it. From our experiments, we find that if the VoLTE call is blocked at the device for 5 seconds then the LTE modem chipset (Qualcomm LTE modem) aborts VoLTE call in favor of making the call through CSFB. This feature has also been reported in several other studies [Tu13, Tu16]. Now, the attacker strategy is to temporarily block (through UL jamming) the signaling messages between victim device and its network. But the question arises (i) when to start jamming after dialing the call?; (ii) how long the attacker can delay in starting jamming because in practice it is hard to start jamming at a precise time?; and (iii) when the attacker should hang-up the call after stopping jamming so that the victim device does not ring? For (i), table 4.2 shows error margin with min, max and average values of 2.2s, 4.3s, 3.3s with a standard deviation of 0.5s for the attacker to start UL jamming. That is, the time he has from initiating the call to sniffing the paging message (voice call indication for victim device in idle state). Once the attacker has decided to start UL jamming, he has an error margin of 0.4s (on average) with a standard deviation of 40ms to start jamming as shown in Table 4.2. This is the time in which victim device establishes the VoLTE call connection with the network, answering question point (ii). The jamming lasts for 5s that induces victim device to perform CSFB procedure to establish voice call connection over 3G/2G network instead of LTE. The attacker hangs-up the call before the victim device rings (i.e. within 3.3 seconds – refer to Table 4.1 Paging to Call ringing time – after stopping the jamming) which addresses our question point (iii). On hanging-up the call, the device switches back to LTE and performs RACH procedure that facilitates attacker to map TMSI with C-RNTI. The attacker has on average 45ms (10ms of STD) to capture RACH Response and/or RRC Connection Request message to successfully establish mapping, as shown in Figure 4.4(a).

② **Administrating `key` reinstallation attack through one-time jamming** After the authentication procedure, the core network activates the Security Mode procedure by sending integrity protected SMC message to the device and sets the message retry timer T3460 (default value of 6s). The attacker who is sniffing the radio traffic finds the SMC message matching the victim's

Table 4.2: Forcing victim to establish CSFB call connection instead of VoLTE: The error margin in terms of time the attacker has to start UL jamming so that the victim device fails to establish the VoLTE call connection. As a result, the victim phone receives call through CSFB procedure.

Operator	**Call init to call indication**				**Paging to VoLTE connection**			
	Min	**Max**	**Avg**	**STD**	**Min**	**Max**	**Avg**	**STD**
OPI	2.2s	4.3s	3.3s	0.4s	0.3s	0.6s	0.4s	0.04s
OPII	2.2s	4.6s	3.3s	0.6s	0.4s	0.7s	0.5s	0.04s

C-RNTI and *starts* UL jamming. The attacker has the error margin of 2 messages in starting UL jamming (i.e. either after sniffing Authentication Response message, or Security Mode Command message). On receiving the SMC message from the network, the device verifies message integrity, resets `counts` (*vulnerability 1* in Section 4.3.1.1), and sends Security Mode Complete message to the network. Because this UL message from the device is blocked over the air, the network does not receive this response message and its timer T3460 expires. The network re-sends SMC message to victim device by resetting the timer T3460. The victim subscriber resets its UL/DL transmission `count` values and sends the Security Mode Complete message which is blocked as well by the attacker. Similarly, the third response to network initiated Security Mode procedure is also blocked. Meanwhile, the TAU timer T3430 at victim device times out. At this point, the device has already enabled ciphering (as it has sent out Security Mode Complete messages thrice). The victim subscriber prepares new TAU Request message and applies ciphering and integrity protection. It sends out the TAU request message which the attacker sniffs and stores it at his end. We call this message TAU_1, that is TAU Request message 1 which is encrypted with `keystream block`[5]. Note that the attacker can recover the TAU message as he himself is jamming resilient. This is because he knows his off tone jamming signals and can cancel interference added to jam the signals [MM08, Yan14, Zen17, Lic14]. However, the TAU message is non-decodable at the network side due to unknown interference. When the Security Mode procedure fails for the fourth

[5] Obviously, this message is also integrity protected, but we are interested to break the ciphering only to carry out our attack

time, the attacker *stops* UL jamming. As a result, the Security Mode procedure succeeds on its fifth try where the network resets `counts` and enables ciphering its end. From this point onward, the network only accepts messages which are both integrity protected and ciphered.

Experiment results In order to make the attack practical, the attacker has to ensure that he (i) identifies the victim over the radio before starting UL jamming, and (ii) starts UL jamming before Security Mode Command complete receives at the network. For (i), he has an error margin of 380ms on average (with STD of 20ms) to identify the victim device through PDSCH. This is the time between RRC Connection request and Security Mode Command messages, as shown in Figure 4.4(b). For (ii), the attacker has on average 48ms (with 5ms STD) to start UL jamming (after Authentication Response message but before Security Mode Complete message), as shown in Figure 4.4(c).

We also perform more than 200 lab experiments to access the success probability of starting jamming within the specific time interval (i.e. 48ms). For this, we first use Qualcomm real-time packet sniffing tool QXDM [Qua] to calibrate the time between performing inter-system switch and starting UL jamming. We modify the HACRF One source code to make jamming effective and achieve UL and DL frequency jamming within 1ms after its initialization. We face two challenges in jamming specific LTE signaling message(s). From our experiments, we find that when we jam signals for more than 6 seconds the device internally triggers radio link failure, and if we continue jamming then the device switches to 3G network. To address this challenge, we systematically switch on and off jamming in an interval of 2.5s such that desired signaling messages remain blocked when they are re-transmitted on their time-out. The other challenge we face was regarding jamming UL signaling messages. We find that the device increases its UL transmit power (as high as 25dBm whereas our HACKRF One max UL transmit power is 15dBm) that renders UL jamming through low-cost SDR device ineffective. To address this challenge, we perform DL jamming instead and block the TAU Accept message reaching towards the device. As the TAU procedure does not succeed after all, the network responds to retransmitted TAU request messages (as well as the spoofed message to be discussed in the next step below) even if it has received TAU request message earlier. Hence, we can successfully execute our attack step in practice.

On the practicality of jamming We briefly discuss that our jamming works even if the attacker lacks LTE dedicated channel sniffing capability. We can always start jamming at the desired signaling message with high probability. To evaluate this, we use three different methods, as discussed below:

Straw-man approach: The attacker makes a CSFB call towards the victim, hangs-up the call as soon as the victim subscriber receives paging message, and starts jamming after waiting for 450ms (calculated according to Figure 4.4). We see that in this case, the success probability of jamming is just 21$\%$. This is mainly because the attacker hangs-up call while the victim device was in the middle of call establishment procedure. This triggers location update procedure in 3G and the device does not release the connection towards LTE network.

Measured approach: To address the problem of straw-man approach, we let the control-plane call establishment procedure to be completed before hanging up the call. The attacker lets the call establishment procedure to be completed before it hangs-up the call (just before call ringing). Hanging-up the call at this time triggers RRC connection release towards LTE network and the victim device immediately switches back to LTE network. The attacker starts the jamming after waiting for 450ms and gets the desired message blocked with the accuracy of 58$\%$. The accuracy is halfed due to variable time of inter-system switch (i.e. how quickly LTE cell is selected).

Adaptive approach: Instead of calculating the jamming start time from call release event, we improve our results by sniffing the LTE broadcast RACH packet before making the jamming decision. Our results improve the jamming accuracy to 78% because in reality, we cannot 100% predict when control-plane signaling message will arrive in future.

In summary, we show that the jamming at the desired occasion can be achieved with the accuracy of roughly 80% even if the attacker does not sniff LTE dedicated channel.

③ **Spoofing location update messages through `keystream block` reuse** Because the attacker has stopped jamming in step ②, the device initiated TAU request (on expiry of TAU timer T3430), we call it TAU_2, arrives at the network[6]. The attacker sniffs this TAU request message as well and retrieves the `keystream block` by xoring either the contents of TAU_1

[6] Careful reader should note that T3430 times out earlier than TAU Accept timer T3450 (default value 6s) at the network, therefore, network does not send TAU Accept message on receiving Security Mode Complete message

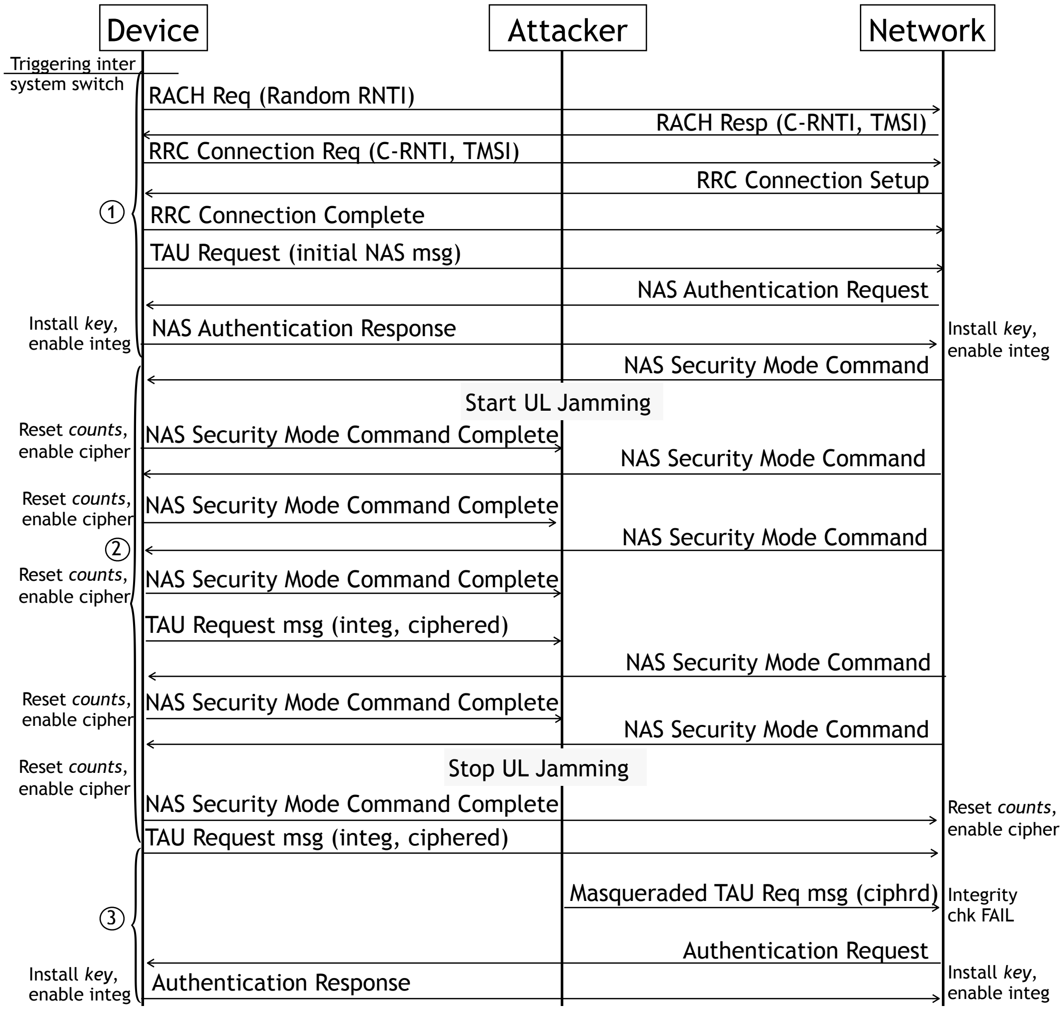

Device
Attacker
Network
Triggering inter system switch
RACH Req (Random RNTI)
RACH Resp (C-RNTI, TMSI)
RRC Connection Req (C-RNTI, TMSI)
RRC Connection Setup
①
RRC Connection Complete
TAU Request (initial NAS msg)
NAS Authentication Request
Install key, enable integ
NAS Authentication Response
Install key, enable integ
NAS Security Mode Command
Start UL Jamming
Reset counts, enable cipher
NAS Security Mode Command Complete
NAS Security Mode Command
Reset counts, enable cipher
NAS Security Mode Command Complete
NAS Security Mode Command
②
Reset counts, enable cipher
NAS Security Mode Command Complete
TAU Request msg (integ, ciphered)
NAS Security Mode Command
Reset counts, enable cipher
NAS Security Mode Command Complete
NAS Security Mode Command
Stop UL Jamming
Reset counts, enable cipher
NAS Security Mode Command Complete
Reset counts, enable cipher
TAU Request msg (integ, ciphered)
Masqueraded TAU Req msg (ciphrd)
Integrity chk FAIL
③
Authentication Request
Install key, enable integ
Authentication Response
Install key, enable integ

Figure 4.3: Control plain attack main steps.

or TAU_2[7]. Recall that, he already gets hold of TAU request message (as initial NAS message) sent in plain text in step ①. Once he retrieves `keystream block` from the ciphered text, he encrypts his spoofed TAU request message that includes wrong device location identity by xoring the retrieved `keystream block`. He replaces his C-RNTI with victim's one[8] and immediately sends the spoofed message to the network. The network receives the spoofed TAU request message

[7] Careful readers will argue that why the attacker needs to wait for second retransmitted TAU when he can create spoofed message at step ②. We do so to avoid the victim device transitioning back to *registered* state from *TAU init* state when the TAU timer T3430 expires while the spoofed TAU is being processed at the network. That can invalidate our attack in which the device initiated TAU rectifies the location identity

[8] C-RNTI spoofing is necessary so that LTE base station forwards the attacker's spoofed message towards victim's S1AP connection.

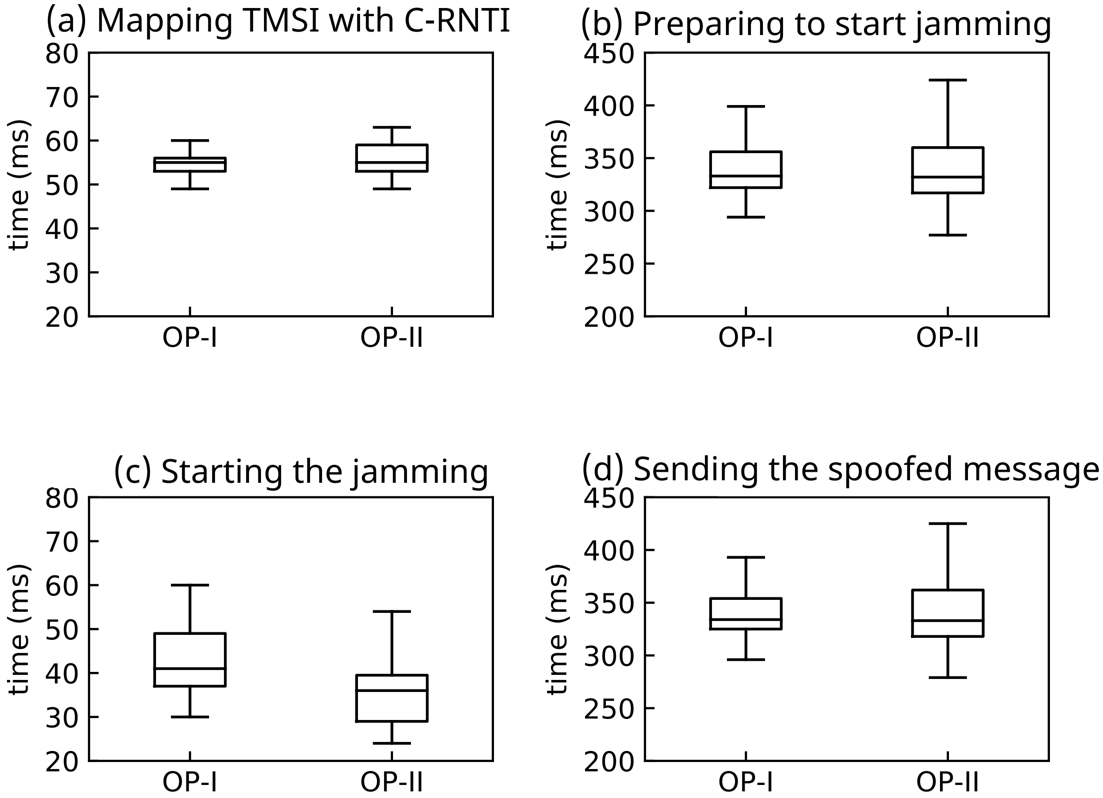


Figure 4.4: Error margin (min, max, avg, and std) for different experiments. Time between (a) RACH Request to RRC Connection Request messages; (b) RRC Connection Request to Security Mode Command messages (c) Authentication Response to Security Mode Complete messages; (d) Security Mode Complete (5th try) to TAU Request (3rd try).

while it was waiting for TAU Complete message from the device (as the network sends TAU Accept message after receiving TAU_2). According to LTE 3GPP standard, the network aborts previously received TAU message and processes the newly arrived message with different location identity (i.e. location information element). It has been stated in LTE NAS specification (3GPP TS 24.301 [3GP13b]):

"*If one or more of the information elements in the TRACKING AREA UPDATE REQUEST message differ from the ones received within the previous TRACKING AREA UPDATE REQUEST message, the previously initiated tracking area updating procedure shall be aborted if the TRACKING AREA UPDATE COMPLETE message has not been received.*"

The network decrypts the attacker originated TAU message and checks the integrity of the message. As the message contents were modified by the attacker, the TAU request fails the integrity check. The network finds that this is a special NAS message (4.4.4.3 Integrity checking of NAS signaling messages in the MME [3GP13b])) and it should be processed when the device fails the integrity check (*vulnerability 1* in Section 4.3.1.1). However, before accepting the message, the network successfully authenticates the victim device (by initiating the Authentication procedure), and sends TAU Accept message to the victim device. The victim devices reply with TAU Complete message to network that registers the spoofed device location identity in its database.

Experiment results To make this step successful, the network must receive the attacker's spoofed message before the TAU Complete message arrives from the victim device. This is the time between receiving Security Mode Complete and TAU Complete messages (a device response to TAU Accept message of TAU_2). From Figure 4.4(d), we can see that the attacker has on average 370msec (15msec STD) to prepare and send its spoofed message to the network. For validating the impact of the spoofed message we modify the non-volatile memory of the LTE modem and used Qualcomm's service-programmer tool (QPST Service Programmer) [qps], and AT-command tool (TeraTerm) [ter] to send the spoofed message.

