UNIVERSITY OF CALIFORNIA

Los Angeles

System Security in 5G/4G/xG Mobile Networks: New Attacks and Countermeasures

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Zhaowei Tan

2022

ABSTRACT OF THE DISSERTATION

System Security in 5G/4G/xG Mobile Networks: New Attacks and Countermeasures

by

Zhaowei Tan
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2022
Professor Songwu Lu, Chair

5G/4G offers anytime, anywhere data services to billions of users every month. It is thus of great importance to proper protection of the 5G/4G systems from adversaries. The current systems deploy a few security mechanisms. Most notably, the user device and network perform a one-time mutual authentication using a secured private key. After the procedure, all subsequent data packets and control-plane signaling messages are encrypted and integrity protected. In addition, the delivery of each data is strictly controlled by the network. That said, the conventional wisdom believes these deployed mechanisms sufficiently ensure 5G/4G data-plane security. Most published works report attacks before mutual authentication or manipulate the procedure with techniques such as man-in-the-middle.

Contrary to common belief, this dissertation finds the current 5G/4G systems to be still vulnerable to new attacks even after the mutual authentication. The data-plane signaling messages, which facilitate data transfer (for scheduling, power management, etc.), are always sent in cleartext over the air. Meanwhile, message authenticity such as integrity protection is not always enabled.

This dissertation makes four intellectual contributions. First, we design novel data-plane forgery attacks, which can be launched assuming no key compromise, malware, or fake base station. The forged data can pass all security fences at receiver and incur serious damages from privacy intrusion to DoS variants. The attacks include `CDS` which targets critical data-plane signaling messages and `FANE` which could forge data-plane packets. Second, we propose a novel, marginal-overhead protection scheme that fundamentally secures 5G/4G data plane without protecting data itself. The goal is to secure the necessary step in data delivery to prevent any forgery attacks. Third, we devise a complementary device-centric detection and mitigation method. It can be achieved on the device side in user space with no infrastructure support, firmware/hardware access, or root privilege. This way, it provides a prompt remedy for legacy devices. Finally, we study a reversed problem of improving the performance without affecting security. We propose an application-layer, rootless solution that facilitates latency-sensitive applications such as VR/AR.

This dissertation shows how a forgery attacker addresses two critical challenges. Firstly, the attacker needs to break 5G/4G access control and make the forgery appear legitimate. To achieve so, the attack leverages the cross-layer vulnerabilities in link layers and infers the authenticated time slots and frequency. The attacker then corrupts the legitimate transmission and sends the forged data-plane signaling as retransmission. Second, the attacker needs to select the timing and content of the forge messages. We propose a smart attacker that eavesdrops on the channel and adapts the forgery content based on the observed context. With the attack methodology, we design `CDS` that leverage data-plane signaling, which includes 3 attacks that cause dysfunction in a single protocol and 4 more that cause cascading damages in multiple layers. They can inflict a range of serious damages from breach of privacy (e.g., inferring the victim's location) to beyond-simplistic DoS (e.g., forcing the victim to repeatedly send the same data). We also introduce `FANE` which forges encrypted data packets by leveraging retransmission. It can redirect a user's HTTP connection to a malicious server or infer the victim's encrypted IP address.

To combat data-plane forgery attacks including `CDS` and `FANE`, the conventional approach takes the per-packet protection philosophy. However, protecting all data (data-plane signaling + user packets) in 5G/4G incurs unacceptable overhead considering the high data rate. Instead, we note that a successful delivery requires using the correct meta-info, such as physical layer ID or modulation parameters. If an attacker is denied from getting meta-info, it cannot launch a successful forgery. This dissertation proposes an alternative solution to protecting meta-info with low overhead. We study which meta-info is suitable for protection among all possible choices. The ideal meta-info should be both necessary for every message and practical to protect. We propose a solution that targets a data structure called DCI. It is magnitude less frequent compared to data packets, while containing critical meta-info for both MAC and PHY. Securing DCI can thus effectively protect 5G/4G systems. We overcome the challenges of lack of variant parameters for DCI by using a novel, time-based scheme to generate keystream. The design leverages the property in 5G/4G scheduling and adopts an inference algorithm to pre-generate the keystream for low solution overhead.

This dissertation further designs a device-side, user-space detection and reaction scheme. The core idea is to verify what's right according to the protocol operations, rather than defend against certain threats. It thus differentiates itself from any defense scheme that can only protect against certain attacks. To operate on the user space, we devise a novel algorithm to infer, decode, and analyze critical 5G/4G signaling messages and configurations to verify if the messages are received in the right context. When a potential attack is spotted, our solution triggers proper countermeasure to void the attacker with techniques such as band switching. Neither detection nor mitigation requires extra privilege (e.g., root access), thus not exposing any new vulnerabilities. This scheme complements the fundamental meta-info protection and benefits the current device and next-generation applications before the new security mechanism is fully rolled out.

To evaluate the attacks and solutions, we build an open-source, standard-compliant software-defined radio 5G/4G system named Flora. Its salient features include in-network analytics, run-time intelligence, and new feature support (carrier aggregation, handover, etc.). Flora does not depend on any specialized hardware (SIM, chipset, or infrastructure) and works with commercial-off-the-shelf devices. We implement and integrate our security defenses in our system testbed. To assess the security, we further develop next-generational mobile VR/AR applications that run on Flora platform. The evaluation has validated the attacks and confirmed the effectiveness of our approach to protection, detection, and mitigation. We also build Sonica, the first software-defined NB-IoT platform that implements 5G/4G CIoT optimizations. We confirm that our attacks and countermeasures in broadband apply to IoT scenarios.

Moreover, we show that improving 5G/4G network performance can be achieved in the application layer without extra privilege. Many performance-boosting algorithms for 5G/4G end-users need additional privilege or even firmware change, as OS and network only expose constrained information without root. However, enabling root privilege is usually unavailable and opens new vulnerabilities for attackers to install malware or steal information. We design a rootless solution `LRP` that can infer critical parameters in 5G/4G networks with deep domain knowledge and learning algorithms on the application layer. With the inferred parameters, the application layer can act accordingly to boost the performance. We show that, an application can ask for 5G/4G radio resources early and bypass unnecessary waiting to reduce delay. `LRP` thus helps latency-sensitive applications, such as VR/AR, to meet stringent requirements. The effectiveness of `LRP` is verified with devices under commercial operational networks.

The results of this dissertation unveil some deeply-rooted design choices in 5G/4G systems that sacrifice their security. One example is the improper trust model, where any data-plane signaling message is received without verifying the data content. The cross-layer vulnerabilities make our attacks (forging data) possible, while our protections (verify meta-info for every delivery) and detection (content verification) shed light on proper security measures in future 5G releases and xG security.

The dissertation of Zhaowei Tan is approved.

Christina Panagio Fragouli

George Varghese

Lixia Zhang

Songwu Lu, Committee Chair

University of California, Los Angeles

2022

To my dear family.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere gratitude to my advisor, Prof. Songwu Lu. His vision, wisdom, enthusiasm, and high standard for research have tremendous influence on me. I greatly appreciate his support, guidance, and suggestions throughout my research career. Besides, he cares for his students and helps us get through hard times and make critical decisions in both career and life.

I also want to thank my other dissertation committee and qualifying examination committee members: Prof. Lixia Zhang, Prof. George Varghese, and Prof. Christina Fragouli. They provide me with constructive comments on my dissertation and research work. This dissertation will not be in its current shape without their help.

I am grateful to Prof. Yuanjie Li from Tsinghua University for his continuous advice on my research projects and Prof. Chunyi Peng from Purdue University & Prof. Guan-Hua Tu from Michigan State University for their generous advice on my research career. I also want to thank Prof. Xinbing Wang and Prof. Xiaofeng Gao from Shanghai Jiao Tong University who advised my undergraduate thesis.

My research projects cannot go this far without the support and hard work from my collaborators. It is my great honor to be a member of the Wireless Networking Group (WiNG), where I am privileged to work with colleagues: Yuanjie Li, Muhammad Taqi Mehdi, Zengwen Yuan, Qianru Li, Zhehui Zhang, Yunqi Guo, Zhehan Li, Jinghao Zhao, Boyan Ding, Kaiyuan Chen, and Yifei Xu. I also appreciate the help from great researchers and friends outside our lab.

I spent two wonderful summers interning at Microsoft Research. I gained insights on commercial data center networks and industrial research lab from these precious experiences. I could not have had such wonderful experiences without my mentors Dr. Yongguang Zhang and Dr. Bozidar Radunovic, who were always helpful and inspirational.

Lastly, I would like to express my deepest appreciation to my parents and girlfriend for their unconditional support and encouragement.

VITA

2016	B.S. (Computer Science), Shanghai Jiao Tong University, Shanghai, China.
2017–2022	Teaching Assistant, Computer Science Department, UCLA, Los Angeles, CA, USA
2017, 2018	Research Intern at Microsoft Research, Redmond, WA, USA.
2016–2022	Graduate Student Researcher, Computer Science Department, UCLA, Los Angeles, CA, USA

PUBLICATIONS

Zhaowei Tan, Boyan Ding, Jinghao Zhao, Yunqi Guo, Songwu Lu, *"Breaking Cellular IoT with Forged Data-Plane Signaling: Attacks and Countermeasure,"* **ACM TOSN**, 2022.

Jinghao Zhao, Zhaowei Tan, Yifei Xu, Zhehui Zhang, Songwu Lu, *"SEED: A SIM-Based Solution to 5G Failures,"* **ACM SIGCOMM 2022**.