4.3.1.3 Attack Damage

The consequence of our attacker is that the network updates the victim device location to erroneous tracking area. When the victim device enters in the idle state, it releases the RRC connection. The device relies upon the paging message from the network for the notification of its data packets during its idle state (e.g., if someone sends a text message or voice call to the victim). Because the attacker has registered the victim device on the wrong location by hijacking TAU procedure, the victim device does not receive the paging message. Hence, the victim device remains unreachable for its incoming voice and data traffic.

Constraints To realize the attack, the device must transition to idle state after performing the TAU procedure. The maximum time the victim device remains under attack is the time until it performs periodic TAU procedure (default value of 54 minutes). Note that, other LTE procedures such as

Service Request procedure or VoLTE call establishment do not have any impact on our attack (i.e. they do not shorten the attack time).

Extending the attack period The attacker can easily re-launch the attack to keep the victim device under attack even if the device updates its location through periodic TAU procedure, establishing a CSFB call, or even rebooting. After launching the attack for the first time, the attacker periodically pages the victim device by initiating a call towards him. If the attacker's call generates the paging message, it means the victim subscriber has recovered from the attack. The attacker then re-launches the attack by following steps ① to ③, and keeps the victim subscriber under attack.

4.3.2 Designing LTE Service Outage Attack

We extend our location hijacking attack to bring more serious attack. In this variant of the attack, the attacker sends Detach Request message (with cause power off) instead of sending the spoofed TAU request message at step ② to the network. There are two scenarios that occur at the network. First, the network receives the device de-registration request in the middle of ongoing TAU procedure (i.e. the network is waiting for TAU Complete message from the device). Second, the detach request being sent by the attacker is bound to fail the integrity check at the network. The 3GPP standard explicitly discusses both these cases in LTE NAS standard [3GP13b]. The first case is defined as an abnormal case for TAU procedure that requires the network to abort the TAU procedure and to process the Detach Request message from the device. It has been stated in [3GP13b]:

"*If the device receives a DETACH REQUEST message before the tracking area updating procedure has been completed, the tracking area updating procedure shall be aborted and the detach procedure shall be progressed.*"

While progressing the detach request message, the network finds the message has failed the integrity check. This is our second scenario and the 3GPP standard requires the Detach Request message (with cause power off) must be processed even of the message fails the integrity check (i.e. our *vulnerability 1* in Section 4.3.1.1). LTE NAS specification states [3GP13b]:

"*The procedure is completed when the network receives the DETACH REQUEST message. On reception of a DETACH REQUEST message indicating "switch off", the MME shall delete the* *current EPS security context.*"

We must point out that this special case only applies to Detach Request with reason power-off; otherwise, the network proceeds with the tracking area updating procedure first before progressing the detach procedure.

No LTE service When the network receives Detach Request message with cause power off, it re-authenticates the victim device first and then releases the device connection by deleting device sessions and freeing its IP address. The device (being unaware of its network registration has terminated) sends Service Request message (when it has some data to send or call to initiate). On receiving the Service Request message from the victim device, the core network rejects the request with error cause #43 (Invalid EPS bearer identity). On receiving the Service Reject message with error cause #43, the device enters into *deregistered* state, according to 3GPP NAS specification [3GP13b] that states:

"*The UE shall abort any ongoing ESM procedure related to the received EPS bearer identity, stop any related timer, and deactivate the corresponding EPS bearer context locally (without peer to peer signaling between the UE and the MME).*"

Now the victim needs to manually register the device with the network (by rebooting the device or by turning on/off the device airplane mode), otherwise LTE service remains unavailable.

4.4 Proposed Remedies

LTE base station and the core-network entities are 3GPP compliant and any vendor-specific implementation, conflicting with the LTE standard, may fail inter-operability between devices and the network functions. Therefore, our proposed remedies should be 3GPP standard compliant. Further, we seek to mitigate discussed vulnerabilities without introducing any other security loopholes.

To achieve these goals, our solution is based on the following two security design principles.

1. Make key installation and count reset procedures atomic (**P1**); i.e., either both succeed or none

2. Message failing the integrity is always re-executed (**P2**)

4.4.1 Remedies

Bounding `key` installation and `count` reset procedures: One of the root causes of control-plane attacks is the disjoint execution of key installation and count reset procedures. To address this, we bound LTE NAS Authentication (that installs the `key`), and NAS Security Mode Command (that resets `count`) procedures. That is, we perform LTE Authentication procedure whenever the Security Mode Command procedure fails (making security procedures atomic). In LTE Authentication procedure, the network sends Authentication Request message by starting timer T3460 (default value of 6s). The timer is stopped when the network receives the Authentication Response message from the device. In our solution, we stop T3460 when the network receives Security Mode Complete message from the device instead of stopping at Authentication Response. It means, the Authentication procedure fails if the device Security Mode Command procedure fails; hence bounding these two procedures. Our approach addresses *vulnerability 1* based on principle *P1*.

Enforcing integrity protection for all signaling messages once security has been established: The other root cause of control-plane attacks is that certain messages (i.e. TAU and Detach Request) are partially accepted even if their integrity check fails. Although, the network authenticates the device afterward but does not validate whether the received signaling message was indeed originated by the authenticated device or not. We mitigate this vulnerability by enabling the device to not accept any signaling messages failing the integrity check if the security association has already been established. Instead, the network *rejects* the message and re-authenticates the device. We should point out that present 3GPP specifications generate integrity failure message response for selected signaling procedures. To provide the integrity check failure feedback for all types of signaling messages, we propose that the network should reject the signaling message with the error cause # 9 (UE identity cannot be derived by the network). On receiving this error message, the device re-registers with the network after executing both authentication and security mode procedures. Our standard compliant solution may arguably delay LTE service for a couple of seconds, but it enforces LTE security at all times; hence mitigating *vulnerability 2* based on principle *P2*.

4.4.2 Security Analysis through Prototyping

We provide the security analysis of our proposal by developing a proof of concept prototype without creating interaction between victim and attacker. We use AT commands to take certain actions emulating the network enforcing above principles to mitigate vulnerabilities 1 to 3. Although there exists hundreds of AT commands, only a few have the privilege to execute over commercial handsets. We create our prototype by using those AT commands which our program can execute over commercial phones (such as Google Pixel or Samsung Glaxy devices).

In our experiment, we check whether the subscriber device is under jamming attack or not. If the signals are jammed to launch key reinstallation attacks by resetting `counts`, we re-activate LTE bearers that re-establish the security by renewing `key`. When the device makes a voice call through CSFB, our program checks for LTE registration by running "*at+creg?*" and "*at+cgdcont?*" commands. "*at+creg?*" tells whether the device is PLMN registered and if true then whether it is registered with LTE network or not. The "*at+cgdcont?*" outputs the IP and APN name that explains with which cellular radio access technology the device has camped-on. For example, fast.tmobile.com tells the device is registered with LTE APN over T-mobile carrier network. Thereafter, our program sends "*at+cgdata="PPP", 1*" command to establish the data connection with the network. If the data request is not entertained, the device AT command returns an error. It means the device data connection request has failed due to jamming. On receiving the error message, our program waits for 2 seconds before running "*at+cgdata*" command again. If the error persists then our approach is to renew the `key` by re-activating LTE service. We run "*at+cops=2*" immediately followed by "*at+cops=0,1*" to force the device to reselect LTE network and perform re-authentication procedure. In short, our approach re-activates the security as needed and does not introduce any other security vulnerability.

CHAPTER 5

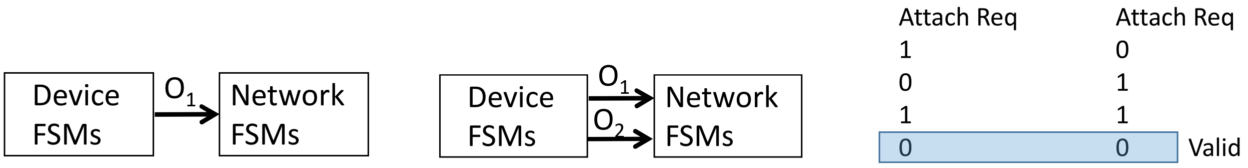
Performing LTE Vulnerability Analysis through Testing

In this chapter, we discuss LTE testing based vulnerability analysis. First, we discuss the importance of LTE testing and its limitations. Because the device was never tested for all possible practical use cases, some of the vulnerabilities go undetected. To address this, we advocate algorithm based complete LTE testing approach. Through our testing based methodology, we have unveiled new security vulnerabilities and we present them as our findings.

5.1 LTE Testing

We now briefly review the importance of LTE testing, its limitations, and applications beyond LTE.

Importance of LTE testing LTE testing is a key factor to make the LTE technology a great success. Operators and network vendors not only require the manufactured device (called User Equipments (UEs)) to follow the standards but also require each operating in the LTE network to be standard compliant. 3GPP defines how the phone and the network should behave in each operation scenario. These operational settings are formalized in the LTE protocol conformance testing specification [TS314], which thus specifies a number of test cases to validate LTE protocol operations. These test cases ensure that all LTE protocols (e.g., radio resource control, mobility management, session management, connectivity management, transport, and tunneling protocols) work correctly in every operational situation. These test cases are validated through message exchanges between the device and a Network Simulator (NS). Both the device and the NS implement their Finite State Machines (FSMs), and ensure the correctness of each test case via the expected state transitions. Figure 5.1a illustrates a test case execution scenario, in which the device FSM generates an output message (O_1) and the NS consumes the message to verify its correctness.



Attach Req	Attach Req	
1	0	
0	1	
1	1	
0	0	Valid

(a) One protocol interaction

(b) Two protocols interaction

(c) Output message combinations

Figure 5.1: LTE test execution scenarios between device and network

Current LTE testing practices and limitations The current LTE Testing practices are rather ad hoc. They do not follow a systematic approach to testing, nor ensure completeness. Our study shows that, although the LTE standards discuss LTE testing in a given operational situation as well as abnormal device behavior, these test cases do not cover every possible test scenario. This is mainly because the current practice has focused on a single protocol execution. When cases for multiple protocol executions arise, not *all* interactions among these multiple protocols are tested. Figure 5.1b illustrates the case when two output messages (O_1 and O_2) are fed to NS, which validates the correctness of their interactions. Take the LTE *Attach Request* procedure as an example. In this procedure, two EMM (EPS Mobility Management) protocols interact through the message of Attach Request. It generates 4 possible output message combinations (as shown in Figure 5.1c). (0 , 1) and (1 , 0) denote the absence of a message from one protocol, thus making it the test case of a single protocol interaction[1]. (1 , 1) represents the message from both protocols, indicating a case of two protocol interactions. (0 , 0) is a test case that corresponds to the absence of messages. The absence of Attach Request message triggers timeout that may affect the interacting protocols[2].

We further discover that LTE testing treats each test case in an isolated manner based on conformance to LTE test standards. A test case does not share its execution information with a later-executed test case in the series of tests. This leads to inefficiencies with repetitive execution of test steps for a new test case to bring device into the certain state (e.g., idle or connected state, etc.).

[1] For example, device protocol generates one EMM message to NS.

[2] Such interactions can be between the same protocol (e.g., two EMM message exchanges) or between two different protocols (e.g., EMM and ESM (EPS Session Management) protocols).

For example, almost all LTE *Attach* related test cases require the device to be switched *off* and then *on*, so that the device can initiate an *Attach* test case. However, such repetitive *power off/on* steps execution can be avoided if the next test case uses the knowledge from the previous test case that has successfully completed these steps.

We also looked into commercially available testing solutions provided by two leading LTE test equipment vendors [ANIb], Anritsu [ANRa] and Anite [ANIa]. Their testing data sheets [ANRb] [TES] and demos [ANIc] show that their test equipments do not make any changes to the implementation guide provided by 3GPP [3GP]. They treat each test case individually and only execute those test cases that are provided by the 3GPP. Their approach to testing is inefficient as they isolate test cases and do not transfer the previous test case execution knowledge to the followup one. We further conclude that LTE chipset manufacturers, device vendors, and network operators perform test cases similarly to how test equipment vendors execute them.

The above observations show that the testing community is focused on telecommunication standards and textbook testing design. We make a case for a paradigm shift to a system design approach from computer science when looking into the LTE testing problem. We aim to design and develop a testing system that treats individual test cases as the building blocks in a coherent testing system framework, along with algorithms to improve efficiency and ensure completeness.

Testing beyond 4G LTE Researchers who are developing systems for the next-generation wireless networks often face the challenge of verifying their designs. Testing plays a crucial role in making their design practical and deployable in reality. Through testing, researchers ensure their systems to adapt properly and quickly under scenarios that are both common and corner-case usage settings. Although we focus on 4G LTE testing, our methodology is generic and applicable to other technologies. Two examples are 5G New Radio (NR) testing and cellular Internet of Things (CIoT) testing. mmWave with 5G NR will introduce new test challenges where devices are using transceivers with integrated phased array antennas. These challenges include repeatability, configuration, and coverage, as well as testing accuracy, test time, and cost. These challenges can be addressed through our proposed testing methodology. Similarly, 5G cellular IoT solutions, including LTE-M and NB-IoT, require extensive testing before their deployment. Their test cases are related to network integration, coverage, battery consumption, and more. Our testing methodology can apply to these emerging technologies.

5.2 The Need of Complete LTE Testing

We next identify limitations of the current practice on providing a complete list of test cases and present our approach.

5.2.1 Limitations and Challenges

We take different test cases as a sequence of message exchanges, instead of a particular test scenario. The LTE test case procedure involves an exchange of messages between the device and the network. The sequence of these messages would inform whether the test case has passed or failed. For example, in the device *Attach* test case, the device and the network exchange a number of messages related to Radio Resource Control (RRC), Security Mode, Authentication, and Packet Data Network (PDN or IP) connectivity. We view the LTE testing as the interactions between the device and the network. If *all* such interactions (*in any order*) are successful, we have completed *all* tests; otherwise not. To provide a complete list of cases, we seek to generate a list of tests that include *all* possible combinations of these messages. Given n device output messages, there are 2^n possible tests, which are practically infeasible to analyze. The issue is to obtain the complete list of test cases without enumerations.

5.2.2 Our approach

We seek to validate LTE protocol interactions to ensure that all combinations of protocol messages are assessed for correctness.

Testing as protocols interaction The purpose of device state transitions and their interactions is to generate a message for the network simulator. Hence, LTE testing looks into message exchanges between the device and the network. We can thus reduce testing efforts on device protocol

FSMs by simply looking at the output message (O_1) that the device FSMs have produced, as shown in Figure 5.1a. If the message produced by the device is the one that the NS is expecting, then the internal FSM state transitions and interactions at the device were correct. If the NS has received an unexpected message from the device, we debug those states that have produced that output message, instead of traversing all states of the device FSM(s). Therefore, by checking device output values, the state transitions can be found, through which a test case can be defined.

Don't care outputs We provide a complete set of test cases that explore all possible output message combinations produced by the device FSMs when two protocols interact between the device and NS. For n possible output messages produced by two protocols running at the device, one is required to test 2^n message combinations.

We reduce the output message combinations for two protocol interactions. We do not generate test cases for those combinations that were never produced by the device FSMs. We annotate these combinations as don't care outputs. To find such outputs, we traverse the device FSMs in the reverse order and check whether the corresponding output is valid or not. Finding don't care outputs, however, is practically infeasible when the device FSMs have too many states.

Compressing device FSMs We can quickly find the don't care outputs if we skip a few states in the device protocol FSM. We can even skip visiting a portion of FSM that might have constraints on message combinations. This motivates us to compress device FSMs by merging states from FSMs.

Finite automaton To compress FSMs, we model these FSMs as a finite automaton. Similar to FSMs in the finite automaton, we have a start state, a final state, a finite set of states, a set of transitions, and a transition function. Some states are deterministic while others are not. If for each pair of states and possible transitions, there is a unique next state (as specified by the transition function), then the finite automaton is deterministic, i.e., Deterministic Finite Automaton (DFA); otherwise, the finite automaton is non-deterministic, i.e., Non-deterministic Finite Automaton (NFA).

Converting NFA to DFA Our goal is to reduce the number of states and to let FSMs run in polynomial time. We find that reducing the number of NFA states (by removing unnecessary states) is shown to be NP complete [GH07]. Moreover, running time for an NFA is $O(n^2m)$ compared to $O(m)$ in case of DFA, where n is the number of states, and m is the number of *identical* transition conditions [YP11] [Sip98]. This is because NFA has n possible next states compared to DFA, which has only 1 path to the next state for a given transition. This motivates us to convert NFA into DFA.

Algorithm 1 Selected NFA states to DFA states conversion

```
input: = {nstates, trans_cond, trans_func, curr_state, next_state}
Call procedure Reverse-Edges()
procedure NFA-TO-DFA-PROCEDURE
    while nstates are not visited do
        if curr_state transitions to two or more next_state then
            if trans_func takes only one trans_cond then
                for all next_states do
                    dfa_state $\cup$ next_state
                divert edge arrows from next_states to dfa_state
                add edge from dfa_state to curr_state
                add trans_cond to the edges
Call procedure Reverse-Edges()
procedure REVERSE-EDGES
    if curr_state transitions next_state(s) on trans_cond then
        for all next_state(s) do
            swap (curr_state, next_state(s))
            reverse edges arrows
            keep trans_cond and trans_func
```

5.3 LTE Testing as Finite Automata

Overview of our solution We model the LTE testing as the problem of finding don't care output combinations in Finite Automata. All possible output combinations *minus* don't care outputs are the complete list of test cases. To find don't care outputs efficiently (i.e., the running time of FSM interactions remains linear), we first convert NFA into DFA. We propose a novel algorithm that reduces FSMs states by converting the selected NFA states to DFA. We further minimize those FSMs states through the DFA minimization procedure. We can do so because two or more DFA states can be equivalent, where these states migrate to the same next states upon the same transition condition. We merge these equivalent states and get compact representations for the LTE protocol FSMs. To merge as many equivalent states as possible, we introduce a new definition of states equivalence. Using the LTE domain knowledge, we argue that a number of FSM states have LTE timing and protocol constraints. These states are equivalent and merged because in reality they never occur together. In this regard, we propose a novel DFA minimization algorithm that converts non-equivalent states to equivalent states and merges these states.