Zhaowei Tan, Jinghao Zhao, Yuanjie Li, Yifei Xu, Songwu Lu, *"Device-Based LTE Latency Reduction at the Application Layer,"* **USENIX NSDI 2021**.

Zhaowei Tan, Boyan Ding, Jinghao Zhao, Yunqi Guo, Songwu Lu, *"Data-Plane Signaling in Cellular IoT: Attacks and Defense,"* **ACM MobiCom 2021**.

Zhaowei Tan, Boyan Ding, Zhehui Zhang, Qianru Li, Yunqi Guo, Songwu Lu, *"Device-Centric Detection and Mitigation of Diameter Signaling Attacks against Mobile Core,"* **IEEE CNS 2021**.

Yunqi Guo*, Zhaowei Tan* (*: Co-Primary), Kaiyuan Chen, Songwu Lu, Ying Nian Wu, *"A Model Obfuscation Approach to IoT Security,"* **IEEE CNS 2021**.

Jinghao Zhao, Boyan Ding, Yunqi Guo, Zhaowei Tan, Songwu Lu, *"SecureSIM: Rethinking Authentication and Access Control for SIM/eSIM,"* **ACM MobiCom 2021**.

Yuanji Li, Chunyi Peng, Zhehui Zhang, Zhaowei Tan, Haotian Deng, Jinghao Zhao, Qianru Li, Yunqi Guo, Kai Liang, Boyan Ding, Hewu Li, Songwu Lu, *"Experience: A Five-Year Retrospective of MobileInsight,"* **ACM MobiCom 2021**.

Antonios Katsarakis, Yijun Ma, Zhaowei Tan, Andrew Bainbridge, Matthew Balkwill, Aleksandar Dragojevic, Boris Grot, Bozidar Radunovic, Yongguang Zhang, *"Exploiting Locality for Fast Distributed Transactions,"* **Eurosys 2021**.

Yunqi Guo, Zhaowei Tan, Songwu Lu, *"Towards Model-Centric Security for IoT Systems,"* **IEEE ICCCN 2020** *(invited paper).*

Zhaowei Tan, Yuanjie Li, Qianru Li, Zhehui Zhang, Zhehan Li, Songwu Lu, *"Enabling Mobile VR in LTE Networks: How Close Are We?"* **ACM SIGMETRICS 2018**.

Zhaowei Tan, Yuanjie Li, Qianru Li, Zhehui Zhang, Zhehan Li, Songwu Lu, *"Supporting Mobile VR in LTE Networks: How Close Are Qe? "* **Proceedings of the ACM on Measurement and Analysis of Computing Systems**, 2018

Haoyang Wu, Tao Wang, Zengwen Yuan, Chunyi Peng, Zhiwei Li, Zhaowei Tan, Boyan Ding, Xiaoguang Li, Yuanjie Li, Jun Liu, Songwu Lu, *"The Tick Programmable Low-Latency SDR System,"* **ACM MobiCom 2017 (Best Community Paper Award)**.

CHAPTER 1

Introduction

1.1 Security is Critical for 5G/4G Systems

Mobile Network Systems (5G/4G) have become an integral component in our daily life to access Internet and voice/text services. To date, it is the only large-scale system that provides ubiquitous network access for billions of devices worldwide. The global market of 5G has grown drastically to 67.5 Billion USD in 2022 and is expected to reach 1870 billion USD in 2030 [Res21]. Around the world, the estimated number of 5G/4G users has increased to 0.4B in 2021 (0.04B in North America) while the number is expected to increase by 10 times in 4 years [Glo21]. With the introduction of support of IoT devices (i.e., C-IoT including Category-M and Narrowband-IoT), 5G/4G system is projected to connect 52% of the global IoT applications by 2025 [Bla20]. By the end of 2020, Cat-M and NB-IoT have been deployed by 73 operators in 73 countries and regions [GSM21].

Given their global adoption and heavy usage, it is of great importance to secure 5G/4G systems. This is especially challenging considering the emerging applications, such as interactive Augmented Reality and Virtual Reality (AR/VR), telehealth, self-driving vehicles, industrial/agricultural IoT, etc. If 5G/4G is attacked by an adversary, the consequences can range from loss of revenue (e.g., a critical data message is manipulated) to even life-threatening events (e.g., a sensor running on cellular IoT technology fails to trigger an alarm). In addition, the applications pose extremely stringent requirements on 5G/4G networks, as they expect the network systems to provide carrier-grade reliability, high throughput, time-sensitive delivery, or year-long battery-saving. To satisfy these requirements, the 5G design might prioritize them and sacrifice the security level to some extent.

1.2 Insecurity of the Current 5G/4G Systems

Given the high priority of system security, 3GPP standards have introduced multiple mechanisms to defend against different attackers [3GP22b]. Mutual authentication between the device and the network offers the first fence. It uses the private keys stored in SIM and core network, which are barely accessible to attackers, to validate the identity of both sides. During this procedure, the network and device securely generate and exchange the same session key. After mutual authentication, the session key is used to generate keystream for each control-plane signaling message and data-plane user packet. To prevent using the same key for different packets, keystream algorithm also takes the unique sequence number of each packet as an argument for key generation. The network and device can use the keystream to encrypt and integrity-protect control-plane signaling and user packets with the proven-secure algorithms. Consequently, such messages cannot be decoded on or altered without internal attacks that could hack the private or session keys.

Nevertheless, numerous studies show that attacks are still feasible against 5G/4G systems. On the control plane, a few messages are not encrypted before mutual authentication, including broadcast messages from base stations or paging messages that are used to notify a certain device. Attacks have been designed to exploit them for revealing a victim's identity information (IMSI), manipulating user connection, or enforcing DoS variants [SBA15, SBP18, HCM18, YBS19, MO17, HRO16a]. A typical attacker will require deploying a fake base station (FBS). An FBS transmits strong signal to lure a victim device to connect to it; the FBS can then send or receive data to/from the victim and manipulate the messages. Data-Plane is also shown to be vulnerable, despite the protection from the control plane. With an FBS that relays mutual authentication procedure, an adversary can further act as a Man-in-the-Middle to hack encrypted data-plane packets and decode video calls or forge any contents [RKH20a, RKH19, RKH20b].

This dissertation further shows that, attacks are also possible after mutual authentication, without targeting the mutual authentication approach, using fake base stations, or insider attack for key compromise. This places 5G/4G devices under threat even with recent

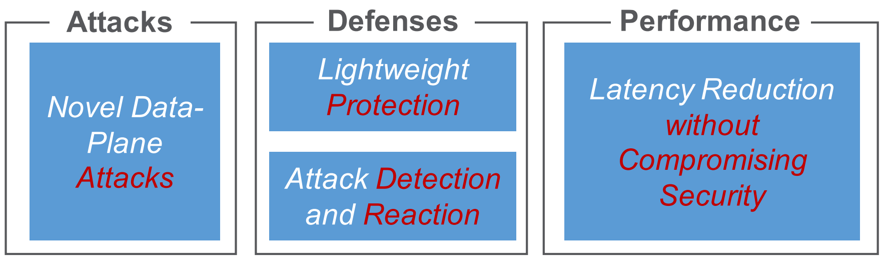

Attacks
Novel Data-Plane Attacks
Defenses
Lightweight Protection
Attack Detection and Reaction
Performance
Latency Reduction without Compromising Security

Figure 1.1: The overview of the contributions of this dissertation.

5G proposals to protecting signaling before mutual authentication or securing the mutual authentication against relay fake base station.

1.3 Contribution of this Dissertation

This dissertation designs, analyzes, implements, and evaluates both emerging threats to 5G/4G systems and our novel countermeasures. We first demonstrate that, even with the forthcoming 5G new security additions (added in new releases or discussed in 3GPP meetings) and thus a more restricted attack scenario, an adversary is still capable of launching novel attacks against mobile systems to inflict serious damages. We demonstrate that, without FBS, a key enabler of most attacks, the 5G/4G data plane still suffers from data forgery attacks after mutual authentication. This includes forging both data-plane user packets and signaling messages. On the other hand, we take a systematic approach to studying the countermeasures for the threats. The current security suffers due to the tradeoff with low overhead and high performance. We explore several design philosophies that are fundamentally different from the current approach in 5G and argue what is the most proper way to tackle each vulnerability. We leverage novel, fitting design philosophies to tackle these threats while not incurring new issues. Finally, we study a reversed problem: how to improve the performance without changing hardware or sacrificing security.

An overview is shown in Figure 1.1. Concretely, we will present the following components in this dissertation.

1.3.1 Novel Attacks under a Restricted Threat Model

After mutual authentication, user packets will be encrypted. Therefore, most existing attacks target unprotected messages before this procedure. One potential approach to forge data after mutual authentication through FBS, which sets up a Man-in-the-Middle node to flip the encrypted bits [RKH20b, RKH19]. FBS has been the key attack technique used by recent studies to forge data in 4G LTE [SBA15, SBP18, RJP16, HCM18, RKH20b, RKH19], but the latest 3GPP releases for 5G aim at eliminating FBS by enhancing the connection setup procedure [3GP20b, 3GP22b]. Without FBS, it seems impossible to launch any forgery attacks, as the message will not be accepted from a base station that the victim is not currently connected to. In addition, the data might be integrity protected, which will deny any forgery attacks.

However, we show that the above fences are insufficient to stop the attackers. Even if the victim device has *already been mutually authenticated with the legitimate 5G/4G network*, an attacker can still launch the data-plane attack. The key to launching the attack without FBS is the improper protection of data-plane meta-info. The meta-info refers to various parameters used to send data properly within 5G/4G standards. It is transmitted *in cleartext* and not strongly bound with control-plane security keys. As a result, even if the device is directly connected with the base station, the attacker can eavesdrop on the public channel between the device and the legitimate base station to catch meta-info and use them to forge messages. The forged data passes all checks on the receiver side, appearing to be from the legitimate sender.

`CDS`: Forging Data-Plane Signaling messages With the possible forgery without FBS, we first design a series of attacks that leverage data-plane signaling messages. They are used for scheduling, power control, reliable transfer, random access, time alignment, etc. Although the user packets and control-plane signaling are protected, we find that data-plane signaling that is used to facilitate data transfer is not secured. We design `CDS`, which forges such signaling messages to incur diversified yet serious damages. The data-plane signaling is widely believed to be not threatening; however, with carefully designed context and content, the forged data-plane messages can cause damages beyond simplistic DoS, including radio resource draining, prolonged data delivery, flexible throughput limiting, location privacy breach, packet delivery loop, and connection reset.

`FANE`: Manipulating Data-Plane Packets with Retransmission Although data packet integrity protection will disable the attacker from forging it, we note that this protection is absent in 4G and optional in 5G due to its high overhead. Although the adversary cannot directly forge arbitrary data packets due to mandatory encryption, an attacker can still flip the bits of the encrypted packets [RKH19, RKH20b]. We present `FANE` attack, which leverages the vulnerability that *retransmission reuses the key for the original transmission.* It can first eavesdrop on the original transmission, flip certain bits, and forge the manipulated data as the retransmission. With this attack methodology, the attacker does not need a FBS to "flip and forward" the data. The receiver cannot distinguish whether the retransmission is from the attacker or the device. With the forgery attacks, the attacker is capable of redirecting Web traffic, forging short data packets with arbitrary contents, or even intruding on the victim's privacy.

1.3.2 New Philosophy of Security Countermeasure in 5G/4G Systems

The conventional approach to 5G/4G security is a per-packet protection scheme, where each data packet or control-plane signaling is encrypted and integrity protected. However, given the high data throughput (up to Gbps) in 5G, not all messages can be protected (e.g., the data-plane signaling) to tradeoff for performance. The result is that any data or signaling not directly encrypted or integrity-protected can be forged by an adversary. In this dissertation, we explore different solution philosophies to securing 5G/4G data-plane from forgery attacks with little overhead.

`DP`^{2}: Protect Lightweight Meta-Info for Data-Plane Security We first explore a new solution for *securing meta-info*, instead of protecting each data message itself. The meta-info refers to various parameters that will be used by the sender to deliver data within 5G standards. Once they pass the checks at the receiver, the data will be accepted. Without accessing these parameters, an attacker cannot forge *any* valid message to be accepted by the victim receiver. The advantage of this approach is the lightweight nature of meta-info compared to heavy 5G data traffic. We identify all meta-info for 5G data-plane operations, and assess the necessity and viability to protect each. It turns out that, we do not need to protect all of them, but securing one is sufficient. We thus propose a solution $\texttt{DP}^2$ that secures the critical meta-info DCI, a scheduling meta-info that is managed by MAC layer. The DCI is needed for each data delivery, while being feasible to be protected. It overcomes the limitation of producing keystream for DCI. $\texttt{DP}^2$ enables efficient time- and frequency-based protection on DCI and reuses the current 5G NEA encryption algorithm. We implement our DCI protection scheme in a C library, and provide API for both device and base station.

CellDAM: Rootless, User-Space Detection and Reaction by Inspecting Lightweight Data-Plane Signaling Messages We next consider the alternative approach of detection and reaction on 5G data-plane attacks. The goal is to let the solution be immediately deployable without infrastructure upgrades or standard changes. This serves as a quick remedy to the legacy devices. Our solution relies on a key observation: to launch any forgery attack on the data plane, all known attacks might *trigger undesired data-plane signaling messages* at the device side. We thus propose CellDAM, a novel solution for **cell**ular data-plane attack **d**etection **a**nd **m**itigation on the device side. It has two salient features. First, instead of targeting certain message scenario, we *verify what's right.* By modeling the 5G data plane and detecting any anomaly, CellDAM can detect both known and unreported data-plane attacks. Second, CellDAM operates on the user-space with a novel "companion node" that can help analyze the traces from/to the device. It requires no extra privilege or firmware access while achieving both detection and damage mitigation at the device.

1.3.3 App Optimization without Sacrificing Security

In our previous components, we have demonstrated that adopting security solutions are usually constrained due to performance trade-off. We argue that, the reversed statement is also true. To optimize the performance of the radio access network, a solution might sacrifice security. For instance, many solutions [TLL18, LPY16] rely on acquiring cellular information for optimization and thus require root privilege. Unfortunately, exposing root privilege is considered dangerous [Lab17] and even inaccessible for most devices. Is there an approach to boost the access network performance, while *not sacrificing security?* We propose a new solution idea that tailors optimization for diverse applications on the APP layer without root. We demonstrate that, critical cellular parameters can be inferred by sending a controlled pattern of traffic and observing the consequences on the application layer. For a category of latency-sensitive applications such as AR/VR, reducing latency in the access network can be achieved by simply sending extra small packets. The timing of these packets relies on the inferred critical 5G/4G parameters. These steps leverage the inferred parameters and require no root access or change of firmware, network, or hardware.

1.4 Organization of this Dissertation

The remaining of this dissertation is organized as follows.

Chapter 2 introduces the background of 5G/4G system design and state-of-the-art security fences. We also introduce some vulnerabilities despite the 5G/4G security measures.

Chapter 3 presents an overview of our technical contributions. We justify the objective, philosophy, and methodology to attack, defend, and optimize 5G/4G system. This chapter offers highlights, features, and immediate gains for each design component.

Chapter 4 illustrates our newly-designed attacks that would work even in a restricted threat model. This makes the attacks more practical compared to the state-of-the-arts. The attacks can target both data and signaling, broadband and cellular IoTs.

Chapter 5-6 detail two distinct, complementary solutions to address the vulnerability, `DP`^{2} and `CellDAM`. The former studies the fundamental yet practical solution with standard updates to eliminate threats to 5G/4G systems. It explores a new philosophy of protecting meta-info to prevent message forgery. Meanwhile, `CellDAM` provides an immediate remedy to detect and mitigate the attacks from the device side. The methods can detect both known and unreported attacks and operates on user-space without any additional privilege.

Chapter 7 highlights the novel solution approach to application optimization without compromising security. We justify the importance of rootless solutions, which pose challenges on network optimization. We demonstrate `LRP`, which is a rootless solution applicable to latency-sensitive applications.

Chapter 8 presents our efforts in building real systems to validate our design. It supports two operational modes, 5G/4G broadband and narrowband-IoT. We implement and evaluate all components in this dissertation.

We summarize our results and lessons in Chapter 9 and propose a blueprint for immediate and future 5G/4G/xG security research.

CHAPTER 2

Background on Mobile Network Systems

2.1 5G/4G Systems: State-of-the-Art

2.1.1 System Architecture

5G system has 3 major components as shown in Figure 2.1: User Equipment (UE), Base stations (gNB), and 5G Core network (5GC). The UE is a 5G user device. It can be either a broadband mobile phone or an IoT device. A UE can equip a SIM card to gain access to mobile network services. The gNB powers up the 5G network and provides radio access network in its coverage area for UE. It also forwards data to and from the core network. 5GC includes several core components to provide user session/mobility management, user authentication, roaming, etc. On the data plane, it also bridges data packet delivery between the access network and the Internet. The core also charges the user traffic.

We note that, 4G shares a similar architecture but uses different terminologies (5GC→4G Core, gNB→eNB). In the rest of the dissertation, we use 5G component name for simplicity

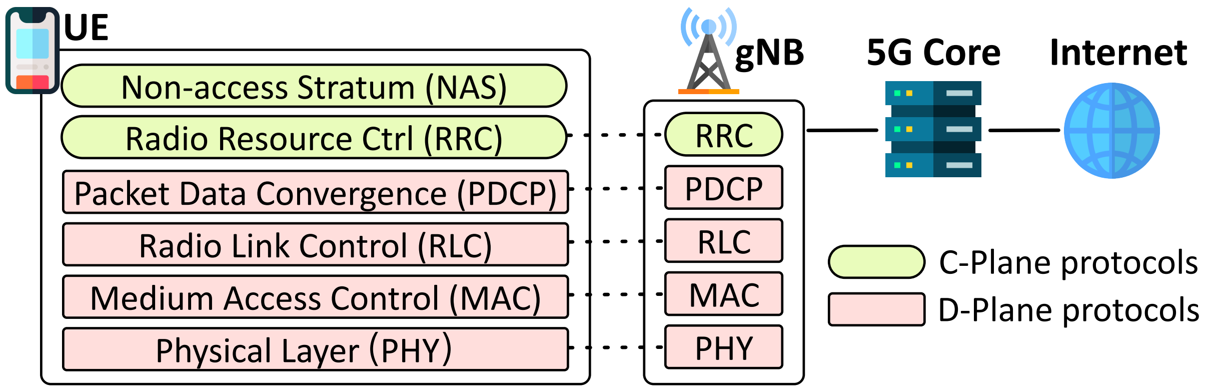


Figure 2.1: Architecture of 5G and its protocols.

if the discussion applies to both 5G and 4G systems, unless explicitly mentioned.