5.3.1 Reducing FSM States

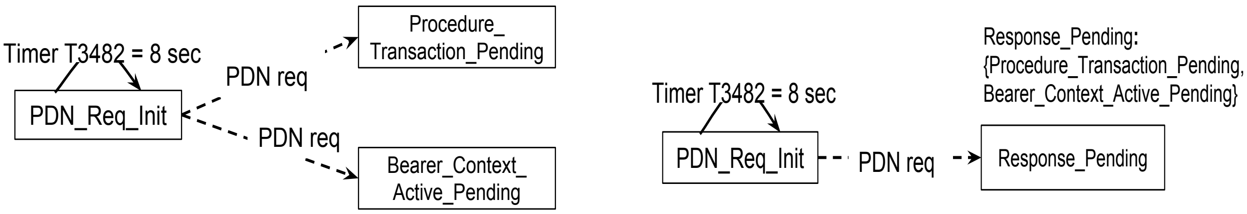
We first convert the NFA states to the DFA states to reduce the total number of states in FSMs.

Inefficiencies in NFA to DFA conversion The Robins and Scott algorithm [RS59] is the best known scheme to convert NFA to DFA [Sip06] [Moo71]. Such a conversion is made through power set construction. The DFA is obtained through a state set 2^n, the power set of n, containing all subsets of the original NFA state set n. The exponential number of DFA states are due to the degree of non-determinism of the current state. The current state can transition to a number of next states for n possible transitions. Through power set construction – which is practically inefficient – all possible states are recorded. It has been shown [SY96] [Man73] that the number of states in DFA dramatically increases when more than one transition conditions are considered. It has been proved that the maximum number of states in DFA reaches $2^{\frac{n}{2}}$, $2^{\frac{2n}{3}}$, $2^{\frac{3n}{4}}$, and 2^n when the number of transitions are 2, 4, 8, and n, respectively.

Converting selected NFA states to DFA We next propose NFA to DFA conversion (Algorithm 1) that converts only those selected NFA states to DFA and does not generate the power set of NFA states. In our algorithm, we focus on two aspects: (1) visiting the states from the output (in reverse), and (2) converting those NFA states to DFA with *only one transition condition* (which is common in LTE, as we will show later).

For (1), we swap the current state with the next state (Step 16) and reverse the edges, while keeping transition condition and function unmodified (Steps 17-18). Thereafter, the FSM can be traversed in reverse and the initial FSM state can be reached that has generated final output value. For (2), our algorithm processes only those states that have exactly one transition condition to the next state. We first check whether the current state is indeed an NFA, i.e., it has more than one next states on a given transition condition (Step 5). Note that, when the current state has only one next state, the current state is already deterministic and the algorithm moves to the next state (Step 4). When the first *if* condition (Step 5) yields true, the algorithm checks whether the state transitions are carried through one transition condition or not (Step 6). If this condition is satisfied, NFA to DFA conversion begins. In such a conversion, our algorithm merges all next states for the current state and creates one DFA state, which is a set of merged states (Step 8). Thereafter, it changes the edges such that all incoming and outgoing edges of all merged next states are diverted to the newly created DFA state, and a transition edge is inserted between the DFA state and the current state (Steps 9-11). Finally, we reverse the edges from the current state to the next state(s) before we take further actions on DFA states. Note that this is an important step before we further reduce DFA states (section 5.3.1.1). If we do not reverse the edges, reducing DFA will guide us NFA back again. This has been shown in Brzozowski's algorithm [CKP02], where DFA minimization converts the input DFA into an NFA by reversing all its arrows and exchanging the roles of the current and next states.

States with one transition condition are common in LTE We next show that one transition condition states are common in LTE; they are related to LTE timers and functionalities. The most common examples are timers that handle *reject* conditions. The standards mandate that, if the device request is *rejected* for a certain number of times, the current state should migrate to a particular next state (with the *reject* transition condition); otherwise, the current state should move (with the same *reject* transition condition) to another different next state. Similarly, most LTE functionalities also have NFA states with the exact one-transition condition. The transition condition remains the same for different actions, such as cell search, camping on cell, multiple or single



(a) ESM NFA

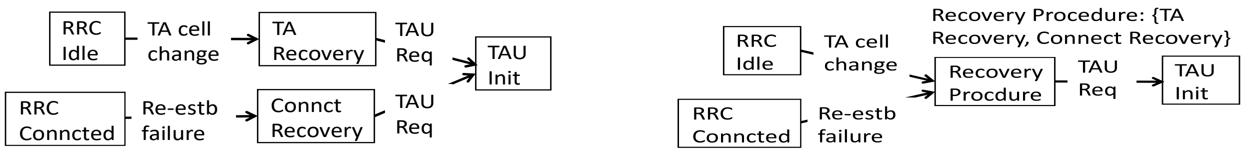
(b) ESM DFA

Figure 5.2: One-transition NFA states are converted into DFA

bearer request, and priority related features. For example, if the device serving the cell state meets a certain threshold value, it should move to the *intra-freq measurement state*; otherwise, it migrates to the *inter-frequency measurement state*. The transition condition for both states is *measurement*.

Discussion Our algorithm processes those states with only one transition condition to the next state. It does not construct the power set of states. The complexity of our algorithm is linear where the *while()* loop (Step 4) iterates over a limited number of states (the finite set of states is provided as an input to the algorithm). The *for()* loop (Step 7) also iterates over a constrained number of next states because our algorithm executes this step only when the current state moves to the next states upon single transition condition. Note that, the number of next states can be found by looking at the number of arrows coming to the current state (note that we are looking at arrows but not tails, as we are processing in reverse).

Example We now provide an example where our algorithm converts an NFA to a DFA, as shown in Figure 5.2. Two different tests (case numbers 10.5.1 and 10.5.1b in [TS314]) move from the current state of *PDN_req_init* into two different states *Procedure_Transaction_Pending* and *Bearer_Context_ Active_Pending* states, respectively, using one transition condition *"PDN req"*. Both cases are requesting PDN from the network. The first case requests an additional PDN for its uplink (UL) data, whereas the second test requests an additional PDN but using NAS signaling low priority. In 10.5.1b, the device establishes a dedicated radio bearer associated with the default EPS bearer context, before sending an additional PDN connectivity request. This is why the device moves to the *Bearer_Context_Active_Pending* state.



(a) DFA states before minimizing

(b) DFA states after minimizing

Figure 5.3: Two equivalent DFA states are combined

5.3.1.1 Minimizing FSM States

In DFA minimization, two or more equivalent DFA states are merged and represented by one state. Two or more states are said to be equivalent if these states migrate to the same next state upon the same transition condition.

DFA minimization overview The Hopcroft algorithm for DFA minimization [Hop71] is the best known solution to minimizing a DFA [AMR07] [GRA13] with its complexity of *O(nlogn)*. The key idea is to partition the states when two states are not equivalent. At first, all states are placed into one partition and thereafter the partition in refined. The states that are not equivalent are removed from the partition, whereas the equivalent states are merged. The key ingredient of the algorithm lies in how partitioning is done. The trick is to not partition on already visited transition conditions until the partition is further split. In that case, the algorithm only checks one of the two new partitions.

DFA minimization of mixed NFA-DFA FSM We propose a new algorithm for DFA minimization, but use the partitioning procedure from the Hopcroft algorithm. Unlike the Hopcroft algorithm that works on DFA FSM only, our algorithm reduces FSM which is a mix of NFA-DFA. Indeed, like many other FSMs, LTE protocol FSMs are the hybrid of NFA and DFA states, where we have also converted few NFA states into DFA states (algorithm 1). From the Hopcroft algorithm, we find that, as the number of equivalent states increases, the number of states in FSM decreases (as the equivalent sates are merged). Although we cannot increase the states equivalence in FSM, we can obtain similar results by merging two or more states that have constraints on each other. We use our LTE domain knowledge and introduce protocol states and timing constraints.

Protocol state constraints In LTE protocol FSMs, some states have constraints on others.

A few of such constraints have been described in the LTE NAS standard specification (Figure 5.1.3.2.2.7.1: EMM main states in the UE) [3GP13b]. For example, when the device is at the idle state and plans to send/receive voice/data packets, it initiates the *Service Request (SR)* procedure and moves to the *Service_Request_Init* state. While the *SR* procedure is ongoing, if the device changes its location and performs the *Tracking Area Update (TAU)* procedure, it cannot do so. This is because the *TAU* procedure can only be initiated from the device state of *EMM_Registered*. We can say that all *TAU* and *SR* related states have constraints on each other, where the device cannot be at both states concurrently and we can merge these states. As a result, our modified FSMs will never produce those output values that capture *TAU* and *SR* interactions (which are don't care outputs). In short, we list all such protocol constraints and merge them.

Timing constraints Similar to protocol state constraints, there are timing constraints between states. For example, to initiate the *normal TAU* request message, the device must have moved to a different LTE base station cell (that is, its location must have been changed). Further, the LTE base station can only correspond to one tracking area, because *System Information Block-1 (SIB1)* broadcasts only one tracking area code for a cell. Therefore, one cell cannot belong to two different tracking areas. This example illustrates the timing constraint situation where the *normal TAU* procedure can only be initiated after the device has roamed to a different cell in a different tracking area.

Algorithm We now explain our algorithm of minimizing DFA states, as described in Algorithm 2. At the start, there is only one partition that contains all states in an FSM (Line 6). Like the Hopcroft DFA minimization algorithm, our scheme iteratively reduces partitions by removing non-identical states. However, at each iteration, it only takes action when the state is deterministic (Line 10); otherwise, it splits the partition and removes the non-deterministic state (Lines 8-9). Before acting on deterministic states, it ensures that (1) for state p, there is no equivalent state, i.e., state p does not belong to the current partition ($p \rightarrow$ other partitions); and (2) there is no state in the current partition that has constraints with state p (i.e. $p \notin constraint$). If both (1) and (2) are false, it implies that the current state has equivalent or constrained state in the partition and *for()* loop (Line 7) moves to the next iteration. If either (1) or (2) is true, state p is moved out of the partition

(Line 11) and becomes a different partition (Line 13). Once the algorithm moves p out of the partition, it checks whether p belongs to the current state or the next state and updates accordingly (Lines 14-17). Note that it is possible that the merged states may not be fully connected to other states. Therefore, at each iteration, we make sure that all incoming and outgoing transition arrows of the merged states are updated accordingly (Line 18).

Algorithm 2 Minimizing DFA states

```
 1: input: = $\{$nstates, trans_cond, trans_func, curr_state, next_state$\}$
 2: procedure MINIMIZE-DFA
 3:     $p_1$ is sub-partition 1; $p_2$ is sub-partition 2
 4:     $constraint$, constraint vector
 5:     while no further partitions can be done do
 6:         parition_1, all set of states in FSM; parition_2, $\emptyset$
 7:         for each element $p$ of parition_1 do
 8:             if $p$ is non-deterministic then
 9:                 split(p, parition_1)
10:             else if ($p \rightarrow$ other partitions) OR ($p \notin constraint$) then
11:                 $\{p_1,p_2\}$ = split(p, parition_1)
12:                 if $p$ belongs to $\{p_1,p_2\}$ and $p \neq \emptyset$ then
13:                     parition_2 = $p$
14:                     if curr_state == $p$ then
15:                         $curr\_state_{min}$ = $p$
16:                     else if next_state == $p$ then
17:                         $next\_state_{min}$ = $p$
18:                     update trans_cond for $p$
```

Discussion Our algorithm does not incur additional complexity compared with the Hopcroft algorithm, which determines the state to be deterministic or non-deterministic in one step (Line 8). It simply checks that the current state must not migrate to more than one next states for a given transition condition. This implies that the state is NFA without even checking the next states. The step of splitting into non-deterministic states can be viewed as the state having no equivalence

Table 5.1: Summary of test cases – our procedure finds new legitimate test cases

Protocol Interaction	**Procedure defined by 3GPP**	**Tests cases defined by 3GPP**	**Procedure *not* defined by 3GPP**	**Total *missing* test cases by 3GPP**
ECM[3]and ECM	141	118	18	14
ECM and EMM	14	10	7	11
EMM and EMM	161	142	29	54
ESM and ESM	25	22	6	8
Total	**341**	**292**	**60**	**87**

behavior. Therefore, the splitting procedure (Line 9) does not add extra complexity. The step of checking the protocol state and timing constraints requires to know whether the current state being partitioned is part of the constraints or not. If it is part of a set of constraints, the algorithm checks whether the current partition contains those states or not. We make these two steps efficient by first logging all such constraints as a constraint vector. The vector is of fixed length and the number of constraints are not large (because these constraints are related to overall LTE functionalities, but not specific to states or timers). Therefore, checking this constraint can always be done in polynomial time. Furthermore, we avoid creating extra steps by merging constraint conditions with partition conditions.

Example We now show an example of three LTE test cases where our algorithm minimizes the equivalent DFA states. Three cases (test case number 9.2.3.1, 9.2.3.1.9a, and 8.5.1.4 in [TS314]) share part of the common procedure. As shown in Figure 5.3, these tests perform the *TAU* procedure but under different triggering conditions. In test case 9.2.3.1, when the device moves to a different cell in a different tracking area, it initiates *TAU*. However, in test cases 9.2.3.1.9a and 8.5.1.4, the device fails to recover from the radio link failures and needs to perform connectivity recovery. Once recovered, the *TAU* procedure is performed. Hence, we can minimize these DFAs by merging *TA Recovery* and *Connect Recovery* states (shown in Figure 5.3b).

[3] EPS Connection Management (ECM) involve signaling connection that is made up of two parts: an RRC connection and an S1_MME connection. We have included RRC test cases as part of ECM procedure.

5.3.2 Implementation

Our implementation includes 3GPP test cases, our proposed algorithms, and creation of FSMs and their representation as a finite automaton, the complete set of test cases by excluding don't care device outputs. Our implementation is highly modular for better code re-use.

Test cases and their execution For 3GPP test cases, we implement RRC, EMM and ESM cases as described by the 3GPP specification (36.523-1 [TS314] section series 8, 9 and 10). We prototype each test case by following the procedure described in the specification and run the test as a message sequence between the device and the network. The device and NS are two processes running on the Linux machine. The device generates a message for NS, where NS produces the response by following the standards. If the device does not receive a response from NS (which is the typical case in our protocol interaction testing), we mark it as vulnerable/missing case and manually confirm it with the 3GPP standards. Before each test starts, our program takes a set of configurations defined from a number of *config* files as required by the test case. We create different *config* files for different purposes. For example, *preamble.config*, *cells.config*, *timers.config*, *ie.config*, denote test start states, cell related config, timers and their values, and information elements with device capabilities, respectively.

Finite-state machine Each test case is executed such that the execution is captured as a transition between different states. Such an implementation choice is important to represent test cases as a finite automaton and generate new test cases for inter-protocol communications. The current states, next states, and transition conditions are *enum* type values. For each state, the set of valid state transitions are stored as a *multimap*. The transition function is an action that the current state performs and migrates to the next state. The action is basically a callback function that informs which steps should be performed by the device and for which next state (*ActionCallbackFunc callback = stateTransition[std::pair(current_State, next_State)]*). Using this logic, we can easily represent our FSM as a finite automaton. In the finite automaton, the current state is allowed to have more than one transition (DFA or NFA). We modify the Hopcroft algorithm code provided by Antti Valmari [hop], and define contradictory states as equivalent states.

Protocol interaction Protocol interaction is represented as test case collisions and their inter-

Table 5.2: Summary of novel findings.

Issue	Protocols	Problem	Root Cause	Impact
Detaching victims	ECM – EMM	Device can send non-integrity protected Detach with cause *power off*	The standard allows certain types of messages can be sent as non-integrity protected	Adversary can let victim device detach.
Service provisioning	ECM – EMM	Local EPS bearer context is deactivated without ESM signaling	The device fails to establish user plane radio bearers when ECM process at network is delayed	EPS bearer context deactivated.
Skipping integrity	ECM – EMM	TAU message without integrity protection is accepted	TAU due to an inter-system change in idle mode is accepted by the MME even without integrity protection	The device reports wrong location to network.
Privacy leakage	EMM – ECM	After 4 retries from MME, the GUTI reallocation procedure stops	MME does not mark the device which has failed to perform GUTI reallocation procedure as vulnerable.	Using old GUTI compromises user location.
Null integrity	EMM – EMM	2^{nd} attach is processed by MME whose IE differs from 1^{st} attach	The device capability related information element (IE) in the Attach Req differs from the ones received earlier	Device attaches as non-integrity protected.
Barring to Attach	EMM – EMM	Processing Attach request without receiving Identity Response	The MME processes the Attach Request while waiting on Identity Response message from UE	Sending Attach instead of Identity Req bars UE.
Inconsistent states	EMM – EMM	Device proceeds Detach procedure whereas MME proceeds TAU	Before the detach request is received at UE, UE initiates TAU procedure. MME aborts detach and proceeds TAU	MME and Device states are inconsistent.
TAU is blocked	ESM – ECM	PDN procedure is blocked by RRC reconfiguration (doing TAU)	Bearer Modification and RRC-Reconfiguration with TAI change collide. TAU is blocked by earlier procedure	UE will end up keeping invalid tracking area.
Unauthorized connection	ECM – ECM	UE keeps radio connection for rejected RRC request	When the user identities are not found at EPC, the RRC req is rejected but UE remains camped on eNodeB cell	Connecting eNodeB with expired USIM cards.
Deadlock	ESM – ESM	UE and network both initiates Dedicated Bearer Procedures	Both UE and network has received/sent Activate Dedicated Bearer Request and enter into undefined behavior	Network and UE wait on each other request.

actions (with/without delay), where each test represents the same or two different protocols. Such interactions have to be tested on each valid output value (in any sequence). To implement this, the messages that a case produces during testing, as well as their corresponding responses, are placed in the queue. The execution of this test case is what the 3GPP testing standard mostly assesses (i.e., single protocol interaction). To find protocol interaction vulnerabilities, we let messages from two protocols interact with the NS, and observe their behavior. Each found vulnerability was manually verified with the 3GPP protocol specifications. For each vulnerability, we propose a new test case that describes the procedure (with the detailed steps) and the fix (the expected behavior).