2.1.2 Protocol Stack

The 5G protocol stack consists of multiple protocols to implement both control-plane and data-plane functions. Non-access Stratum (NAS) and Radio Resource Control (RRC) are in charge of control-plane signaling. NAS protocol facilitates control-plane signaling message exchange between the UE and 5GC. RRC protocol is the channel between the UE and the gNB, which carries the data-plane parameters for set-up, power management, and handover behavior.

The data-plane enables IP packet delivery. We describe 4 protocols involved in data-plane operations: Packet Data Convergence (PDCP), Radio Link Control (RLC), Medium Access Control (MAC), and Physical Layer (PHY). PDCP is in charge of control-plane and data-plane packet encryption and integrity protection. RLC performs data concatenation and reorganization to ensure reliable, in-order data transfer. MAC is for radio access control. PHY performs wireless signal processing. The radio link is split into different channels: PDCCH/PUCCH (Physical Downlink/Uplink Control Channel) for exchanging signaling messages and PDSCH/PUSCH (Physical Downlink/Uplink Shared Channel) for exchanging data.

2.1.3 Data Delivery Procedure

The 5G data access takes a scheduling-based approach. gNB assigns radio resources by sending Downlink Control Information (DCI) to the UE through Physical Downlink Control Channel (PDCCH). DCI specifies the time, frequency, and parameters for packet delivery. Upon receiving DCI, the UE decodes data in Physical Downlink Shared Channel (PDSCH) or sends data in Physical Uplink Shared Channel (PUSCH).

Multiple HARQ processes run at MAC layer, with each identified with an ID. A data packet is associated with a HARQ ID. The receiver sends ACK/NACK with the HARQ ID based on whether the received data is decoded correctly. The sender issues retransmission

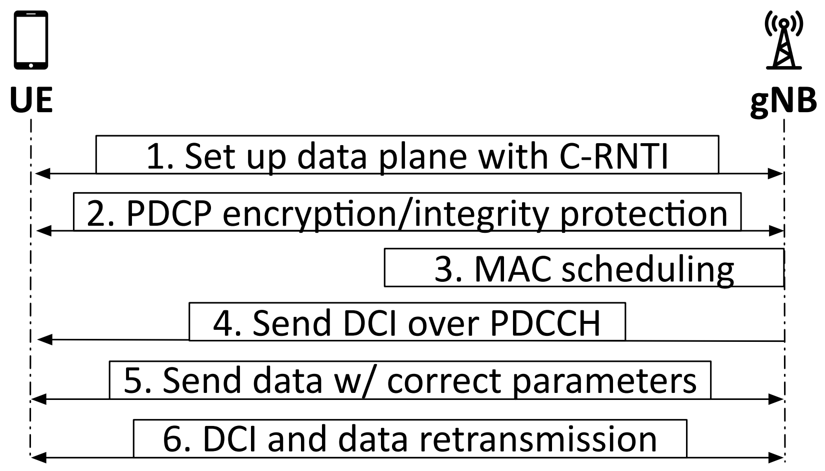

UE
gNB
1. Set up data plane with C-RNTI
2. PDCP encryption/integrity protection
3. MAC scheduling
4. Send DCI over PDCCH
5. Send data w/ correct parameters
6. DCI and data retransmission

Figure 2.2: Data delivery procedure in 5G.

upon NACK. Each HARQ process continues to send the next packet upon receiving an ACK. Therefore, multiple processes are used in 5G/4G to prevent head-of-line blocking.

We present the *core* steps to deliver uplink and downlink data in the standardized procedure, as shown in Figure 2.2.

①: Data plane setup. The device receives C-RNTI assignment from the gNB. It is a unique ID used by both sides to identify different devices under the same cell. For instance, when a gNB sends data, the device checks if the precoded C-RNTI matches with the assigned value to get the messages intended for itself. The gNB can update this ID any time when the connection persists.

②: PDCP processing for data security. When a 5G UE (for UL) or gNB (for DL) receives a pending IP data packet for delivery, it encrypts the packet and (optionally) adds integrity protection code to the packet. The keystream is generated using the algorithms and session key agreed during RRC setup.

③: Data scheduling. In this stage, the gNB schedules data transmission for the UE that has pending data to send or receive. The basic unit for scheduling in 5G is a resource block (RB), which indicates the time and frequency resource that is allocated to the UE to send or receive data.

④: DCI transmission. The scheduling information is carried in DCI (Downlink Control

Information) through PDCCH. The information is in the format of DCI, which can carry either uplink (type 0) or downlink (type 1) grant for data transmission. With the grant, RLC can segment or concatenate the PDCP packets based on the allocated size. The receiving UE uses the DCI to either decode DL data in PDSCH or send UL data over PUSCH.

⑤: Data transmission over the radio link. For both uplink and downlink, the data is required to use the correct encoding to be properly decoded. The parameters include the modulation scheme, HARQ process ID, etc. The data must use the correct parameters carried in DCI message in step ④.

⑥: Data retransmission. This step is required when the original transmission is corrupted. gNB produces a new DCI and the sender uses it to retransmit the data.

2.1.4 Signaling Messages in Control and Data Plane

The network and devices exchange control messages for user state and data session management. For example, a UE device needs to send messages to 5GC for registration in 5G network and establishment of data channel when it exits the airplane mode. Another example is the RRC message, in which the base station configures the data-plane parameters for the device (e.g., the periodicity of sending certain messages). The message could also carry mobility-related information, such as measurement configurations and handover commands.

In addition to signaling that traverses the control plane, 5G also adopts data-plane signaling between UE and gNB. 5G/4G uses such messages to facilitate data transfer. These data-plane signaling messages are also unicast data, transferred over PUSCH/PDSCH similar to data packets. The receiver of the signaling messages subsequently processes them based on the 3GPP standards. Data-plane signaling messages are designated to provide functions including power control, radio resource management, time alignment, etc. that are critical for successful and reliable data transfer between a gNB and a UE device. The data-plane signaling messages are different from control-plane signaling messages, which manage the devices' connection or mobility state. RLC control is the data-plane signaling for reliable transfer. It has only one type, STATUS PDU [3GP21f]. MAC encodes its data-plane

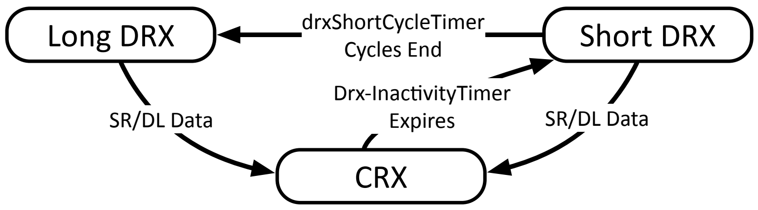


Figure 2.3: State transition for mobile network DRX power-saving.

signaling in a special structure, named Control Element (CE). There are 14 DL CEs and 14 UL CEs in the latest release [3GP21e], with each one used for a different purpose.

2.1.5 Power Saving

The power-saving mechanism DRX (Discontinuous Reception) , in a nutshell, is a technique for a device to save power over 5G/4G (see Figure 2.3). Instead of continuously waking up for potential downlink delivery (i.e., DCI) from the BS, the device might sleep in the absence of data transfer, thus reducing its energy consumption. In DRX, a device has three states: Long DRX Cycle, Short DRX Cycle, and Continuous Reception (CRX) [3GP19c]. In CRX, the device wakes up during the ON period to monitor downlink channels. In long/short DRX, the device only wakes up for a short period of time (set by the onDuration Timer) at the start of each DRX cycle. It dozes off during the OFF period for the remaining time.

The DRX state transition is shown in Figure 2.3. In the Long/Short cycle state, if any downlink data is received during the ON period, the device enters the CRX state and starts the drx-InactivityTimer. Upon sending an uplink data packet, the device initiates an SR request. It then switches to the CRX state as well. If the device receives downlink data or initiates another SR request, the timer restarts. The short DRX state is entered once the drx-InactivityTimer expires. In this state, the device enters long DRX after the number of drxShortCycleTimer short cycles. All such involved timer parameters are negotiated between the device and the BS during connection setup through RRC.

2.1.6 C-IoT: 5G/4G for IoT Devices

In addition to the broadband services for mobile devices, 5G/4G also supports simple, flexible modes for narrowband IoT devices with extended coverage and extremely low power requirement. This is called Cellular IoT (C-IoT) It provides "anytime, anywhere" wireless Internet access for IoT devices. It has defined two operation modes of Cat-M and NB-IoT through 3GPP specifications. They support low-rate, power-constrained machine-to-machine and Internet of Things applications. Cat-M operates at 1.4 MHz bandwidth. It can support data rates of 1 Mbps at maximum and voice call applications. Meanwhile, NB-IoT supports ultra-low complexity devices with extreme narrow bandwidth of 200 kHz. By the end of 2020, Cat-M and NB-IoT have been deployed by 73 operators in 73 countries and regions [GSM21]. The current Cat-M and NB-IoT designs already meet 5G's requirements [3GP17]. 5G RAN allows Cat-M and NB-IoT transmissions to be placed directly in a 5G frequency band [3GP21g]. 3GPP plans to rollout the standalone C-IoT in 5G from 2023 [ubl20]. It is projected that 52% of IoT applications will run on Cat-M and NB-IoT at the end of 2025 [Bla20].

2.2 Security Mechanisms in 5G System

2.2.1 Mutual Authentication

5G performs mutual authentication procedure, a critical security measure inherited from 4G with little change. The UE and the network perform a secure Authentication and Key Agreement (AKA) procedure during connection set-up, and establish session keys afterwards. The key is further used to create several session keys for NAS control-plane signaling, RRC control-plane signaling, and data packets. Each type maintains a separate, increasing sequence number to identify the messages.

2.2.2 Data Encryption and Integrity Protection

The corresponding key for control signaling or data packet will be used to generate a keystream for every packet based on several parameters, including the sequence number, direction, size, etc. [3GP22b]. The keystream is generated to prevent key reuse, as the input variables are different for each message. The correctness is ensured as all parameters will be synchronized or exchanged in cleartext between sender and receiver.

The sender also utilizes a keystream to generate an integrity code attached to the message. The receiver uses this code to verify the message integrity. 5G aims at enforcing integrity protection on all data packets. Given that ensuring integrity is of high overhead, adding it for high throughput 5G is considered unrealistic for certain devices. Consequently, 5G release 15 does not specify full-rate support for integrity protection due to its high overhead [3GP19a]. Nevertheless, every device and gNB should support integrity protection at full rate starting release 16, although its usage is still optional due to the increasing processing load on both the UE and the gNB [3GP22b]. Once enabled, this protection will effectively void some 4G attacks that leverage the lack of data packet integrity protection. For example, [RKH19, RKH20b] propose two data-plane attacks that can manipulate and even forge arbitrary data-plane packets. They can be protected with the new 5G mechanism.

3GPP also has discussed proposals to eliminate false base stations (FBS) [3GP20b]. FBS is considered a major threat to 5G/4G access network. An FBS appears as a legitimate base station by transmitting strong wireless signal and luring the victim device. It could subsequently establish connection with the victim device. It can then forge data to the victim device or relays data to the authentic gNB. With the new scheme from 3GPP to mitigate such attack methodology, forgery attacks with FBS [SBA15,SBP18,RJP16,HCM18,RKH20b, RKH19] will be mitigated.

2.2.3 Additional Security Measures for C-IoT

In addition to common 5G/4G security mechanisms, C-IoT employs additional security techniques. Most notably, a C-IoT device can encapsulate its data packets in NAS messages.

Therefore, the data packets will be both encrypted and integrity protected as a control-plane signaling message. A C-IoT device can specify that it enables this feature during the connection setup. The network will accept the request and notify the device if the feature is supported. This mechanism provides a quick alternative to protect C-IoT data packets before the full roll-out of the data packet integrity protection. It makes the attacker increasingly difficult to target C-IoT data-plane packets.

2.3 Vulnerabilities Found in 5G/4G Systems

In this section, we elaborate on the vulnerabilities on the data plane that enables the attacks. These vulnerabilities are not simple design slips. Instead, they are all 5G's compromise for providing low-overhead services. They trade off for high throughput or even system correctness.

2.3.1 Weak Binding Between Control- and Data-Plane IDs

One-time mutual authentication between the device and the 5G/4G network at the start of establishing radio connectivity through the standardized AKA procedure. However, device authentication on the control plane does not ensure data authenticity on the data plane for wireless transfer at underlying layers of PDCP, RLC, and MAC. Instead, the 5G/4G sublayers (PDCP, RLC, and MAC) all share the same C-RNTI as the device temporary session ID for data-plane transfer. There is no secure binding between C-RNTI and the control plane keys. This C-RNTI is transmitted over-the-air in cleartext, which can be eavesdropped by the attacker. An attacker can use various techniques to correlate a user identity with the C-RNTI [KRH19].

2.3.2 Cleartext Data-Plane Signaling and Protocol Headers

C-IoT does not protect data-plane signaling at RLC/MAC layers even after mutual authentication. Such messages are sent in *clear-text*, either as individual packets or as embedded packet headers. The examples include RLC Control (for data acknowledgment) and MAC control elements (for power control, exchanging scheduling information, etc.) If an attacker exploits them with other C-IoT vulnerabilities in PHY and MAC sublayers, it can both eavesdrop on these unencrypted messages and forge fake ones (detailed in later sections).

This design choice is an engineering trade-off as encrypting or integrity protecting data-plane signaling is nontrivial. Since current 5G/4G encryption and integrity check are enforced at PDCP, lower layers (RLC/MAC/PHY) cannot leverage such protections. Second, a keystream has to be adopted to avoid possible key reuse. PDCP uses the sequence number (SN) to generate consistent keys between the device and the eNB. In contrast, MAC does not have SN for each data-plane signaling message.

Unfortunately, such a design choice greatly compromises the security. Although conventional wisdom does not anticipate much damage beyond the simplistic DoS, cleartext data-plane signaling may cause a range of severe and unexpected damages with other vulnerabilities.

2.3.3 Optional Integrity Protection

5G aims to adopt new security measures to enforce integrity protection on all data packets. The data integrity protection incurs high overhead and is considered difficult for legacy devices. Although the UE and gNB should support integrity protection at the full speed, the usage is optional after considering the increased processing load on the UE and gNB [3GP22b]. If not enabled, those 4G data-plane attacks [RKH19,RKH20b], which manipulate and forge arbitrary data packets, remain effective in 5G.

CHAPTER 3

A New Look at 5G/4G Security

In this chapter, we build upon one observation that motivates this thesis: the current 5G/4G security mechanisms are insufficient to protect its data plane. The claim remains true even with new efforts from 5G to continuously improve its security, as novel attacks are still possible (§3.1) even without fake base station (FBS) or other unrealistic assumptions. The threats, however, are not a result of design mistakes. Instead, they result from trade-off or compromise for high-performance requirements of the 5G/4G system. Therefore, we explore new design philosophies for security countermeasures that incur much lower overhead compared with conventional wisdom (§3.2). We further show that, achieving better performance can be achieved without sacrificing any security (§3.3).

3.1 New Attacks against 5G/4G Data Plane without FBS

We propose new attacks against 5G/4G data plane, even with a more constrained threat model without FBS as man-in-the-middle. With the development of new 5G data-plane security fences, such as mitigating FBS or redesigning connection setup procedure, attacks in 5G are still deemed possible given the vulnerabilities in §2.3. The attacker can eavesdrop on the access control info and directly forge the data over-the-air without actually connecting to the device or base station. With the forgery, an array of attacks are feasible. On one hand, the attacker can forge data-plane signaling, which is sent over-the-air in cleartext. These messages are considered not threatening, but we show that they can incur significant damages. Besides, given the integrity protection being optional, the packets are still prone to manipulation attacks. We illustrate a new attack methodology to achieve so without using

FBS as man-in-the-middle.

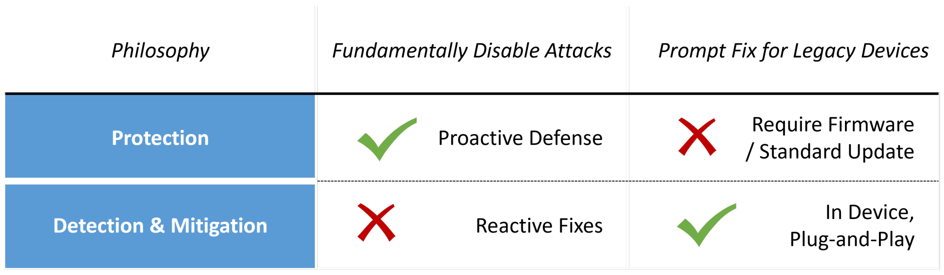
3.1.1 Attacks that Forge Data-plane Signaling

We assume the victim device has *already been mutually authenticated with the legitimate 5G/4G network.* Even with this restricted scenario, we spot new attacks that can threaten data plane operations. Specifically, we find that data-plane signaling is not secured. It is neither ciphered nor integrity protected. Data-plane signaling provides important control functions to facilitate data-plane packet transfer in 5G/4G, including random access, power control, radio resource request, time alignment, reliable transfer, etc. We design `CDS`, which forges such signaling messages. This data-plane signaling is widely believed to be not threatening. However, we combine it with other vulnerabilities in PHY/MAC/RLC protocols to launch attacks that can incur serious damages. It passes all checks on MAC and PHY layers, including accurate timing, access control, proper coding, etc. Thus, the forgery appears to be from a legitimate sender.

Damages We carefully design the context and content of forged messages to inflict diversified damages on devices beyond simplistic DoS, including radio resource draining, prolonged data delivery, flexible throughput limiting, location privacy breach, packet delivery loop, and connection reset. The attack damages include both problems in a single layer and cascading effects across multiple protocols.

3.1.2 Attacks that Forge Data-plane Packets

An attacker can also forge data-plane packets without FBS. Note that, it cannot directly forge arbitrary data packets due to mandatory encryption, while an attacker can still flip the bits of the encrypted packets [RKH19, RKH20b]. However, the challenge under the new threat model is that how to flip bits if a man-in-the-middle FBS does not exist. We present `FANE` attack, which leverages the vulnerability that *retransmission reuses the key for the original transmission.* Consequently, the attacker can first eavesdrop on the original transmission, flip certain bits, and forge the manipulated data as the retransmission. When



Philosophy	*Fundamentally Disable Attacks*	*Prompt Fix for Legacy Devices*
Protection	✓ Proactive Defense	✗ Require Firmware / Standard Update
Detection & Mitigation	✗ Reactive Fixes	✓ In Device, Plug-and-Play

Figure 3.1: Two complementary solution approaches.

timed correctly, the receiver cannot check whether the retransmission is from the attacker or the device.

Damages With the forgery attacks, the attacker is capable of redirecting Web traffic, forging short data packets with arbitrary contents, or even intruding on the victim's privacy with an IP inference attack.