5.3.3 Analysis

Once we have reduced the device-side FSMs and identify don't care output values, we can generate the *complete* test cases for LTE protocol interactions. We next provide analysis on test completeness.

Our procedure generates 30% more test cases compared with the test cases defined by 3GPP. Our result is summarized in Table 5.1. We also find that 60 protocol interactions are not specified

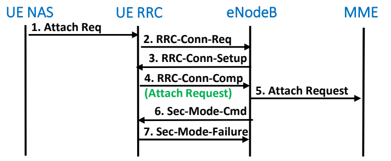


Figure 5.4: RRC integrity and ciphering is not applied

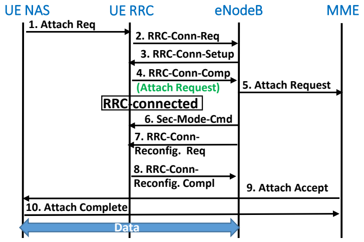


Figure 5.5: Data transmission starts without activating RRC security

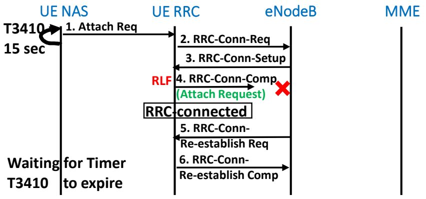


Figure 5.6: Re-Attach Request is delayed due to Radio Link Failure

by the 3GPP protocol standards. Table 5.2 shows the list of 10 new test cases and identifies new vulnerabilities in the 3GPP standard. The remaining 74 test cases expose relatively less serious issues, such as delay in service access, procedure repetition, downgrade to the lower-priority cell, the temporary loop between the device states (i.e., idle and connected states), two procedures temporarily blocking each others, etc. Now we provide a brief analysis on three novel vulnerabilities discovered by our test cases (other than those discussed in Table 5.2).

Integrity and ciphering is not enforced We discover that the device can skip the RRC ciphering and integrity protection even if both are enabled at the network. Such a scenario has been shown in Figure 5.4, where the *Attach Request* message is forwarded to MME (Mobility Management Entity) (Step 4) before the RRC security procedure starts (Step 6). If the device sends the *Security Mode Failure* message to eNodeB (i.e., the LTE base station), the 3GPP standard allows the device to communicate with the network without any protection, whereas the *Attach* procedure is allowed to complete. We verify this with the LTE RRC standard (Section 5.3.4 Initial security activation procedure in 3GPP TS 36.331 [3GP12b]). The standard mandates that after sending the *Security Mode Failure* message, the UE shall "continue using the configuration used *prior* to the reception of the *Security Mode Command* message, i.e., neither applies integrity protection nor enables ciphering."

To address this issue, we create a test case that makes 5 retries on the *Security Mode Command* message, upon receiving the *Security Mode Failure* message from the device. Once receiving the 5^{th} *Security Mode Failure* message, the eNodeB bars the device from camping on its cell for 60 seconds in our test case.

Sending data without RRC security success We find that, the device can send uplink data even if it has failed to complete the *RRC* security procedure, as shown in Figure 5.5. The device sends the *Attach Request* message as piggybacked with the *RRC Connection Complete* message and migrates to the *RRC-Connected* state. In the *RRC-Connected* state, the eNodeB sends its *Security Mode Command* to the device (Step 6). However, before receiving the *Security Mode Response* from the device, eNodeB establishes its Signaling Radio Bearer 2 (SRB) for the device's uplink/downlink data by performing the *RRC Connection Reconfiguration* procedure (Steps 7-8). Meanwhile, the *Attach* procedure completes (Steps 9-10); whereas the device does not generate the security mode response at all. We find that the device is able to send its uplink data even if the RRC security procedure did not conclude. This is indeed a loophole in the standard (see Section 5.3.5.3 Reception of an RRC Connection Reconfiguration procedure in 3GPP TS 36.331 [3GP12b] and Note 3). It has been stated "if the *RRC Connection Reconfiguration* message includes the establishment of radio bearers other than SRB1, the UE may start using these radio bearers immediately, i.e., there is no need to wait for an outstanding acknowledgment of the *Security Mode Complete* message."

Note that eNodeB does not have any timer linked to the security mode command and cannot resend the *Security Mode Command* message, if the response to the previous request is not made. To address this vulnerability, we add a test case to ensure the device has sent the security mode response (i.e., complete or reject).

Re-Attach request is delayed In this issue, the device registration procedure is delayed up to 15 seconds (the default value for timer T3410). Figure 5.6 shows that, although the eNodeB has failed to receive the *RRC Connection Complete* message piggybacked in *Attach Request*, the device enters the *RRC Connected* state. Such a failure of message arises because of the Radio Link Failure (RLF). Upon RLF, the device recovers its radio connectivity by performing the *RRC Connection Reestablishment* procedure (Steps 5-6), but does not resend the *Attach Request* message. Therefore, the NAS layer at the device times out for the Attach Request message and resends the request. One can argue that the *RRC Connection Complete* message sent over SRB1 will be recovered by the Radio Link Control Acknowledgement procedure (RLC ACK). However, the RLC procedure recovers the bit errors or retransmission failures over the wireless link, and does not recover the failure because of the device being out of sync with the eNodeB cell (RLF scenario). Moreover, the RLC ACK mode has small timer value (45 milliseconds as the default value [3GP12b]) and cannot recover the failure when the radio recovery procedure takes too long.

To address this issue, we create a test case where the EMM layer requests the RRC layer to notify its piggybacked request. If RRC is not able to deliver the piggybacked message within a couple of seconds, the EMM layer will resend the packet.

CHAPTER 6

Achieving High Availability of LTE Network Functions Virtualization

In this chapter, we first argue that when LTE network functions are moved over the cloud for scalability purpose, they bring LTE service unavailability. We propose a new way of LTE Network Functions Virtualization that ensures the mission-critical LTE procedures are timely executed and achieve high LTE service availability. Later, we discuss evaluation results of our approach.

6.1 Motivation and Problem Scope

Network operators are looking for Network Functions Virtualization (NFV) of LTE NFs to meet exponentially increasing traffic demands for their subscribers. LTE NFV replaces carrier grade LTE core network functions with software running on commercial off-the-shelf servers in a cloud data center. On the one hand, NFV reduces operational and capital expenditure at traditional LTE network operators; on the other hand, it opens the cellular network business to small network operators. Network operators can take advantage of dynamic load balancing, the resource elasticity and scalability that the cloud offers. Traditional LTE EPC NFs are static in nature and are connected, or chained, in a certain way that achieves the desired overall functionality or service that LTE network is designed to provide. These NFs exchange a number of control messages to execute a specific network event. For example, during *device Attach* event, MME obtains device security keys from HSS, authenticates the device, creates device session information at SGW and PGW. Then SGW and PGW establish data bearer connection with the device and configure a specific QoS profile. Thereafter, the device is said to be registered with LTE network. The delay or fail-

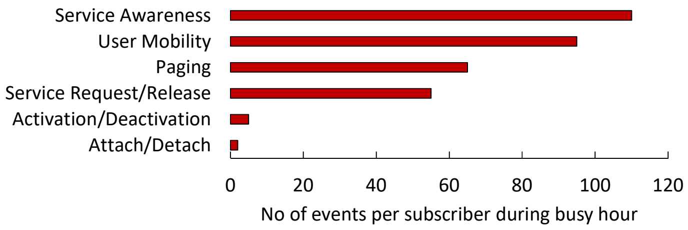


Figure 6.1: Number of events per user during busy hours

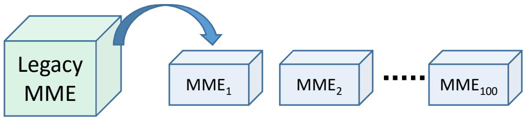


Figure 6.2: One legacy MME is decomposed into multiple virtualized MME instances

ure in one control-message results into complete event failure [3GP13b]. However, the software implementation of LTE EPC NFs over virtual instances cannot provide such guarantees.

Instance-based vEPC implementation is not right To understand deployment strategies of virtualizing LTE by research and industry communities, we surveyed ETSI (the LTE standardized body) documents on NFV [etsb], white papers from industry [nfv], and recent work on LTE NFV [BH14, CX15, HG15]. We find that most of these efforts talk about *instance-based* vEPC implementation, where vEPC NFs (MME, SGW, PGW, etc.) are installed as virtual machine instances. However, we stimulate the discussion that although such contemporary NFV architecture (which is highly distributed) suits the web-based applications' needs well, it may not be a good choice for implementing vEPC.

We address two major issues in the context of *instance-based* vEPC implementation, i.e. (1) signaling storm during peak hours, and (2) timely execution of mission-critical events.

Controlling network signaling storm LTE devices frequently interact with LTE network to execute their events. These events are *Device Attach, Service Request and Release, Handover, Paging, Bearer Activation, Modification and Deactivation, Detach Request*, and many others. Out of these, some events are executed more frequently than others. As shown in Figure 6.1, *Handover* event is executed at least 50 times more than device *Attach* incident during busy hours [NSN12] [Ba15]. Moreover, to execute one such event, different NF components interact with each other and generate a greater number of signaling messages. Some events produce more number of signaling message than others. For example, one *handover* event generates 32 signaling messages compared to *paging* event that produces only 6 signaling messages. When all events are combined from all devices during busy hours, a signaling storm is generated at EPC NFs. Therefore, we are motivated to provide a solution that controls the signaling storm at LTE core without restricting devices network access (the solution operational LTE networks use to control signaling storm [mmeb]).

Signaling messages within vEPC incur latencies LTE eNodeBs are connected with vEPC over a dedicated fiber link. The geographical separation between eNodeB and vEPC adds a constant round trip latency, which is about 10 milliseconds over a circuit of 1,000 km geographical distance [DIS]. However, latencies within vEPC are caused by VM hypervisor as well as switching contention and port queuing [BAMb] that can reach upto hundreds of milliseconds [MRD15]. This is mainly because for each event request from eNodeB, a greater number of signaling messages are exchanged among different VNFs in vEPC which may all suffer network delays. For example, a single *handover request* message from device generates up to 32 signaling messages within vEPC. Therefore, we limit our scope to signaling messages within vEPC which bring higher latencies compared to fixed latency over the eNodeB-vEPC fiber link.

Administrating mission-critical events Our preliminary study on LTE operational network discloses that during rush hours, average events completion time at EPC is significantly high, reaching up to 3 seconds (as shown in Figure 6.3)[1]. This higher latency directly affects user QoS experience, resulting from Voice over LTE (VoLTE) call drop and voice jitter, to affecting TCP based services (refer to Table 6.1.7: Standardized QCI characteristics in [3GP13a]). Therefore, in this work, we are also motivated to provide timely execution of mission-critical events, even during higher service load at LTE NFs.

Defining mission-critical events We categorize those events being mission-critical whose delay or failure has a direct impact on ongoing user services (i.e., voice, data, and multimedia services). These events are:

- *Handover event* that ensures seamless user traffic flow during user mobility.
- *Paging event* wakes the device from idle state when voice/data traffic is pending at LTE network.
- *Service Request event* provides on-demand network resources to the device.

[1] We gather LTE traces at device and ignore radio retransmissions (at both MAC and RLC LTE layers) and also excluded device and radio RTT from results.

Interestingly, these 3 events make-up 50% of all network signaling traffic [NSN12]. Therefore, by addressing these events, we not only ensure timely execution of user service sessions but also address highly occurring network signaling messages.

Assumptions This work neither assumes special data center network topology and high performance server boxes nor requires changes in LTE standard. We address timely execution of important events on commodity data center network (with no dedicated/express links) while obeying LTE standard to provide plug and play solution for any carrier network.

6.2 Challenges in Virtualizing LTE-EPC

We discover multiple challenges from our implementation of LTE-EPC virtualization, and from our study on LTE standard documents and virtualized network infrastructure.

6.2.1 On Data-Center Network Characteristics

NFV envisions the implementation of NFs as software-only entities that run over NFV infrastructure. NFV infrastructure consists of commodity servers that run Virtualized Network Function (VNF) over cloud platforms. In contrast to legacy LTE NFs implementation, NFV implementation introduces a number of changes. First, unlike traditional NFs which are connected over single hop, these VNFs may be located over multiple hops. Therefore, long queueing delay in switches introduces high latency [BAMa] [MRD15]. Second, during high data-center utilization, packet loss probability increases that can adversely affect traffic flows where the loss of an ACK may cause TCP to perform a timed out retransmission. Third, data-center network traffic exploits the inherent multi-path nature of data-center networks [MRD15] that causes out of order packet delivery. Fourth, the data-center network is designed to meet application deadlines which provide mechanisms to meet traffic flow deadlines (e.g. mice flows), rather than per packet guarantees [MRD15]. In short, data-center network is designed to meet Service Level Agreement (SLA) by protecting execution bounds on traffic flows. However, in LTE, service guarantees are made by timely execution of mission-critical events.

6.2.2 On Inter-VNF Delay

LTE-NFV framework provides the flexibility and network scalability by decomposing original NFs into multiple VNFs [etsb]. In order to ramp up the original capacity of NF, multiple VNF instances are desirable. For example, we need hundreds (if not thousands) of MME-VNF instances implemented over commodity servers in order to facilitate 10 million subscribers as supported by conventional MME function [eria]. As shown in Figure 6.2, legacy MME is decomposed into multiple MME instances, where each MME holds the profiles of a subset of customers. These VNF instances are distributed within data-center. Ideally, related EPC VNF instances (e.g. MME, SGW, PGW etc.) are placed *within the same rack* that eliminates network delays between two EPC NFs. However, during mobility, device switches to target eNodeB – connected to different MME instance. As a result, the device session migrates from its source MME to target MME during handover. Thereafter, new serving MME and rest of old serving EPC NFs end up residing at *different racks*. Now network delays play an important role on timely execution of network signaling messages. We find that LTE-NFV framework is not able to cope with varying delays among different VNF instances because of following reasons.

Expiry of a timer at any NF may lead to event failure LTE was designed for fewer EPC NFs which are directly connected over a dedicated fiber link. Therefore, in the legacy LTE network, the variation in RTT values is negligible. This motivates LTE network designer to use RTT for two purposes (1) path management, and (2) calculating message retransmission timer between a pair of EPC NFs.

Path management: As a matter of fact, all EPC NFs and the connection between them must always be active to serve users. To determine that a peer NF is active, the NFs exchange echo-request and echo-response messages [3GP13c]. This exchange of the echo-request and echo-response messages between two NFs allows for quick detection if a path failure occurs [3GP13c].

Retransmission timer: Echo-request and echo-response also help in calculating packet retransmission time at EPC NF. Retransmission timer is calculated based on RTT measurements (i.e. time between echo-request and echo-response) [3GP13c]. Although such timer value incorporates arbitrary RTT value delays, it does not include larger RTT value variations because network com-

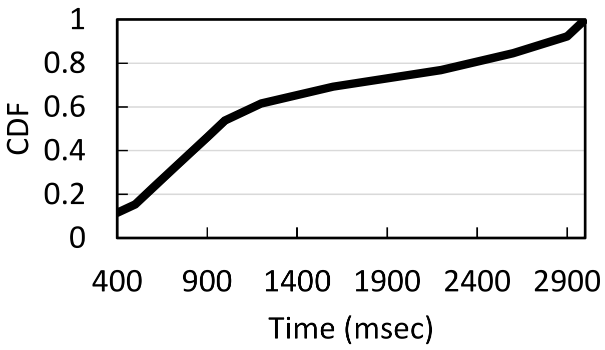


Figure 6.3: Event completion time (average) during busy hours in operational EPC network

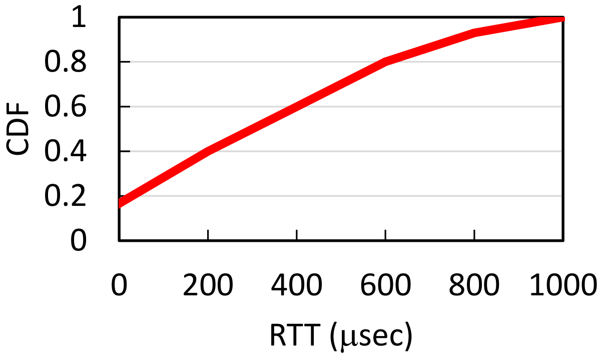
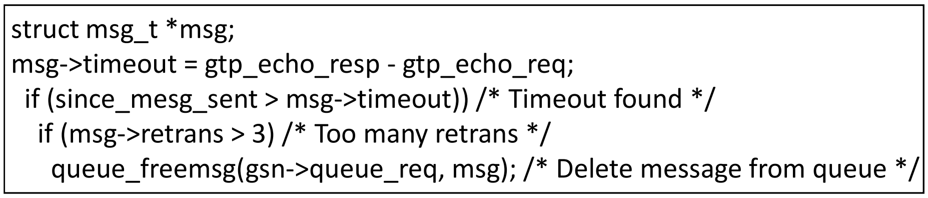


Figure 6.4: RTT fluctuation (average) in virtualized network

munication delay does not vary for one-hop legacy LTE NFs.

However, virtualized EPC implementation needs to address significant RTT variations. Data-center network's link redundancy provides multiple paths for each pair of NF [MRD15]. This means, for each RTT calculation, echo-request and echo-response packets may traverse through two different paths. This can potentially cause a significant variation between two subsequent RTT measurement readings. To make things worse, data-center network congestion can cause RTT spikes up to tens of milliseconds that make EPC retransmission timer calculation even harder. Figure 6.4 shows variation of RTT values in a virtualized network [MRD15]. The RTT varies from few microseconds to 1000 microseconds under normal network load. This 1000X RTT difference converts into hundreds of different timer values.

When an NF selects smaller timer value based on smaller RTT value, the signaling messages from that NF are unnecessarily retransmitted, as shown in Figure 6.5. This unnecessary signaling messages retransmission lead to the overall delay in event execution, and at times expiry of event-timer running at the device that results into event failure. We capture this through code snippet below.



```
struct msg_t *msg;
msg->timeout = gtp_echo_resp - gtp_echo_req;
  if (since_mesg_sent > msg->timeout)) /* Timeout found */
    if (msg->retrans > 3) /* Too many retrans */
      queue_freemsg(gsn->queue_req, msg); /* Delete message from queue */
```

Expiration of a timer has a domino effect For one event execution, multiple EPC NFs are chained such that one NF output is an input of second NF, and so on. For example, in *handover* event execution, signaling messages are exchanged between 5 different NFs (i.e. source MME, source SGW, target MME, target SGW, and PGW). Each pair of NF is running a different retransmission timer value. When one timer value expires, it produces a domino effect that causes expiration of timer to preceding NF. This has been shown in Figure 6.6, where source MME sends handover signaling message (e.g. *Forward Relocation Request message*) to a target MME. Target MME sends another handover message (e.g. *Create Session Request message*) to SGW. But even in the presence of mild network congestion, the response from SGW is delayed that results into the expiration of timer at target MME. Because source MME is waiting for a response from target MME, eventually the timer value at source MME also expires. This can potentially create a domino effect to chained NFs for *handover* event execution.

In short, EPC by design is not only sensitive to network delays but also does not tolerate even mild delay variance. However, it is challenging to provide both constant and smaller network delay in virtualized data-center network, where packets may face network congestion and take multiple packet traversal paths.

It's not about adjusting timer values: It can be argued that inter-VNF delay issue can be addressed by adjusting a timer value by choosing larger RTT value among all vEPC NFs. However, there are two issues with this approach (1) it requires modification in LTE standard such that the source code from different vendors remain compliant; (2) any congestion between two NFs can lead to arbitrary higher RTT value causing timeouts at the device. Note that vEPC should execute events within the tolerable amount of time, in which user should not perceive delays at LTE core network.

We should also clarify that engineering efforts, such as off-loading traffic from busy servers, do not work for vEPC. This is mainly because current data-center networks are designed using web services in mind; whereas vEPC (by EPC design rule) requires that all active user sessions remain within same EPC NFs, otherwise the GTP-tunnel [3GP14] between two LTE NFs break and incurs further delays.

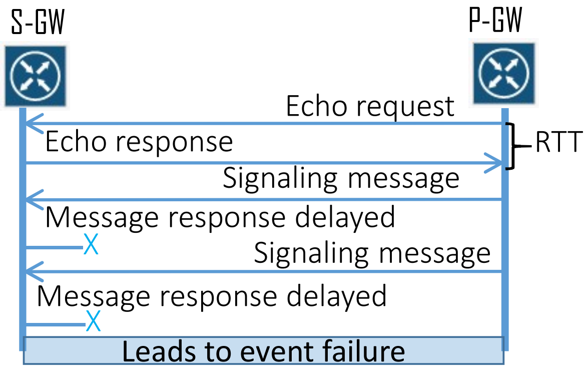


Figure 6.5: Signaling messages expire prematurely when timer based on RTT value is too short

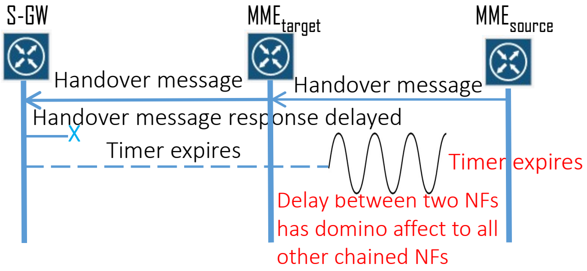

S-GW
MME_{target}
MME_{source}
Handover message
Handover message
Handover message response delayed
X
Timer expires
Timer expires
Delay between two NFs has domino affect to all other chained NFs

Figure 6.6: Expiry of timer between two NFs has a domino effect that leads to an event failure

6.3 System Design

We now describe our key insights from the legacy LTE network and then put forward our vEPC design.

6.3.1 Design Insights

1. Critical events execute in blocking-mode 3GPP standard requires the critical events should execute in blocking-mode. In blocking-mode operation, when one event is being executed, no other event can execute. For example, when *handover* is being executed the MME does not process any other request (such as the request for SGW or/and the MME relocation) and simply rejects such request [3GP13b]. Similarly, the device cannot report multiple events at the same time. In case the device has to report multiple events (e.g. *Handover, Service Request (SR), or Location Update*), then these events should be executed one by one. Figure 6.7 shows state transition diagram of different network events. Once a particular network event is initiated, no other event can be executed. As shown in Figure 6.7, on handover request, the device state transitions from "Registered" state to "Handover initiated" state. In "Handover initiated" state, the device cannot transition to any other state (such as location update initiated) until the handover is accepted or failed.

2. Network operations can be mutually exclusive A single NF is designed by combining multi-

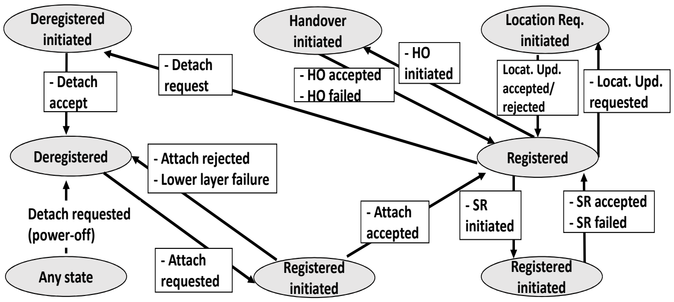


Figure 6.7: State transition diagram shows that only one event at a time can be executed

ple protocols. As shown in Figure 6.8, MME NF is made of S1AP protocol, MME core protocol, and GTP protocol. The MME's interaction with eNodeB via S1AP protocol is independent of MME's communication with SGW through GTP protocol. Therefore, communication between MME and eNodeB, and communication between MME and SGW can be carried out in parallel. For the handover event example, the handover module consists of operations using these three protocols (i.e. S1AP, MME core, and GTP). Such network operations are mutually exclusive and can run in parallel.

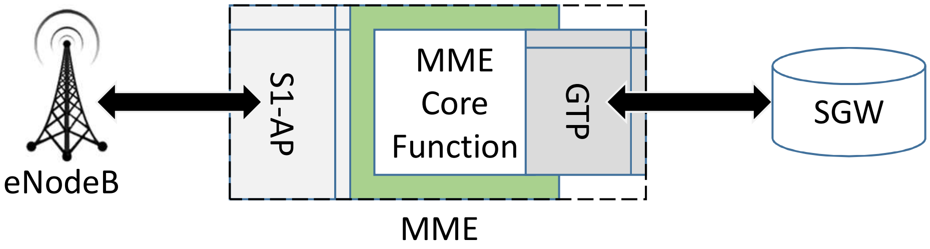


Figure 6.8: MME NF can support mutual exclusive operations

3. Exploiting data center redundancy Current data-center networks typically provide redundant servers to allow traffic flow along an alternate route when a device or link becomes unavailable [GJN]. However, these redundant servers do not take charge unless the primary server fails, and remain underutilized when the primary server is functioning. Therefore, we exploit the underutilized capacity of these redundant servers to our advantage. We use the servers to host event specific logic and facilitate event execution.

6.3.2 Design Overview

The comparison of our design with contemporary EPC architectures is shown in Figure 6.9. Network Functions (NFs) pass messages among one another and perform operations to execute an event. We see in Figure 6.9 that the legacy LTE EPC has dedicated NFs (for example, MME, SGW, and PGW), connected via fiber links, hence the message passing between NFs experience minimum delays. Therefore, these NFs almost always guarantee event execution. As we move from legacy EPC to Virtualized EPC network, we see multiple instances of the same NFs joined via unreliable IP links, distributed in data-center networks. These NFs in vEPC are designed to support few thousands users as compared to millions of users' support in the legacy network; yet they are simple, scalable, and provide plug and play solution to the service providers. However, the main culprit of EPC virtualization is the high delay across consecutive NFs for a single event. When the delay guarantees are not provided, the system fails to execute events and provide services.

First contribution We propose an alternative approach for EPC virtualization that solves the above problems. We base our design on logic-based NFs segregation instead of Instance-based NFs segregation in vEPC. For a single event, we extract the logic for that event from each NF in the form of a module. Then we assemble the event-based modules extracted from all NFs into a Fat-Proxy as shown in Figure 6.9. This Fat-Proxy acts as an NF for that event. Note that we make Fat-Proxys' only for three critical events (*handover*, *paging*, and *service request*), since we already established that these critical events constitute 50% of the control signaling traffic. These Fat-Proxys' are hosted over the data-center network's redundant servers (design insight 3). The advantage of extracting logic based modules and assembling them into a Fat-Proxy is two-fold. First, the Fat-Proxy acts as a single NF and is made to handle only a single type of event. This reduces delays and avoids timeouts. Second, huge storm of critical events' signaling traffic is diverged from the EPC to the Fat-Proxy, and the EPC can handle all the other event requests on time while ensuring the timely execution of critical events through the Fat-Proxy. In short, our approach of logic based segregation is not only distributed and scalable but also mitigates the disadvantages that ride along with distributed solutions.

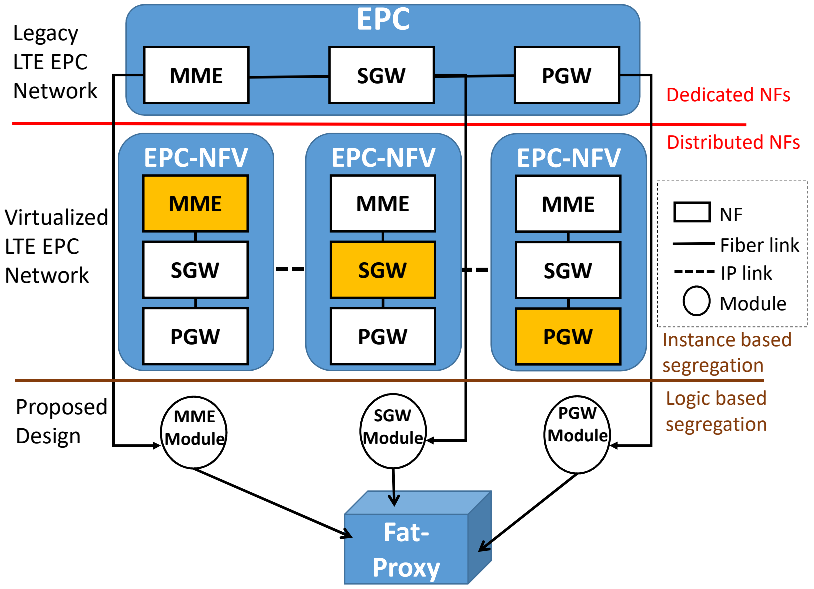


Figure 6.9: Design overview

Second contribution From the design insight 2, we came to know that there is parallelism inherently present in the logic within one module. We make use of this insight, and identify the mutual exclusive logic inside one module and partition the logic. A single partition runs one network operation. We can run multiple network operations in parallel since the logic in their respective partitions is independent of the other. This further speeds up event execution and help us choose tighter periods for the timeout timers.

6.3.3 Fat-Proxy Design

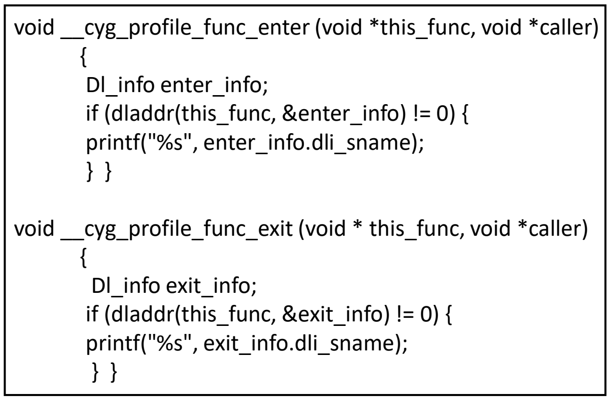
Our goal is to develop a Fat-Proxy for an event by assembling all the event specific logic components from different EPC NFs. For example, Handover Management (HoM) Fat-Proxy is made for handover event by extracting logic components from MME, SGW and PGW NFs. To develop event-specific Fat-Proxy, we perform three major steps.

1. Functional decomposition: Decomposing event-specific functions from the source code.

2. Event logic extraction: Extracting critical event functions after resolving decomposed functions dependencies.

3. Logic-based partitioning: Partitioning the mutually exclusive logic of critical event.

1) Functional decomposition The first step in our design is to decompose a NF into its constituent components.

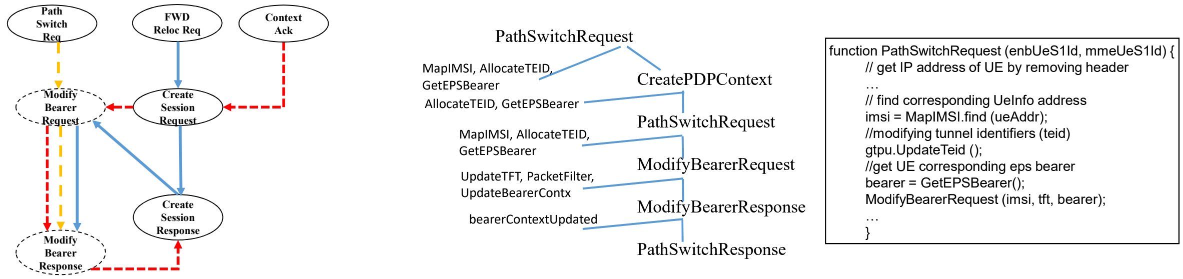


```
void __cyg_profile_func_enter (void *this_func, void *caller)
        {
         Dl_info enter_info;
         if (dladdr(this_func, &enter_info) != 0) {
         printf("%s", enter_info.dli_sname);
         } }

void __cyg_profile_func_exit (void * this_func, void *caller)
        {
          Dl_info exit_info;
         if (dladdr(this_func, &exit_info) != 0) {
         printf("%s", exit_info.dli_sname);
          } }
```

Figure 6.10: Generating functional call graph through gcc instrument functions

We use OpenEPC source code [ope] to construct function call graph. We extended etrace [etr], a run-time function call graph tool, to extract function call information and global variable usage. Our goal is to automatically identify functional dependencies over complete source code coverage. The function call graph captures the caller-callee relationship. If function A calls function B, the function call graph contains two nodes, A and B, and a directed edge from node A to node B. If a function A accesses a variable defined in function B, we also add an edge from A to B. To achieve this we use gcc feature called "instrument-functions", that adds a couple of function calls to all functions in source code. Every time a function starts, a function called *__cyg_profile_func_enter()* is called, and every time a function exits, a function called *__cyg_profile_func_exit()* is called. We simply redirect the information gathered at each function call to a trace text file. These "instrument-functions" write the addresses of functions, in which they were called, to the trace file. The data does not contain any symbol/function names. To resolve function pointer addresses to their human-readable names, we use BSD library function *dladdr()*. It takes a function pointer and tries to resolve name and file where it is located. The source code of above mentioned procedure has been shown in Figure 6.10.

Our functional decomposition methodology captures the dynamic linking of function calls (at runtime). The dynamic call graph records how decomposed functions chain when they are called in an event and under a specific scenario. This is especially important for the correctness of true functional dependencies among different events (step 2).



```
function PathSwitchRequest (enbUeS1Id, mmeUeS1Id) {
        // get IP address of UE by removing header
        …
        // find corresponding UeInfo address
        imsi = MapIMSI.find (ueAddr);
        //modifying tunnel identifiers (teid)
        gtpu.UpdateTeid ();
        //get UE corresponding eps bearer
        bearer = GetEPSBearer();
        ModifyBearerRequest (imsi, tft, bearer);
        …
        }
```

(a) Dynamic functions call graph

(b) Call graph as a tree data structure

(c) A function has nested functions

Figure 6.11: Function decomposition: (a) Different functions chain differently based on the event logic. (b) The function call graph consists of those functions (i) which are part of an event (right nodes) and (ii) which are called within a particular functions (left nodes). (c) In other words, functions are nested and only runtime call graph can identify these functions (i.e. ii)

We show this in Figure 6.11a where different functions interact differently depending upon event execution logic. Figure 6.11a captures part of *handover* and *TAU* event execution. We have intentionally omitted certain functions from this call graph to highlight the fact that same functions can be chained differently depending on the event they are executing. First, we show that there are two different ways same event (*handover* event) can execute depending upon two different scenarios. In first scenario, eNodeB$_{source}$ and eNodeB$_{target}$ are not directly connected (that is two eNodeBs are not connected over LTE *X2* interface). In this case, the downlink user packets, while the UE is in the handover process, are tunneled through EPC. The PGW forwards the packets to SGW$_{source}$ that forwards them to eNodeB$_{source}$. However, eNodeB$_{source}$ is unable to forward to eNodeB$_{target}$ because there is no *X2* interface. Then eNodeB$_{source}$ reflects back these packets to SGW$_{source}$ that uses "indirect" tunnel and forwards these packets to SGW$_{target}$. SGW$_{target}$ finally forwards them to eNodeB$_{target}$. This *ForwardRelocationRequest()* function requires that MME$_{target}$ creates a session with SGW$_{target}$ by calling *CreateSessionRequest()* and *CreateSessionResponse()* functions. Create Session procedure sets-up a new device entry (that include IMSI, APN name, Link EPS Bearer ID, PGW S5/S8 Address for Control Plane) for tunneling of downlink packets. Thereafter, device bearers are updated through modify bearer functions. In Figure 6.11a solid blue arrows show chain of call graph (FWD Reloc Req $\rightarrow$ Create Session Req $\rightarrow$ Create Session Resp $\rightarrow$ Modify Bearer Req $\rightarrow$ Modify Bearer Resp). The second scenario of handover takes place when eNodeB$_{source}$ and eNodeB$_{target}$ are connected over LTE *X2* interface. In this case, the user downlink data packets do not require to be forwarded through EPC tunnel. The eNodeB$_{target}$ sends the path switch request message (by calling *PathSwitchRequest()* function) to MME and informs that the device has moved away from eNodeB$_{source}$. The SGW needs to forward the incoming packets to a different destination. So the MME invokes *ModifyBearerRequest* procedure to the SGW and updates the downlink tunnel identifiers. This handover procedure has been shown in Figure 6.11a through yellow dashed arrows (Path Switch Req $\rightarrow$ Modify Bearer Req $\rightarrow$ Modify Bearer Resp).