3.2 New Philosophies for Defense Solutions against Data-Plane Attacks

To defend against such threats, the conventional approach is to design per-packet protection. This philosophy naturally incurs high overhead, especially given the high throughput in 5G. Enforcing integrity and encryption protection on all messages, including data packets and data-plane signaling, will demand further advancement in the hardware or security algorithms efficiency. Our solution, however, aims at protecting 5G data plane with a different philosophy. The goal is to design correct, practical, yet lightweight solutions with high security guarantee.

We explore two complementary solution approaches. For one, we design a proactive protection scheme that can fundamentally address the issues. By changing the standard and updating the firmware, a device will no longer be exposed to our proposed threats. Despite its potential, this type of solution cannot 1) address the security requirements of the legacy devices, and 2) will require time to be adopted and updated. Therefore, we also design passive detection and reaction schemes, which spot the potential threats and mitigate the attack damage. This approach does not require any hardware or network side assistance. We conclude the pros and cons of both choices in Figure 3.1. We now elaborate on their design details.

3.2.1 Protect Meta-Info for Data-Plane Protection

The fundamental cause lies in its per-packet protection scheme, which shows that any data or signaling not directly encrypted or integrity-protected can be forged by an adversary. In this dissertation, we take a different perspective on the problem, and explore an alternative solution of *securing meta-info rather than protecting each data message itself.* The meta-info refers to various parameters used to send data within 5G standards. Without accessing these parameters, an attacker cannot fabricate *any* valid message to be accepted by the victim receiver. We identify all meta-info for 5G data-plane operations, and assess the viability to protect each. It turns out that, we do not need to protect all of them, but securing one (i.e., DCI) is sufficient. We thus propose `DP`^{2}, a secure, low-overhead meta-info protection scheme for 5G data-plane security. `DP`^{2} secures DCI, a scheduling meta-info that is managed by MAC layer. It overcomes the limitation of producing keystream for DCI. `DP`^{2} enables efficient time- and frequency-based protection on DCI and reuses the current 5G NEA encryption algorithm. We implement our DCI protection scheme in a C library, and provide API for both device and base station. Our security analysis and empirical evaluation on an SDR-based testbed have validated that `DP`^{2} ensures data-plane security at marginal overhead.

3.2.2 User-Space Detection and Mitigation by Inspecting Signal

In this part of the dissertation, we take the alternative approach of detection and reaction on 5G data-plane attacks. The goal is to let the solution be immediately deployable without infrastructure upgrades or standard changes. Our solution relies on a key observation: to launch any forgery attack on the data plane, all known attacks might *trigger undesired data-plane signaling messages* at the device side. If an adversary launches a data-plane signaling attack via forged signaling, this forged message would be sent in the wrong context to incur damages; this can be readily detected. Moreover, attacks that forge data-plane user packets will also trigger undesired data-plane signaling, which is required to facilitate every packet transfer. We thus propose `CellDAM`, a novel solution for **cell**ular data-plane attack **d**etection **a**nd **m**itigation on the device side. `CellDAM` detects data-plane attacks by inspecting signaling messages, which are lightweight compared with the high-rate data packets themselves. Our trace analysis shows that, processing data-plane signaling involves 1-2 magnitude less overhead compared with monitoring all data packets. It requires no extra privilege or firmware access while achieving both detection and reaction at the device.

3.3 Rootless Inference for App Optimization without Compromising Security

To optimize the network performance for certain applications running on 5G/4G, traditional approaches require information on the current network, such as the parameters for data delivery, run-time channel condition, etc. This is a natural choice as an application shall adapt itself to the current network environment for the best performance. However, it is not possible to acquire such info without root privilege on the device. Enabling root privilege is considered a dangerous act that exposes new security threats [Lab17].

This dissertation argues that a rootless application can still infer the network info and use it for optimization. The core idea is, with domain knowledge, an application can send a few probing packets with a specific pattern and use their feedback to infer targeted network-specific parameters. We design and implement `LRP`, a device-based, software-only 5G/4G latency reduction solution that is readily usable for *every* commodity smartphone device. A salient feature of `LRP` is that it does not require root/system privilege, firmware modification, or hardware change. It is thus applicable to every off-the-shelf commodity device, including Android and iOS. `LRP` explores the application-driven network latency reduction at the device, which complements those existing infrastructure-centric solutions that are good at downlink latency reduction. `LRP` focuses on reducing LTE uplink network latency, which is the bottleneck based on our empirical analysis.

CHAPTER 4

Novel Data Plane Forgery Attacks

In this chapter, we aim at designing new attacks that bypass the emerging 5G security measures, such as the mitigation of the fake base station (FBS). In particular, this dissertation focuses on data-plane forgery attack, in which the attacker tries to send forged data to the victim device or authenticated base station. We overview the requirements for launching such attacks in §4.1. §4.2-§4.3 elaborate on the details of the forgery attacks for signaling and data packets, respectively. §4.4 and §4.5 explain how to leverage the forged signaling and data packets to cause serious damages.

4.1 Threat Model and Requirements of Launching Data-Plane Forgery

4.1.1 Threat Model

We consider an active attacker who aims at breaking 5G/4G data-plane service. It targets *forging either user data packets or signaling messages from users or infrastructure.* The attacker is in the proximity of the victim device and knows which 5G cell the device is connected to. This is possible as the attacker can target its nearby devices or use other attacks such as fingerprinting to locate certain victim [KRH19]. The adversary is capable of eavesdropping on the physical channels and transmitting data on them. We consider the attacker can choose to *comply with 3GPP standards or stay non-standard-compliant* for more powerful data-plane attacks.

We assume the victim device is directly connected to a legitimate gNB. The attacker does not rely on FBS to launch the attack given the incoming security measures against this methodology [3GP20b]. We do *not* consider an insider attack where the adversary will know all security keys stored in SIM/networks, or compromise the 5G base station (i.e., gNB).

4.1.2 Challenge 1: Pass the PHY Checks

To craft a data-plane message that could be decoded at PHY, the forgery attacker must handle the following. First, the attacker adopts the correct common configurations for data forgery. Second, the encoding needs to be correctly used for each transmission based on DCI. They ensure that the forged data can be decoded correctly. Third, the forgery must be sent with proper power to guarantee that the forged data can overshadow the legitimate transmission while not causing saturation effect.

4.1.3 Challenge 2: Pass the MAC Checks

In 5G/4G, the gNB allocates radio resources for both UL and DL transmissions, with RB (or RU in NB-IoT C-IoT networks) being time-frequency resources scheduled to a device. Both uplink and downlink data must be sent in authorized RBs (time-frequency blocks). An attacker sending data in the unauthorized RB/RU will be immediately rejected. Therefore, the forgery attacker must acquire the scheduling information to send the forged data in the correct RB/RU. RB assignment and encoding parameters are continuously changed upon each transmission. The gNB notifies each device of such info through MAC layer meta-info DCI.

4.1.4 Challenge 3: Leverage Forged Messages for Damages

Although data-plane signaling messages are not protected, a forged data-plane signaling message might cause little or no impact even if it is accepted and decoded correctly. Moreover, if the message content is out of context, it could be directly ignored or discarded by MAC/RLC. Therefore, the attacker needs to properly decide the content and the sequence to forge messages to inflict serious damages.

For data-plane user packets, the attacker can leverage the lack of integrity protection to flip the encrypted bits to forge packets. However, this attack methodology must know the bits before the encryption to forge the correct content. It also needs to bypass or elude the possible higher-layer protection schemes.

4.2 `CDS`: Forging Data-Plane Signaling with Overshadowing

In this section, we will introduce how to forge data-plane signaling messages and bypass all the security fences. This effectively addresses challenge 1-2. We will start by introducing attacks that target *C-IoT* devices. This is because, the signaling messages in C-IoT networks provide more functionality compared to broadband devices. In addition, some attacks are more damaging to narrowband devices, such as draining the battery of devices. Therefore, we will base our discussion on attacking IoT devices and we will later introduce how the attacks could be adapted to broadband devices.

Note that, C-IoT currently operates in 4G only. Therefore, we will use 4G terminologies such as eNB to describe the attacks in this section.

4.2.1 Forge Signaling with Correct Encoding

Configurations The attacker needs to adopt the common configurations for all UL or DL signaling messages. They can be acquired from monitoring broadcast messages. Some C-IoT specific ones are dmrs-Config-r13 (to encode UL reference signal correctly), ul-ReferenceSignalNPUSCH-r13 (to get group hopping to send UL reference signal formula), etc.

Data Encoding The encoding and modulation parameters should be consistent with the assigned value in the DCI, so that the forged signaling can be correctly decoded. In DL forging, if the original eNB uses transmission diversity, the attacker needs to apply correct precoding and layer mapping. When using RU in UL, NB-IoT splits a single RB further to enable fine-grained resource allocation. `CDS` derives these parameters by sub-carrier spacing

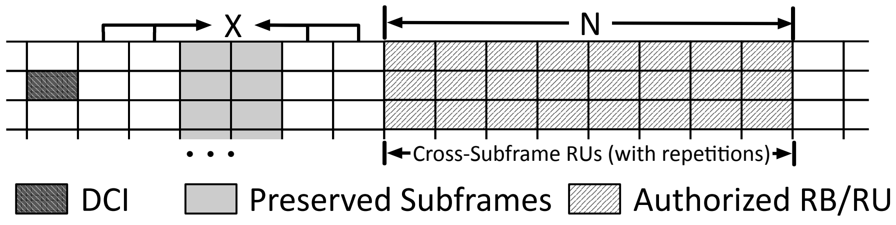


Figure 4.1: Cross-subframe scheduling for C-IoT.

(available in broadcast) and the number of sub-carriers in one RU (available in DCI).