We next show that how same functions interact differently depending on two different events execution logic. Similar to *handover* event, the *TAU* event also requires *Session Creation* and *Bearer Modifications* procedures are executed, as shown by TAU function call graph in Figure 6.11a (red dotted lines). In *TAU* event, when MME selects a new SGW, it sends a *Create Session Request* message per PDN connection to the selected new SGW. The PGW address and traffic flow template are indicated in the bearer *Context Request* message. The SGW informs the PGW(s) about the change by invoking *ModifyBearerRequest()* per PDN connection to the PGW(s) concerned. The PGW updates its bearer contexts and generates *ModifyBearerResponse()*. Finally, SGW generates a *Create Session Response* message to MME. Note that *Modify Bearer Req/Response* is sandwiched between *Create Session Request/Response* (which is different from *handover* event execution call graph). This has been captured in Figure 6.11a through red dotted lines (Context Ack $\rightarrow$ Create Session Req $\rightarrow$ Modify Bearer Req $\rightarrow$ Modify Bearer Resp $\rightarrow$ Create Session Resp).

Our functional call graph also considers global variable usage as a reason for functional dependency. This has been shown through an edge from *PathSwitchRequest* to *CreatePDPContext* in Figure 6.11b when former function accesses global variable (*bearer*) defined in the later function. Moreover, through source code snippet in Figure 6.11c, we show there are other functions which are although seemingly not part of event execution function, but indeed creates a dependency for that event. For example, *PathSwitchRequest()* function needs to update TEID (Tunnel Endpoint Identifier), and to get corresponding IMSI and EPS bearer before invoking *ModifyBearerRequest()* function.

2) Event logic extractions The second step in our design is to extract critical event execution logic from respective vEPC NFs and then combine them as that event's Fat-Proxy. This requires us to first identify those critical event functions on which other events rely too. Because extracting dependent functions would make original vEPC failure prone. Through our functional decomposition procedure (step 1), we construct an execution graph for each event using the function call graph. We noticed two kinds of dependencies, logic and data dependencies. The logic dependency occurs between two events when both events need that logic to execute; when not identified and handled properly, can affect the functionality of the events. Data dependency occurs during the execution of the event, when it needs to exchange user data with an external entity. Another variant of this dependency is when the start of an event depends on the end of other event. Once we have all events' execution graphs, we compare the graph of critical events with all the other events' (critical and non-critical) graphs. Our goal is to find the Common Subgraph Isomorphism (CSI) between critical events and other events' graphs; this subgraph reveals the shared components between events, which in fact are the logic dependencies between the two events. Since the CSI problem is NP-Complete, we use an improved back-tracking algorithm [KH04] from the CSI literature. Even though the backtracking algorithms are not efficient in terms of computational time, we argue that our event logic extraction is a one-time and offline procedure, hence the time complexity of CSI does not affect our approach.

For all the common components in the execution graphs of two events, we retain a copy of those common components in the NF while extracting the same components for the critical event. If there is no logic dependency for a component used by a critical event, we extract it without maintaining its copy in the NF. The identification of these logic dependencies help us maintain functional correctness for all the other events (such as *Attach*, *Detach*, *Paging* etc.). In Figure 6.12, we show execution graphs of 'Handover'event and 'Service Request'event. The highlighted components in

Figure 6.12 in step 1 (*create session request, modify bearer request, create session response, and modify bearer response*) are the common components between the two events. This implies that we cannot extract these components for handover event unless we maintain their copy in the NF to be used by service request event as well.

Data dependency arises when the handover event is in 'modify bearer' component[2]. Device bearers are to be modified at actual SGWs and PGW of EPC. This is shown in Figure 6.12 in step 2, where handover event's 'modify bearer' component needs to interact with the EPC. In our design, we restrict any communication with EPC unless the event is complete, and remove such mid-way data dependency by always generating fake 'modify bearer success' response for the device. Once the handover event finishes and returns user state to vEPC, then the vEPC runs the actual 'modify bearer' request. It is possible that such bearer modification step fails (e.g. one of SGW or PGW fails), even though the device is already notified of a successful bearer modification. This is not a problem because the vEPC can simply initiate the re-attach event in this case.

In Figure 6.12 step 3, we also see the second kind of data dependency between the handover event and the tracking area update (TAU) event. The TAU event waits for the handover event to finish and then execute. Such a dependency is mitigated when events execute in blocking mode (discussed in 6.3.1, design insight 1).

We follow the above procedure for the other two critical events, service request event and paging event. At this step, we have working Fat-Proxies for all three critical events.

3) Network protocols' logic-based partitioning The third and last step is about optimizing Fat-Proxy execution (i.e. our second contribution). We recall that delay in executing any message of a critical event will add to critical event's delay; and its any message execution failure may abort the whole critical event procedure. Therefore, there is a need to speed-up event execution by executing some messages in parallel. We propose network protocols' *logic-based* partitioning to achieve this goal. We partition the mutually exclusive logic of independent protocols in a module (module is event-based logic from one NF). We identify the opportunity of *logic-based* partitioning through analysis of these standardized protocols.

[2] Bearer modification procedure is used to modify device QoS and/or TFT (Traffic Flow Template) of an EPS bearer

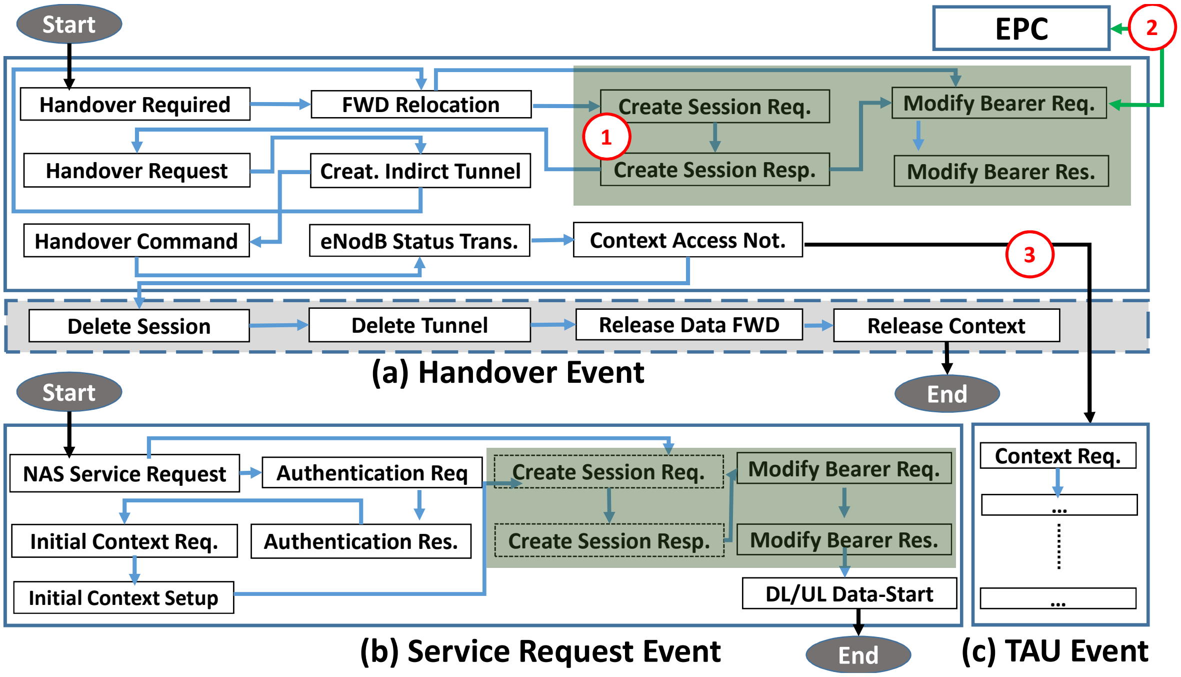


Figure 6.12: Identifying logic dependencies through Common Subgraph Isomorphism (ICS) and data dependencies through protocol analysis

We explain this through *handover* event example. The *handover* event triggers coordination between a series of NFs (MME, SGW, and PGW). Such coordination takes place between different NFs within EPC, and between EPCs and the radio network (eNodeB). The MME NF requests eNodeB to establish secure radio connection with UE, instructs eNodeB to establish device context, initiates the connection between SGW NF for user UL/DL traffic, and many more. The signaling messages exchange between MME and eNodeB are carried out by S1AP protocol, while the communication between MME and other NF (i.e. SGW) is carried out using GTP protocol. The legacy EPC infrastructure tightly couples S1AP protocol with other protocols like GTP protocol in MME. However, through design insight 3 (Figure 6.8), we know that these protocols are designed for different purposes and are independent of one another. Therefore, we partition the logic of S1AP, MME core function, and GTP protocols inside MME NF's module. Such a design choice results in faster communication between eNodeB and vEPC using S1AP protocol, and between vEPC's MME and SGW NFs using GTP protocol, even during data-center network congestion.

A protocol defines message exchange procedure between different entities. Figure 6.13 shows subset of these messages exchanged between EPC NFs during *handover* event. MME$_{serving}$ (S-MME) receives *Handover Required* message from eNodeB and triggers *Forward Allocation* message to MME$_{target}$ (T-MME). After receiving, T-MME sends "Create Session Request'" to SGW$_{target}$ (T-SGW). On successful session response from T-SGW, T-MME sends *Handover Acknowledgement* to eNodeB. Note that the direction of "Create Session Request" and "Handover Request" messages are opposite (both messages sent simultaneously), where former is a part of GTP protocol, and the later is a part of S1AP protocol (all dotted lines in Figure 6.13 show parallel execution). The simultaneous transmission of these messages is possible since their respective protocols are mutually exclusive.

Therefore, we accelerate handover event by identifying such protocol level modularity in an NF and executing their corresponding messages in parallel. We find that in most network events, there exist 40% to 60% messages that can be executed concurrently, and significantly improves the network performance.

There is a chance that concurrent message execution may fail and this failure may provide an inconsistent view of network states to eNodeB. For example, on receiving the "Handover Request" message from T-MME, the eNodeB believes that the T-MME has successfully established the device connection with T-SGW. But such eNodeB's view of network state may become false, in case T-MME failed to establish device session with T-SGW. Therefore, to handle such network state inconsistencies, we propose transaction rollback by sending a network failure message. As stated earlier, message execution failure at any step terminates whole handover process, therefore, by sending a failure message (even at later step) to eNodeB addresses any inconsistency previously caused by concurrent message execution. We should highlight that network states within different NFs of vEPC remain consistent because handover event executes in blocking mode (discussed in 6.3.1, design insight 1).

6.3.4 Robustness

We are mainly concerned that (1) our system is robust and does not introduce any side effect to original vEPC logic, and (2) the Fat-Proxy execution is correct. For (1), we do not bring

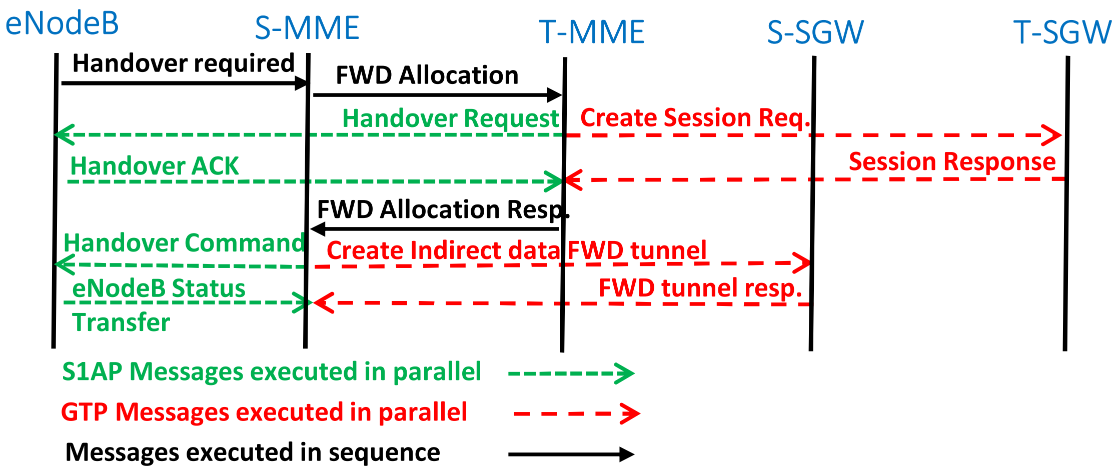


Figure 6.13: Concurrent execution of messages

any changes to source code which can be executed other than the critical event. In developing an event-specific Fat-Proxy, we extract those functional modules which are local to the critical event. Those modules which have any sort of dependency to other functions are duplicated in Fat-Proxy while retaining them in vEPC. For (2), we perform sanity checks on the operations executed by the Fat-Proxy. The rationale of performing sanity checks is to ensure the correctness of event execution. Such sanity checks are performed by finding whether Fat-Proxy returns the complete list of session states as expected. We recorded session states list against each critical event before deployment of the Fat-Proxy. These include: *imsit, stmsi, ueAddr, teid, teidCount, ulGtpTeid, dlGtpTeid enbAddr, eNBTeid, eNBS1uAddress, eNBSctpId, enbTransportLayerAddress, oldEnbUeX2apId, newEnbUeX2apId, sourceCellId, targetCellId, sn_Status_Transfer, sGWAddr, sGWTeid, m_s11SapSgw mmeUeS1Id, mmeUeS1apId, s11SapMme, mmeVid, mmePid, tft, cgi, ecgi, erabToBeReleaseIndication, erabSetupList, erabToBeSwitchedInDownlinkList, erabLevelQosParameters, erabId, epsBearerId, erabId, epsBearer, apn* . When vEPC receives a critical event, it delegates the event execution to concerned Fat-Proxy. vEPC also sends UE session and states to Fat-Proxy which are required for processing of a critical event. At this point, vEPC does not allow any other event to execute (6.3.1, design insight 1). Now, Fat-Proxy mimics a full-fledge EPC to the outside world (i.e. to serving and target eNodeBs) and executes the critical event. On completion of the event, the Fat-Proxy reports session states which it had updated while executing critical event to vEPC. vEPC performs sanity checks (by comparing the reported session variable with the expected list) and updates new session and states variables to respective vEPC NFs. In case the sanity check is failed, the vEPC asks the device to perform an *Identity* procedure that ensures that device can communicate with vEPC.

Note that it is improbable that sanity check fails because by design our Fat-Proxy is robust where it generates a failure to vEPC when it encounters internal/external failures, and performs transaction rollback procedure (discussed in subsection 6.3.3, item 3).

6.4 System Implementation

Our system implementation consists of OpenEPC LTE implementation and LTE EPC NFs virtualization.

OpenEPC LTE deployment Our test-bed consists of a LTE eNodeB (nanoLTE Access Point [nan]), OpenEPC software EPC platform [ope], and Samsung S6 smartphones. The eNodeB is a 3GPP Release 9 compliant LTE small cell on 700 MHz band. Considerable effort, involving code modifications to OpenEPC components, is made to integrate eNodeB (closed-source) with EPC to ensure interoperability with commercially available LTE clients (i.e., Samsung S6 smartphones). Our EPC network consists of MME, HSS, PCRF for control plane and SGW and PGW for data plane functions. In addition, the Internet gateway provides connectivity to the Internet. Samsung S6 smartphones use USIM cards programmed with the appropriate identification name and secret code to connect with eNodeB. Since eNodeB and the device communicate on T-Mobile's licensed band, we use custom built frequency converters. These converters convert the frequency in both downlink and uplink from 700 MHz to 2.6 GHz, where we have an experimental license to conduct over the air experiments.

For the evaluation of our second design choice (protocols' logic partitioning), where S1AP-MME simultaneously communicates with S1AP-eNodeB and SGW, we require changes at eNodeB-S1AP. Because our eNodeB is closed-source, we use device emulation provided by OpenEPC. The OpenEPC provides *client-Alice* module that emulates user device and eNodeB and interacts with EPC NFs. The *client-Alice* module has basic S1-AP functionality, enough to show performance improvement when protocols' logic partitioning is used.

Virtualizing LTE EPC After LTE testbed deployment, we virtualize EPC NFs. EPC virtualization includes the deployment of decomposed EPC NFs over VMs, and exposing them to the real LTE traffic load.

EPC's NFs decomposition and placement: We virtualize EPC NFs over vMware vSphere, which is a server virtualization platform with consistent management. We first decompose OpenEPC into a number of LTE NFs (i.e. MME, SGW, PGW, HSS, and PCRF). To implement an event-specific Fat-Proxy, we first identify the Fat-Proxy's logic based on the event's state transition diagram (as shown in Figure 6.12). We then traverse through OpenEPC NFs source code to locate that implementation logic. This event logic is deployed on a separate VM (after extracting/copying from OpenEPC NFs) and we call it that event's Fat-Proxy. Now this Fat-Proxy's VM acts as stand-alone NF. Thereafter, we configure the interfaces for all virtual NFs (LTE NFs as well as Fat-Proxy) by changing OpenEPC boot process (init) so that the OpenEPC can discover installed Fat-Proxies at start-up and allow relaying packets to and from these virtual NFs (VNFs).

During actual execution of a critical event, OpenEPC stores most of the information needed by an NF (such as device states) in a back-end database. To reduce the overhead of Fat-Proxy communication with the database back-and-forth to access these device states, we duplicate this information at Fat-Proxy when the critical event triggers.

We gather results by changing testbed configurations for two different settings:(a) placing the VNFs (with no Fat-Proxy) over different servers, which are then connected through network tunnel, and (b) installing the Fat-Proxy VNF on servers

Considering real data-center network loads: Because research lab's testbed environment does not (a) add round trip time to data-center network (b) consider dynamic loads at servers (c) take data-center network congestion into account; we consider data-center network performance metrics while compiling our results. We parsed system logs provided by HP Helion cloud infrastructure and gather inter-data-center network latency metrics. We measure round trip time from query entering and exiting the data-center network.

6.5 Evaluation and Results

We evaluate our design based on these aspects: (1) controlling signaling storm, (2) timely execution of mission critical events, and (3) performance impact. We compare our approach against contemporary instance based vEPC LTE design, which is not only deployed by a number of network operators [etsb], but also discussed in recent research papers [BH14, CX15, HG15]. We run our tests on a local network of servers with 10-core Intel Xeon E5 - 2650 v3 processors at 2.3Ghz, 25MB cache size, and 16GB memory. We build our prototype and tested it using real smartphones (Samsung S6 smartphones) and device emulation mode of OpenEPC. To consider real operator network scenario, we use a network trace as our input packet stream; results are representative of tests we run using these traces.

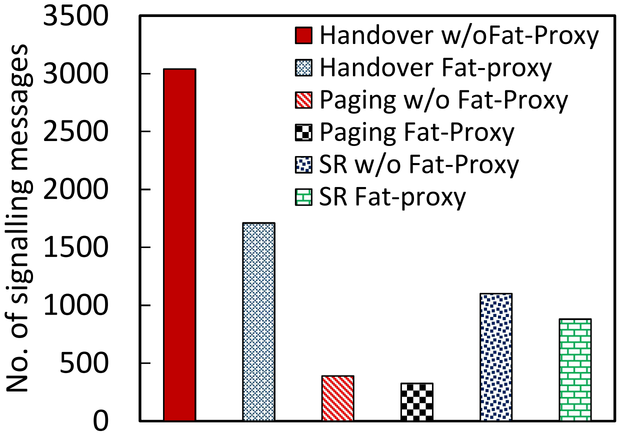
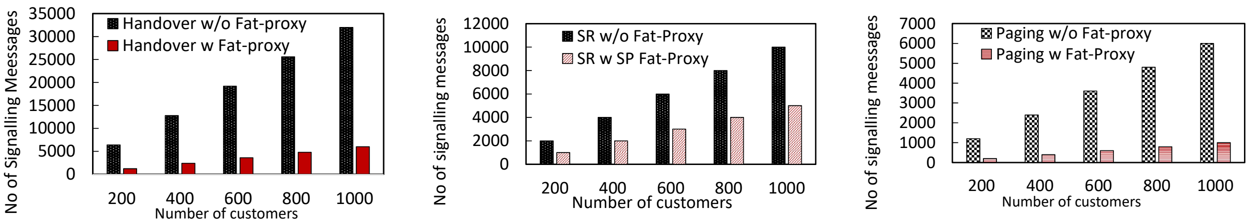


Figure 6.14: Total no of signaling messages per subscriber/hr

Signaling load at vEPC As mentioned earlier in this chapter, during busy hours, operational LTE core is exposed to signaling storm. First, we show that our design reduces signaling storm by diverting highly occurring network events to Fat-Proxy. Although Fat-Proxy is part of LTE core, all execution remains local to Fat-Proxy NF. In this way, LTE core NFs (such as MME, SGW and PGW) are not exposed to high signaling messages exchange and remain functional at all time. Figure 6.15 shows that for *handover*, *service request* and *paging* events, LTE core is exposed to 5X, 6X and 2X less signaling traffic respectively, compared to when Fat-Proxy is not used. We see that *paging* event benefits the most from our design which is due to the fact that paging Fat-Proxy

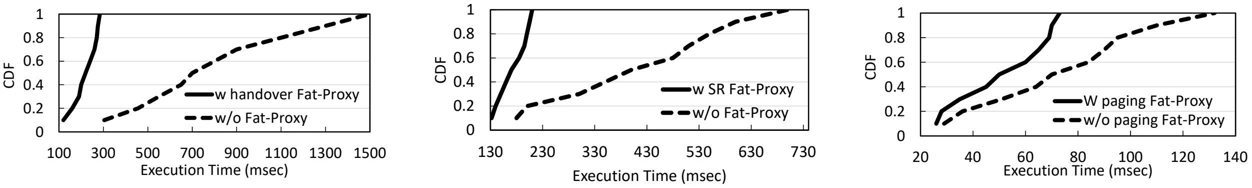


(a) Handover

(b) Service Request

(c) Paging

Figure 6.15: Signaling load vEPC is exposed during peak hours with and without Fat-Proxy



(a) Handover

(b) Service Request

(c) Paging

Figure 6.16: Event execution time during peak hours with and without Fat-Proxy concept

handles all the paging signaling with eNodeB directly after MME delegates paging execution to Fat-Proxy. Whereas, *service request* event requires bearer modifications at actual SGW and PGW which relatively increases vEPC signaling load even in case of *Service Request* Fat-Proxy.

Total number of signaling messages We show that less number of signaling messages are generated by each event with Fat-Proxy compared to the case when Fat-Proxy is not used. This is mostly because we are able to execute some messages in parallel, and also skip few messages from being executed. Figure 6.14 shows the reduction of signaling message per event (in one hour window) when Fat-Proxy is used. Note that, we are mainly interested in determing signaling load in vEPC. Therefore, we count two parallel messages as one, but in actual implementation exactly two messages are generated. The rationale of treating a pair of the parallel message as one is that these two messages are traveling in the opposite direction, i.e. one out of EPC and the other towards EPC NF. Therefore, both of these messages are independent to each other execution. Figure 6.14 shows that *handover* event produces around 40% less signaling messages, when *handover* event is handled by Fat-Proxy. This is mainly because vEPC NFs modules (specific to handover event) implemented in Fat-Proxy are local to Fat-Proxy. Therefore, these modules do not need to use signaling messages to communicate with each other and avoid unnecessary signaling exchange. The signaling messages that *handover* Fat-Proxy skips include *create session request/response*[3], *delete session request/response*[4], *UE context release command/complete*[4], *delete indirect data forwarding tunnel request/response*[4]. Figure 6.14 shows that *paging* event can only skip one message of *Uplink-Nas-Transport*. This NAS message carries the information about the service that the device wants to receive from LTE network.

Event execution time Figure 6.16 shows CDF of event execution time for *handover*, *service request* and *paging* events with and without Fat-Proxy implementation. We see that with *handover* Fat-Proxy event latency decreases by the factor of 6X on average. This improvement is observed because (1) all event execution remains local to *handover* Fat-Proxy and does not suffer any network delays, and (2) *handover* Fat-Proxy executes 6 signaling messages in parallel and skips total of 8 messages. We note that even with *handover* Fat-Proxy, handover latency is higher than 100 ms. This is because in our experiment we handle worst case handover scenario in which both MME and SGW are relocated. Although event execution time does not meet QoS time-bounds (100msec for voice over LTE call, and 300msec for TCP based traffic [3GP13a]), it does not affect user QoS experience where users' data packets are tunneled from old serving SGW to target SGW and then delivered to the user.

Service Request (SR) Fat-Proxy, on average can reduce only upto 50% *service request* event execution time mainly because at the end of *service request* event execution, SR Fat-Proxy needs to update device bearers with vEPC.

Paging event execution time with and without Fat-Proxy is not significantly high. We observe in *paging* event, all of the signaling messages are exchanged between S1AP of MME and eNodeB which diminishes intra-vEPC NFs delay. The improvement we see in Figure 6.16c is mainly achieved by pushing S1AP to the edge of cloud and executing one pair of message in parallel.

[3] Conventionally it is used to communicate with two distant NFs

[4] These messages are used to remove states from memory. In Fat-Proxy the states are automatically deleted when the Fat-Proxy responds back to vEPC

CHAPTER 7

Conclusion and Future Work

We present a summary of our work and provide insights and lessons learned from our study. We conclude the dissertation with future work.

7.1 Summary

We first summarize our main results in the dissertation.

Insecurity on LTE protocol's inter-layer interactions We show that the notion of LTE security is preserved for end-to-end user communication in which secure channel is established prior to transmitting user traffic over the shared wireless link. However, we show that such security mechanism does not guard inter-layer traffic flow, where one protocol layer can communicate with far-reaching protocol processes. The adversary exploits this unconditional trust among LTE protocol layers and is able to send the spoofed message of a special category to the network. The network receiving the special message assumes to be originated from the victim device and wrongly executes the message. This gives birth to an attack in which an adversary can kick victim out of the LTE network.

Insecurity on LTE key installation We show that the re-transmission of certain signaling messages resets the counter values multiple times that lead to reuse of keystream block for ciphering of plain text messages. In consequence, the attacker can hijack LTE location update procedure and can de-register the victim device from the network.

Testing based LTE vulnerability analysis We present the methodical approach to LTE testing. We provide a complete list of test cases by considering multiple protocols interaction and exclude those test cases whose corresponding output message combinations are not generated in protocols interaction. We identify new vulnerabilities in the LTE such as service delay, repetition of procedures, stuck to a lower priority cell, and more others.

Enabling LTE NFV over commodity data center We disclose a number of issues in virtualizing LTE core that mainly arises from *instance-based* decomposition design. We propose a new way of thinking to virtualize LTE core so that LTE events are executed within the time-bounds. We leverage the *logic-based* modularity of NFs, decompose the NFs based on events logic and assemble into a Fat-Proxy. This Fat-Proxy takes the message-intensive critical events away from the core network. We further speed up event execution by executing event messages in parallel.

7.2 Insights and Lessons Learned

We now present the insights and lessons we have learned from our work.

The network should only process those messages that pass the integrity check We have discovered that certain device messages are processed by the LTE network even when the Message Authentication Code (MAC) that fails the integrity check or cannot be verified. It is done because under certain situations these messages can be sent by the device whose security context t is no longer available in the network. Such a design decision helps network maintaining device service access and keeping the user's voice and data session. In other words, there is a tradeoff between maintaining service access and security, and we have found that the network sacrifice security over network service availability. An adversary can exploit this design loophole by spoofing victim control-plane message that is processed by the network without integrity protection. The adversary can do so because LTE protocol layers do not maintain cross-layer binding. As a result, one LTE layer blindly accepts the message coming from the layer above or below. We argue that the network should drop any device message that has failed the integrity check. It should re-authenticate the device before processing any of future device message(s).

The key installation and count reset procedures should be made atomic LTE has divided its security establishment process into two separate procedures. One procedure installs the security key while the other procedure resets the count. The design approach in which a security procedure is split into two disjoint procedures is fault tolerant where only one part of the security procedure needs to be re-executed in case of a failure. We argue that achieving fault tolerance in security compromises security. The adversary can maliciously keep resetting the count values at victim device by inducing the failure to a procedure that resets the count. In this way, the same security key (a combination of secret key and count) is used multiple times that makes the encryption vulnerable. We stipulate that security procedures must be atomic, i.e. either all security procedures successfully execute or none.

Multi protocols interaction through testing reveal new device and network interaction scenarios LTE test cases tests must be executed on every LTE capable device model before they are commercially released. These test cases ensure that the device conforms to the established procedures for its control and user plane functionalities and all practical scenarios are well tested. We find that LTE testing practices are unmethodical where the focus of testing is to follow the test cases specified in the standard as they are; resulting in incomplete testing. We discover that the standardized test cases are incomplete in which a number of test cases related to multiple protocol interactions are missing. We argue that there is a need for a complete paradigm shift from ad hoc testing to a methodical approach for telecommunication testing. A systematic and algorithmic approach to testing can only be exhaustive in which new dimensions of device and network interactions are revealed. In this way, we can find new vulnerability use cases which are not envisioned by the LTE testing standards.

LTE network functions should be loosely decomposed for timely execution of critical procedures LTE-NFV scales user-services in a low-cost fashion. It does so by transforming the centralized legacy LTE Core architecture to a distributed architecture. The distributed architecture makes multiple instances of LTE NFs and virtualizes them on commodity data-center network. We discover that the functionality of LTE-NFV architecture breaks since the distributed NF instances connected via unreliable IP links delay the execution of critical events. The failure of time-critical events results in users' quality of service degradation and temporary service unavailability. We argue that *logic-based* NFs segregation should be done for NFV, instead of *instance-based* NFs segregation done in current NFV implementation. This gives an opportunity to stitch the logic of an event into a centralized entity that becomes the stand-alone execution engine for that event. This way, only the localized entities exchange signaling messages, and the events do not experience large delays.

7.3 Future Work

There are three future research topics that need to be explored in the context of this dissertation.

A vulnerability analysis tool We aim to test every test case based on all testing input combinations: the test purpose, conditions, inputs, requirements, procedures, and execution output. We are interested in completely automating this hefty process and identifying the abnormal test cases. We will only look at those input and output combinations that question the secure operations of LTE network. To do so, we will implant three security checkpoints (i.e. authentication, authorization, and access control) in the test case execution path. We mark the operation vulnerable if 1) test case skips any of three security checkpoints, or 2) the test case is a success, even though one of the security operations fails.

In the second phase of this work, we aim to perform such vulnerability analysis in real-time. It will be more like an iOS or Android app that checks the control-plane device input/output values against the permissible set of values. The app will raise an alarm if the device deviates from the established conditions.

Edge Microservices for low latency data access in 5G Motivated from Software Defined Networking (SDN), LTE standard body (3GPP) has recently proposed splitting monolithic LTE NFs into their control-plane and data-plane modules. The data-plane logic is pushed at the edge of the network while retaining control-plane functionality at the core. Both network edge and the core modules involve in executing LTE control-plane procedures (e.g., device registration/derigstration, and mobility, etc.) as well as LTE data-plane services (e.g., voice over LTE, and video streaming, etc.). As a future work, we will study whether SDN style approach truly works for LTE NFs which are monolithic by design or not. In this research, we will focus on studying the interactions between LTE edge and the core and what types of research challenges such interactions pose. After resolving the challenges on LTE control and data plane split, we will propose application specific microservice. The idea behind such microservice is to tailor the data plane forwarding according to the specific requirements of the application using it. Our approach has the potential to reduce the data latencies for the application supported over a microservice.

We can extend our work in designing effective 5G Internet of Things (IoT) based services. We consider those IoT applications, such as highly sensor-intensive and thus data-intensive applications, that generate a lot of data. We can think these IoT devices as edge devices where the data is generated at the edge and design an edge based solution that meet application-specific latency and performance requirements.

Reliability support for virtualized IP Multimedia Subsystem As future work, we will make the virtualized IP Multimedia Subsystem (IMS) highly reliable. We divide this future work into two phases. In the first phase, we will perform an empirical assessment to find how well current IMS and cloud-based mechanisms perform during failures. We will study whether the existing state of the art approaches maintain service and network connectivity during failure or not; and whether the failure recovery procedures are fine-grained to log session level updates for every device? In the second phase of the work, we will take lessons learned from the empirical assessment in designing highly reliable virtualized IMS. In our design, we will focus on both failure detection and efficient failure recovery mechanisms. We aim to exploit both LTE and IMS specific information to achieve high reliability in the virtualized environment.

REFERENCES

[3GP] "3GPP implementation conformance statement." http://www.etsi.org/deliver/etsi_ts/136500_136599/13652302/11.03.00_60/ts_13652302v110300p.pdf.

[3GP12a] 3GPP. "TS24.008: Core Network Protocols.", 2012.

[3GP12b] 3GPP. "TS36.331: Radio Resource Control (RRC).", 2012.

[3GP13a] 3GPP. "TS 23.203: Policy and Charging Control Architecture.", 2013.

[3GP13b] 3GPP. "TS24.301: Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS); Stage 3.", Jun. 2013.

[3GP13c] 3GPP. "TS29.281: General Packet Radio System (GPRS) Tunnelling Protocol User Plane (GTPv1-U).", 2013.

[3GP13d] 3GPP. "TS33.401: 3GPP SAE; Security architecture.", Sep. 2013.

[3GP13e] 3GPP. "TS36.304: User Equipment procedures in idle mode.", 2013.

[3GP14] 3GPP. "TS29.274: Tunnelling Protocol for Control plane (GTPv2-C).", 2014.

[AGM11] M. Alizadeh, A. Greenberg, D.A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan. "Data center tcp (dctcp)." *ACM SIGCOMM computer communication review*, 2011.

[Ahm13] Sassan Ahmadi. *LTE-Advanced: A Practical Systems Approach to Understanding 3GPP LTE Releases 10 and 11 Radio Access Technologies*. Academic Press, Waltham, Massachusetts, USA, 1 edition, 2013.

[AMR07] Marco Almeida, Nelma Moreira, and Rogério Reis. "On the performance of automata minimization algorithms." *Logic and Theory of Algorithms*, p. 3, 2007.

[ANIa] "Anite conformance test systems." http://www.anite.com/businesses/handset-testing/our-products.

[ANIb] "Anite maintains LTE conformance testing lead." http://www.anite.com/businesses/handset-testing/news/anite-maintains-lte-conformance-testing-lead.

[ANIc] "UE demonstration of conformance testing – Anite." http://www.anite.com/businesses/handset-testing/our-products.

[ANRa] "Anritsu conformance test systems." `https://www.anritsu.com/en-US/test-measurement/mobile-wireless-communications/conformance-test-systems`.

[ANRb] "Anritsu: How conformance tests are carried out." `http://dl.cdn-anritsu.com/en-en/test-measurement/files/Product-Introductions/Product-Introduction/me7873la-el1100.pdf`.

[AT] "AT Commands List." http://www.lte.com.tr/uploads/pdfe/1.pdf.

[ATT] "Network Functions Virtualization: Challenges and Opportunities for Innovations." `http://web2-clone.research.att.com/export/sites/att_labs/techdocs/TD_101400.pdf`.

[Aum11] Aumasson, Jean-Philippe et al. "A note on a privacy-preserving distance-bounding protocol." In *International Conference on Information and Communications Security*, pp. 78–92. Springer, 2011.

[AYK12] Mohammad Alizadeh, Shuang Yang, Sachin Katti, Nick McKeown, Balaji Prabhakar, and Scott Shenker. "Deconstructing datacenter packet transport." In *ACM Workshop on hot topics in networks*, 2012.

[B16] Böck, Hanno, Aaron Zauner, Sean Devlin, Juraj Somorovsky, and Philipp Jovanovic. "Nonce-Disrespecting Adversaries: Practical Forgery Attacks on GCM in TLS." *IACR Cryptology ePrint Archive*, **2016**:475, 2016.

[Ba15] Arijit Banerjee and et al. "Scaling the LTE control-plane for future mobile access." In *ACM CoNEXT*, 2015.

[BAMa] T. Benson, A. Akella, and D. Maltz. "Network traffic characteristics of data centers in the wilds." In *ACM IMC*.

[BAMb] T. Benson, A. Akella, and D. Maltz. "Network traffic characteristics of data centers in the wilds." In *ACM IMC*.

[Beu] Beurdouche, Benjamin and et al. "A messy state of the union: Taming the composite state machines of TLS." In *IEEE Security and Privacy, year=2015*.