Victim C-RNTI `CDS` scrambles the data-plane signaling with correct C-RNTI, which is the identity assigned to each device. Due to the vulnerability that the C-RNTI is sent in cleartext, multiple approaches are available to acquire it if its IMSI [DPW16, SBA15, Pos18, KRH19, MO17, HEC19] or traffic pattern is known [KRH19].

Power Requirement The data forging requires capture effect, in which the attacker injects a stronger signal to force the victim to decode it instead of the legitimate signal with lower power. In order to be successfully decoded at the victim, the power of injected signal needs to reach certain levels relative to the original one. For example, with NB-IoT, where QPSK is mainly used, an SNR (signal-to-noise ratio) of 6.2dB can ensure successful decoding [Sch13].

4.2.2 Forge Messages with Correct RB

Time synchronization The attacker must first synchronize with the eNB to eavesdrop and forge data. Our approach resembles the one in [YBS19] but with one difference: C-IoT adopts an extra clock HFN (Hyper Frame Number) to increase the time span for synchronization. C-IoT devices need this timer as they could be in sleep mode for a much longer time compared to a broadband device. The attacker can synchronize this HFN from eavesdropping on the SIB2 broadcast messages.

Acquire scheduling information After synchronization, the attacker must acquire the scheduling information (i.e., DCI) for forging the data at correct RB/RU. Since DCI is transmitted without encryption over-the-air, the adversary eavesdrops on the PDCCH and finds the DCI scheduled for the victim.

C-IoT adopts cross-subframe scheduling; the timing to use a DCI is specified by its included parameters. A DCI indicates a set of RB/RU that a device can use to send UL or receive DL data. The RBs/RUs are not in the same subframe as the DCI. Instead, a delay exists between DCI and the authorized RB/RU. This is depicted in Figure 4.1. The delay is either a constant or can be calculated from DCI [3GP19b]. Its value is based on direction (UL or DL), spectrum usage techniques (TDD or FDD), access technologies (Cat-M or NB-IoT), and the delay field in the DCI (scheduled by the eNB). In any case, it is no less than 2ms. This leaves sufficient time for the attacker to prepare the signaling forgery.

The above design choice is not a mistake. C-IoT adopts cross-subframe scheduling for two reasons. First, IoT devices are limited in processing capacity. Handling control and data channels separately in time helps reduce processing complexity. Second, the channel bandwidth is low for C-IoT (especially NB-IoT). A single subframe is often insufficient to carry both data and control information.

Another option is to forge a DCI directly. The attacker sends a forged DCI in PDCCH and uses any parameter of its choice to forge the data. The benefit of this approach is, the attacker does not have to wait for any data transmission from the victim device and can initiate the attack at any time. This comes with certain limitations: this approach only works for downlink while forging a DCI of UL grant will certainly fail. This is because the uplink data will be decoded by the eNB. If the attacker uses the configurations in the fake DCI instead of the RB/RU notified by the eNB, the transmission will fail to pass the checks at the eNB. For downlink, this is not an issue as the device will decode the forged data with the forged DCI.

Note that, a fake DCI can only be successfully decoded by the victim device in DRX ON state. Otherwise, the device in DRX OFF does not listen to the channel for data or signaling messages. However, forging a DCI during DRX OFF will not cause any issue, as the victim device will not be capable of detecting it without listening to the channel. Therefore, the attacker can keep forging DCI and opportunistically send forged signaling messages once the DCI forgery succeeds. In addition, the attacker can avoid continuous DCI sending by eavesdropping on the channel. If the attacker detects that the device has active data transmission, it knows that the device will keep DRX ON for a certain period (usually more than tens of ms). The attacker can forge a DCI in this time span.

Note that the attacker has to infer two more parameters for forging authorized data. First, a C-IoT DCI can grant a device with resources across multiple subframes. Due to the narrow channel in C-IoT, it is common that a data transfer exceeds the maximum RBs/RUs in a single subframe. Second, C-IoT designs repetition method [3GP19b] for coverage enhancement. Doubling the transmission results in 3 dB of coverage gain [Cha19]. The eNB allows the device to repeatedly send the same data multiple times with a single grant. The attacker can infer both timing span and repetition number from the DCI.

Interpret valid RB/RU from DCI Given the DCI and the parameters received from the broadcast channels, the attacker derives which resource blocks and encoding scheme to use for signaling forgery. [3GP19b] mandates that a UL grant is used x UL subframes after receiving the DCI. A consecutive N RBs/RUs are assigned to the user (Figure 4.1).

• **UL in CAT-M** UL grant for CAT-M is transmitted as format 6-0A/6-0B DCI in mPDCCH. The behavior is standardized in [3GP19b], Section 8.0. For FDD, $x = 4$. For TDD, the attacker can infer x from the subframe number and the cell's TDD configurations. The information is available to the attacker. In any subframe-configuration combination, $x \geq 3$. An attacker needs to skip DL frames in TDD and half-duplex FDD. Since an RB is fixed in time length, N is solely dependent on the repetition number. Therefore, an attacker can derive $N = N_{Rep}$ from the DCI.

• **DL in CAT-M** DL scheduling information grant in CAT-M is transmitted as format 6-1A/6-1B DCI in mPDCCH. Regardless of FDD or TDD, $x = 2$ (Section 7.1.11 in [3GP19b]). Similar to UL, the attacker needs to skip UL frames and derive N from the repetition number.

• **UL in NB-IoT** UL grant is transmitted as format N0 DCI in nPDCCH (Section 16.6 in [3GP19b]). Since NB-IoT has a much narrower bandwidth, a device needs to wait longer for a UL/DL transmission. A DCI carries a field called scheduling delay index (I_{delay}). It is a discrete value from $\{0, 1, 2, 3\}$, and $x = 8 \cdot 2^{I_{delay}}$. The attacker needs to skip UL frames in TDD and preserved frames in FDD.

An attacker in NB-IoT calculates N as follows. It gets the parameter (I_{Rep}, I_{Sc}, I_{RU}) in the DCI [3GP19b, 3GP21d]. From I_{Sc}, it can calculate the number of slots for each RU as N^{UL}_{slots}. From I_{RU}, the attacker learns the number of consecutive RUs assigned to the victim, as N_{RU}. The data is repeated by N_{Rep} times. The total time slots can be calculated as $N = N_{RU} N^{UL}_{slots} N_{Rep}$.

• **DL in NB-IoT** DL scheduling information grant in NB-IoT is transmitted as format N1 DCI in nPDCCH. Based on Section 16.4 in [3GP19b], $x = 5 + k_0$, where k_0 can be calculated from I_{delay} in the received DCI. Similar to our discussion in UL in NB-IoT, the attacker needs to skip the preserved subframes. We next calculate N. The DCI includes I_{SF} and I_{Rep}. From the information, the attacker can infer the consecutive subframes for each assignment (N_{SF}) and repetition counts (N_{Rep}). The total consecutive frames can be calculated as $N = N_{SF} N_{Rep}$.

4.2.3 Applying Signaling Forgery for Broadband

We note that, the same attack methodology can be adapted for broadband devices. Although broadband and C-IoT have differences in channel design or device capability, they share some similarities in their features. In specific, the encoding scheme and resource block assignments are still exposed to the attacker. Using an eavesdropping attacker, the adversary can use the inferred information to send the forgery. However, `CDS` need to be adapted for 5G/4G broadband with three caveats. 1) The channels are not designed the same. The attacker needs to target the corresponding channels for control and data signaling; 2) Broadband DL has a different scheduling mechanism, as the downlink scheduling DCI and actual data delivery could happen in the same time. For this type of forgery attack, the attacker can also forge the DCI. Since all the parameters are unknown and DCI is not protected, such an attempt will succeed. In addition, the uplink forgery attacks can remain the same, as 5G/4G also uses cross-subframe scheduling for uplink.

4.3 FANE: Forging Data-Plane Packets with Retransmission

Forging data works for broadband only Notably, C-IoT has a special operational mode called data-over-NAS (§2.2.3). Although this method is designed to lower energy consumption by removing data bearer, it also results in extra security protection for data packets. The data packets are both encrypted and integrity protected similar to control-plane signaling NAS messages. This mechanism provides a quick alternative to protect C-IoT data packets before full roll-out of the data packet integrity protection. It prevents forgery attacks on data packets. Therefore, in our discussion of data forgery attacks, we only target broadband devices, where integrity protection is absent or optional due to high overhead on heavy data rate.

4.3.1 Challenges and Solution Ideas of Forging Data Packets

Breaking the access control for data packet forgery can utilize the same methodology as CDS. This is because the data packets are delivered similarly as the data-plane signaling messages. However, since data packets are encrypted, the attacker cannot directly forge any encrypted data. It needs to flip the bits on the encrypted data. This raises several new challenges to launching the flipping without a FBS, which can store, flip, and forward the data.

Naive approaches To modify data over the air, one approach is to send data y from the attack node and the receiver will superposition y and ENC(x, k) into F(ENC(x, k), y) = ENC(x', k). However, such y cannot be derived without the unknown ENC(x, k) and F(). Another approach is to send a strong signal of ENC(x', k) that overshadows the original data. However, this is also not possible as ENC(x', k) is unknown in advance, as the key k is unknown.

Our solution: Leveraging retransmission We take a different approach and design FANE. It first prevents the receiver from accepting ENC(x, k) by injecting noises. It later retransmits ENC(x', k). This is possible, as the cipher key k can be learned from ENC(x, k) observed from the first transmission. The attacker gets ready before the retransmission to derive the key and generate a new ciphertext. To make the idea work, we address three concrete issues.