[BH14] Wolfgang Kellerer-Marco Hoffmann Hans Jochen Morper Basta, Arsany and Klaus Hoffmann. "Applying NFV and SDN to LTE mobile core gateways, the functions placement problem." *In Proceedings of the All things cellular: operations, applications, and challenges*, 2014.

[BKO10] David Barrera, H Güneş Kayacik, Paul C etcvan Oorschot, and Anil Somayaji. "A Methodology for Empirical Analysis of Permission-Based Security Models and its Application to Android." In *ACM CCS*, 2010.

[Bor01] Borisov, Nikita and Goldberg, Ian and Wagner, David. “Intercepting mobile communications: the insecurity of 802.11.” In *ACM Mobicom*, 2001.

[BU14] Ravishankar Borgaonkar and Swapnil Udar. “Understanding IMSI privacy.” In *Vortrag auf der Konferenz Black Hat*, 2014.

[CFG11] Erika Chin, Adrienne Porter Felt, Kate Greenwood, and David Wagner. “Analyzing Inter-Application Communication in Android.” In *ACM Mobisys*, 2011.

[CKP02] Jean-Marc Champarnaud, Ahmed Khorsi, and Thomas Paranthoën. “Split and join for minimizing: Brzozowski’s algorithm.” *Stringology*, **2002**:96–104, 2002.

[CL10] Ching-Hsiang Chuang and Phone Lin. “Performance study for HARQ–ARQ interaction of LTE.” *Wireless Communications and Mobile Computing*, **10**(11):1459–1469, 2010.

[CMZ] Matt Calder, Rui Miao, Kyriakos Zarifis, Ethan Katz-Bassett, Minlan Yu, and Jitendra Padhye. “Don’t drop, detour!” In *ACM SIGCOMM Computer Communication Review*.

[CX15] Vijay Gopalakrishnan-Bo Han Murad Kablan Oliver Spatscheck Chengwei Wang Chiu, Angela and Yang Xu. “EdgePlex: decomposing the provider edge for flexibilty and reliability.” *In Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*, 2015.

[Dan87] Dana Angluin. “Learning Regular Sets from Queries and Counterexamples.” In *IEEE Information and Computation*, 1987.

[De 15] De Ruiter, Joeri and Poll, Erik. “Protocol State Fuzzing of TLS Implementations.” In *USENIX Security Symposium*, 2015.

[DIS] “Why Distribution Matters in NFV.”
`http://www.tmcnet.com/tmc/whitepapers/documents/whitepapers/2014/10462-why-distribution-matters-nfv.pdf`.

[eria] “Ericsson SGSN-MME.”
`http://www.ericsson.com/ourportfolio/products/sgsn-mme?nav=productcategory004%7Cfgb_101_256/`.

[ERIb] “Network Functions Virtualization and Software Management.”
`http://www.ericsson.com/res/docs/whitepapers/network-functions-virtualization-and-software-management.pdf`.

[etr] “Run-time function call tree with gcc.”
`http://ndevilla.free.fr/etrace/`.

[ETSa] “Carrier NFV Project - One Year Milestone.”
`http://www.layer123.com/download&doc=ETSI-1013-Lopez-NFV_One_Year_Milestone`.

[etsb] “Carrier NFV Project - One Year Milestone.” http://www.layer123.com/download&doc=ETSI-1013-Lopez-NFV_One_Year_Milestone.

[Far] Nathan Farrington. “Multipath TCP under Massive Packet Reordering.” In *UC San Diego, Technical Report*.

[Fay16] Fayaz, Seyed Kaveh and Yu, Tianlong and Tobioka, Yoshiaki and Chaki, Sagar and Sekar, Vyas. “BUZZ: Testing Context-Dependent Policies in Stateful Networks.” In *NSDI*, 2016.

[Fit16] Fiterău-Broştean, Paul and Janssen, Ramon and Vaandrager, Frits. “Combining model learning and model checking to analyze TCP implementations.” In *International Conference on Computer Aided Verification*. Springer, 2016.

[GH07] Hermann Gruber and Markus Holzer. “Computational Complexity of NFA Minimization for Finite and Unary Languages.” *LATA*, **8**:261–272, 2007.

[GJN] Phillipa Gill, Navendu Jain, and Nachiappan Nagappan. “Understanding network failures in data centers: measurement, analysis, and implications.” In *ACM SIGCOMM Computer Communication Review*.

[GN16] Philip Ginzboorg and Valtteri Niemi. “Privacy of the long-term identities in cellular networks.” In *Proceedings of the 9th EAI International Conference on Mobile Multimedia Communications*, pp. 167–175. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2016.

[God97] Godefroid, and Patrice. “Model checking for programming languages using VeriSoft.” In *Proceedings of the 24th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, 1997.

[GPG12] Mikhail Gerasimenko, Vitaly Petrov, Olga Galinina, Sergey Andreev, and Yevgeni Koucheryavy. “Energy and delay analysis of LTE-advanced RACH performance under MTC overload.” In *Globecom Workshops (GC Wkshps), 2012 IEEE*, pp. 1632–1637. IEEE, 2012.

[GRA13] Pedro García Gómez, Damián López Rodríguez, and Manuel Vázquez-de-Parga Andrade. “DFA minimization: from Brzozowski to Hopcroft.” 2013.

[HG15] Wolfgang Hahn and Borislava Gajic. “GW elasticity in data centers: Options to adapt to changing traffic profiles in control and user plane.” *In Intelligence in Next Generation Networks (ICIN)*, 2015.

[Hon18] Hong, Byeongdo and et al. “GUTI Reallocation Demystified: Cellular Location Tracking with Changing Temporary Identifier.” 2018.

[hop] “DFA minimization using Hopcroft alogirithm: C++ implementation.” http://www.cs.tut.fi/~ava/DFA_minimizer.cc.

[Hop71] John Hopcroft. "AN/N LOG N ALGORITHM FOR MINIMIZING STATES IN kF IN ITE AUTOMATON." 1971.

[HQG13] Junxian Huang, Feng Qian, Yihua Guo, Yuanyuan Zhou, Qiang Xu, Z Morley Mao, Subhabrata Sen, and Oliver Spatscheck. "An in-depth study of LTE: effect of network protocol and application behavior on performance." In *ACM SIGCOMM*, 2013.

[Hus18] Hussain, Syed Rafiul and et al. "LTEInspector: A Systematic Approach for Adversarial Testing of 4G LTE." In *NDSS*, 2018.

[Kah96] Kahn, David. "The codebreakers." *New York, NY: Scribner*, 1996.

[KH04] Evgeny B Krissinel and Kim Henrick. "Common subgraph isomorphism detection by backtracking search." *Software: Practice and Experience*, **34**(6):591–607, 2004.

[Khu03] Khurshid, Sarfraz and Păsăreanu, Corina S and Visser, Willem. "Generalized symbolic execution for model checking and testing." In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2003.

[KKK15] Hongil Kim, Dongkwan Kim, Minhee Kwon, Hyungseok Han, Yeongjin Jang, Dongsu Han, Taesoo Kim, and Yongdae Kim. "Breaking and fixing volte: Exploiting hidden data channels and mis-implementations." In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 328–339. ACM, 2015.

[Lee97] Lee, David and Netravali, Arun N and Sabnani, Krishan K and Sugla, Binay and John, Ajita. "Passive testing and applications to network management." In *International Conference on Network Protocols*. IEEE, 1997.

[Lic14] Lichtman, Marc and et al. "Detection and mitigation of uplink control channel jamming in LTE." In *IEEE Milcom*, 2014.

[Lic16] Lichtman, Marc and et al. "LTE/LTE-a jamming, spoofing, and sniffing: threat assessment and mitigation." *IEEE Communications Magazine*, **54**(4):54–61, 2016.

[lte] "LTE protocol layer stack."
`http://www.tutorialspoint.com/lte/lte_protocol_stack_layers.htm/`.

[LTP15] Chi-Yu Li, Guan-Hua Tu, Chunyi Peng, Zengwen Yuan, Yuanjie Li, Songwu Lu, and Xinbing Wang. "Insecurity of voice solution volte in lte mobile networks." In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 316–327. ACM, 2015.

[Luc14] Alcatel Lucent. "LTE Subscriber Service Restoration." In *ALU Technical Report*, 2014.

[Man73] Robert Mandl. "Precise bounds associated with the subset construction on various classes of nondeterministic finite automata." In *Princeton Conference on Information and System Sciences*, pp. 263–267, 1973.

[mic] “Of Mice and Elephants.”
http://networkheresy.com/2013/11/01/of-mice-and-elephants/.

[MM08] MM, Galib Asadullah. *Robust wireless communications under co-channel interference and jamming*. PhD thesis, Georgia Institute of Technology, PhD thesis, 2008.

[mmea] “MME Pool Overlap.”
http://lteuniversity.com/get_trained/expert_opinion1/b/johnmckeague/archive/2012/03/06/mme-pool-overlap.aspx.

[mmeb] “MME Trigger OVERLOAD START Towards ENodeB.”
http://tech.queryhome.com/103073/different-situations-trigger-overload-start-towards-enodeb.

[mob] “Mobile Insight.”
http://mobileinsight.net/.

[Moo71] Frank R Moore. “On the bounds for state-set size in the proofs of equivalence between deterministic, nondeterministic, and two-way finite automata.” *IEEE Transactions on computers*, **100**(10):1211–1214, 1971.

[MRD15] Mittal, Radhika, Nandita Dukkipati, Emily Blem, Hassan Wassel, Monia Ghobadi, Amin Vahdat, Yaogong Wang, David Wetherall, and David Zats. “TIMELY: RTT-based congestion control for the datacenter.” In *ACM SIGCOMM*, August 2015.

[MRF12] Claudio Marforio, Hubert Ritzdorf, Aurélien Francillon, and Srdjan Capkun. “Analysis of the Communication between Colluding Applications on Modern Smartphones.” In *ACM ACSAC*, 2012.

[nan] “nanoLTE Access Points.” http://www.ipaccess.com/en/lte/.

[Nas14] Naseef, M. “Vulnerabilities of LTE and LTE-Advanced Communication White Paper.” 2014.

[Net14] Juniper Networks. “GPRS Tunneling Protocol (GTP) for Serving Gateway.” In *Juiper, Technical Report*, 2014.

[nfv] “Vendor documents: NFV World Congress.”
https://www.layer123.com/current-events/.

[NSN12] NSN. “Signaling is growing 50% faster than data traffic.”, 2012.

[ope] “Open EPC - open source LTE implementation.”
http://www.openepc.net/.

[OPN] “Open Platform for NFV (OPNFV).”
https://www.opnfv.org/.

[Pet17] Petrenko, and Alexandre. "Toward testing from finite state machines with symbolic inputs and outputs." *Software & Systems Modeling*, 2017.

[PLT] Chunyi Peng, Chi-yu Li, Guan-Hua Tu, Songwu Lu, and Lixia Zhang. "Mobile data charging: new attacks and countermeasures." In *Proceedings of the 2012 ACM conference on Computer and communications security*.

[PLW14] Chunyi Peng, Chi-Yu Li, Hongyi Wang, Guan-Hua Tu, and Songwu Lu. "Real threats to your data bills: Security loopholes and defenses in mobile data charging." In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp. 727–738. ACM, 2014.

[PNN12] Rahul Potharaju, Andrew Newell, Cristina Nita-Rotaru, and Xiangyu Zhang. "Plagiarizing Smartphone Applications: Attack Strategies and Defense Techniques." In *ACM ESSoS*. 2012.

[QM12] Zhiyun Qian and Zhuoqing Mao. "Off-Path TCP Sequence Number Inference Attack-How Firewall Middleboxes Reduce Security." In *IEEE Security & Privacy*, 2012.

[qps] "QPST— Qualcomm service programmer tool." `https://github.com/botletics/SIM7000-LTE-Shield/tree/master/SIM7000\%20Documentation/Firmware\%20Updater\%20Tool/QPST\%20Tool`.

[Qua] Qualcomm. "QxDM Professional - QUALCOMM eXtensible Diagnostic Monitor." http://www.qualcomm.com/media/documents/tags/qxdm.

[RS59] Michael O Rabin and Dana Scott. "Finite automata and their decision problems." *IBM journal of research and development*, **3**(2):114–125, 1959.

[Rup18] Rupprecht, David and et al. "Breaking LTE on Layer Two." In *IEEE S&P*, 2018.

[ser] "QPST Service Programming." http://forum.xda-developers.com/showthread.php?t=1180211.

[Sha48] Shannon, CE. "(1948), "A Mathematical Theory of Communication", Bell System Technical Journal, vol. 27, pp. 379-423 & 623-656, July & October." 1948.

[Shi16] Shi, Jinghao and Lahiri, Shuvendu K and Chandra, Ranveer and Challen, Geoffrey. "Wireless protocol validation under uncertainty." In *International Conference on Runtime Verification*. Springer, 2016.

[Sip98] Michael Sipser. "Chapter 1: Regular Languages." *Introduction to the Theory of Computation*, pp. 31–90, 1998.

[Sip06] Michael Sipser. *Theorem 1.19 in Introduction to the Theory of Computation*, volume 2. Thomson Course Technology Boston, 2006.

[SP15] Wouter Tavernier-Didier Colle Sahhaf, Sahel and Mario Pickavet. "Network service chaining with efficient network function mapping based on service decompositions." *In Network Softwarization (NetSoft), 2015 1st IEEE Conference*, 2015.

[SS16] Ravishankar Borgaonkar-N. Asokan Valtteri Niemi Shaik, Altaf and Jean-Pierre Seifert. "Practical Attacks Against Privacy and Availability in 4G/LTE Mobile Communication Systems." In *NDSS*, 2016.

[SSK16] Ramprasad Subramanian, Kumbesan Sandrasegaran, and Xiaoying Kong. "Benchmarking of real-time LTE network in dynamic environment." In *Communications (APCC), 2016 22nd Asia-Pacific Conference on*, pp. 20–25. IEEE, 2016.

[Ste11] Matthew Baker Stefania Sesia, Issam Toufik. *LTE - The UMTS Long Term Evolution: From Theory to Practice*. John Wiley & Sons, Hoboken, NJ, USA, 2 edition, 2011.

[Str07] Daehyun Strobel. "IMSI catcher." *Chair for Communication Security, Ruhr-Universität Bochum*, **14**, 2007.

[Stu04] Stubblefield, Adam and et al. "A key recovery attack on the 802.11 b wired equivalent privacy protocol (WEP)." *ACM transactions on information and system security (TISSEC)*, **7**(2):319–332, 2004.

[SY96] Kai Salomaa and Sheng Yu. "NFA to DFA transformation for finite languages." In *International Workshop on Implementing Automata*, pp. 149–158. Springer, 1996.

[SZZ11] Roman Schlegel, Kehuan Zhang, Xiaoyong Zhou, Mehool Intwala, Apu Kapadia, and X Wang. "Soundcomber: A Stealthy and Context-Aware Sound Trojan for Smartphones." In *NDSS*, 2011.

[ter] "Tera-Term-A Terminal Emulator." http://ttssh2.sourceforge.jp/index.html.en.

[TES] "Anite test documents."
`http://www.keysight.com/en/pd-2372474-pn-E7515A/uxm-wireless-test-set?pm=PL&nid=-33762.1078013&cc=US&lc=eng`.

[thi] "ThinkRF: Real-Time Spectrum Analyzer."
`https://www.thinkrf.com/real-time-spectrum-analyzers/`.

[TLP] Guan-Hua Tu, Yuanjie Li, Chunyi Peng, Chi-Yu Li, Hongyi Wang, and Songwu Lu. "Control-Plane Protocol Interactions in Cellular Networks." In *ACM SIGCOMM*.

[TOT] "3GPP Specification series."
`http://www.3gpp.org/dynareport/36-series.htm/`.

[TS78] T.S. Chow. "Testing Software Design Modeled by Finite-State Machines." *IEEE Transactions on Software Engineering*, 1978.

[TS314] "3GPP TS 36.523-1: Protocol conformance specification.", 2014.

[Tu13] Tu, Guan-Hua and et al. “How voice calls affect data in operational LTE networks.” In *ACM CCS*, 2013.

[Tu16] Tu, Guan-Hua and et al. “Detecting problematic control-plane protocol interactions in mobile networks.” *IEEE/ACM Transactions on Networking*, **24**(2):1209–1222, 2016.

[Van17] Vanhoef, Mathy and Piessens, Frank. “Key reinstallation attacks: Forcing nonce reuse in WPA2.” In *ACM CCS*, 2017.

[Vau02] Vaudenay, Serge. “Security Flaws Induced by CBC Padding—Applications to SSL, IPSEC, WTLS...” In *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 534–545. Springer, 2002.

[wav] “IntelliJudge: WaveJudge LTE packets sniffer.” `http://www.sanjole.com/our-products/intellijudge-lte/`.

[WIN] “Wireless Network Virtualization: Ensuring Carrier Grade Availability.” `https://www.sdxcentral.com/wp-content/uploads/2014/10/Wind-River_CRAN-white-paper.pdf`.

[XWS14] Yu Xia, Ting Wang, Zhiyang Su, and Mohamed Hamdi. “Preventing passive TCP timeouts in data center networks with packet drop notification.” In *CloudNet*, October 2014.

[Yan14] Yan, Qiben and et al. “MIMO-based jamming resilient communication in wireless networks.” In *IEEE Infocom* , 2014.

[YP11] Yi-Hua E Yang and Viktor K Prasanna. “Space-time tradeoff in regular expression matching with semi-deterministic finite automata.” In *INFOCOM, 2011 Proceedings IEEE*, pp. 1853–1861. IEEE, 2011.

[Zen09] Zenner, Erik. “Nonce generators and the nonce reset problem.” In *International Conference on Information Security*, pp. 411–426. Springer, 2009.

[Zen17] Zeng, Huacheng and et al. “Enabling jamming-resistant communications in wireless MIMO networks.” In *IEEE CNS*, 2017.

[ZIK13] D. Zats, A. P. Iyer, R. H. Katz, I. Stoica, and A. Vahdat. “FastLane: An agile congestion signaling mechanism for improving datacenter performance.” In *UC Berkeley, Technical Report*, 2013.

[ZJ16] Hongli Zhao and Hailin Jiang. “LTE-M system performance of integrated services based on field test results.” In *Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), 2016 IEEE*. IEEE, 2016.