Issue 1: How to ensure retransmissions from `FANE` are accepted? 5G/4G communication is synchronized with its grant-based UL transfer. If an attacker sends a fake packet in an unallocated RB, it will be rejected.

Vulnerability 1: Weak access control for retransmission The retransmission in 5G/4G also relies on DCI. Therefore, `CDS` can be applied for inferring RB assigned for data transmission. One exception is that 4G might use the synchronized mode, where the retransmission will send the retransmission in fixed RBs after the original transmission. However, this mode is even more vulnerable as the attacker can infer the RB by simply eavesdropping on transmission.

Issue 2: How to prevent retransmission from the authentic sender? Even if legitimate RBs are used, signals can interfere with retransmissions from the victim device. We cannot simply corrupt them, since the noises will also corrupt the forged retransmissions. `FANE` must exclusively use the legitimate RBs.

Vulnerability 2: Unprotected retransmission indicator The HARQ retransmission is *unprotected*. ACK/NACK is a one-bit indicator without the integrity check. This bit can be easily flipped from NACK to ACK via corruption [TLL18]. If NACK is not correctly decoded, the sender does not initiate retransmission. HARQ retransmission is introduced for fast loss recovery. High-layer protocols (RLC in AM mode and TCP) ensure reliable data transfer. This vulnerability could address Issue 2: If the attacker blocks the original transmission *and* the subsequent NACK, the sender will falsely believe retransmission is not needed, while the receiver still expects retransmission. The attacker thus forges retransmission of manipulated data, without any interference from the authentic sender.

Issue 3: How to ensure the encryption key remains unchanged? The attacker needs to ensure the key to decipher the manipulated packet is consistent with the encryption key for the authentic packet. If the manipulated data is ENC(x', k), its decipher key must be k. Otherwise, the attacker fails to forge the packet to the intended content. This is

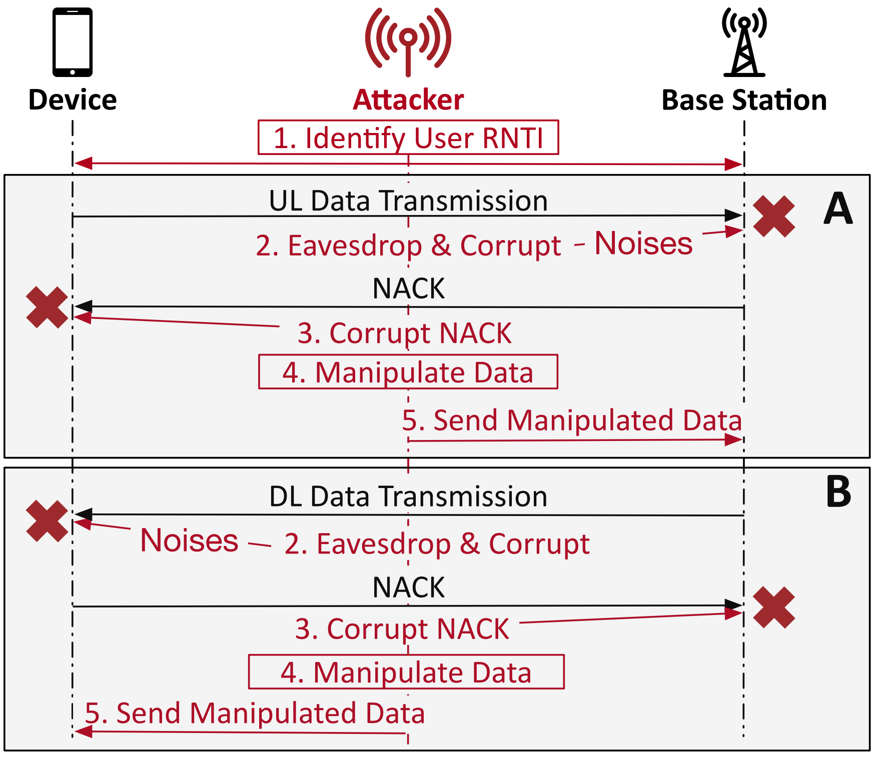


Figure 4.2: Attack procedure for MitM without FBS.

demanding as 5G/4G adopts stream cipher key, where the key is updated for every packet based on the count, etc.

Vulnerability 3: Key reuse upon retransmission Although stream cipher key is used for every PDCP packet, underlying protocols are unaware of the key. They do not update the counter for the key generation or re-cipher the packet upon retransmission. Therefore, multiple RLC, MAC, or PHY packets can be ciphered with the same key over the air. Our attack exploits a special case: *key re-use upon HARQ retransmission*. The MAC-layer HARQ uses the same ciphered packet upon retransmissions. We could leverage the vulnerability to address Issue 3. The attacker blocks the initial transmission and eavesdrops on the ciphered packet. It then manipulates the packet and sends it via retransmission. Since the victim receiver uses the same key for retransmission, it thus deciphers the forged data.

4.3.2 The Complete Attack Procedure

The detailed attack procedure is shown in Figure 4.2. In ①, the attacker acquires the victim user's C-RNTI used in transmission. The following steps have two symmetric branches, A for UL and B for DL. In step ②, the attack captures the UL/DL data and simultaneously corrupts the transmission by sending noises. The receiver will issue retransmission by sending an NACK. In step ③, the attacker corrupts the NACK. As a result, the sender will not perform retransmission. However, the sender is still expecting the retransmission. In step ④, the attacker prepares the manipulated data based on the decoded ciphertext. In step ⑤, the attacker sends manipulated UL/DL data as forged retransmission.

Our contribution The procedure follows similar approaches in `CDS`. ②, ③, and ⑤ are key steps for MitM with HARQ retransmission. ① is a preparation phase that uses the known vulnerability of vulnerable C-RNTI. ④ leverages the known vulnerability of malleable 5G/4G ciphering.

We next explain the details for each step and emphasis the vulnerability leveraged to launch it. We will also demonstrate that each step in our attack is still feasible in the 5G networks.

4.3.2.1 Step ①: Identify the victim's C-RNTI

As we discussed, C-RNTI is necessary to decode and scramble data. The attack must acquire the victim's C-RNTI to enable the attacks.

Unprotected device C-RNTI C-RNTI can be easily captured over-the-air, as it is not protected during the assignment. An attacker can monitor the RACH channel and read cleartext RNTI in the messages or decode any channel that scrambles with C-RNTI to learn its value.

Capturing RNTI for a victim user The attacker can match the C-RNTI with the victim user if its other identity is acquired by the attacker. Victim's C-RNTI can be learned if its IMSI is leaked, which is possible by IMSI catchers and other methods [DPW16,SBA15, Pos18, KRH19, MO17, HEC19]. Additionally, a fingerprinting attack can be used to find the victim's C-RNTI [KRH19, bae22, RKH19]. Both 5G and 4G use the same design for C-RNTI [3GP20a, 3GP20e].

4.3.2.2 Step ② A: Decode and Corrupt UL data

This step forces the retransmission by corrupting the UL data transmission using noises. Meanwhile, it decodes the UL data in PUSCH channel for manipulating the retransmission. To find UL data from the victim device, the attacker monitors PDCCH channel and searches for UL grant (DCI 0) for the target user (identified by C-RNTI). A UL grant specifies *which resource blocks* the device can use *4ms later.* All channels can be trivially located if the attacker checks broadcast messages [3GP20c, 3GP19b]. The attacker can infer which part of PUSCH to corrupt by finding UL grant in PDCCH.

Monitoring PDCCH for DCI 0 The attacker first decodes PCFICH (Physical Control Format Indicator Channel), which carries critical information to decode PDCCH. In PDCCH, a DCI is transmitted in several consecutive RBs, called a CCE (Control Channel Element). Based on the user's assigned C-RNTI, only a subset of CCEs can potentially contain the DCI for the user. The attacker can follow the standardized equation [3GP19b] and search all the candidate CCEs to determine if there is a DCI 0 for the intended user.

Decoding and Corrupting Uplink Data The UL data in PUSCH can be decoded with the captured DCI 0. The attacker locates the RBs used by the device and applies the standardized decoding mechanism. It simultaneously sends noises in those RBs to the BS for corrupting the transmission. Note that the attacker needs to be careful that the noises should not corrupt its own decoding. We will discuss how to implement this by utilizing directional antennas in §8.3.

4.3.2.3 Step ② B: Decode and Corrupt DL data

Similar to step ② A, the attacker is required to learn when and where the BS sends DL data in PDSCH. Unlike UL transmission, PDSCH is decoded by information in PDCCH sent in the same time slot. It seems that the attacker cannot locate DL data in advance; however, DL data to manipulate is sent from malicious DNS/Web servers in our attacks (§3). Therefore, it can notify the attack node before sending the fake DL data. The attack node can then limit the time period that the DL data can be observed in milliseconds. It decodes the PDCCH for the victim's data and corrupts the channel. Although this potentially affects other users, they can quickly recover the corrupted data by retransmission.

4.3.2.4 Step ③ A/B: Corrupt NACK

As shown in the vulnerability of unprotected NACK, the attacker could corrupt the NACK to stop the data sender from initiating retransmission. Meanwhile, the data receiver is still expecting it. This opens an opportunity for the attacker to forge manipulated data as retransmission. For both UL/DL in 4G, the NACK is fixed 4ms after the original data transmission. An attacker can thus easily find and corrupt the target NACK in PHICH (for UL) and PUCCH (for DL). Both control channels are in the fixed location in the 4G band. Corrupting the NACK can be done by sending white noise in the channels in the time slot.

5G keeps a similar HARQ design with two slight changes [3GP20a]. First, instead of a dedicated PHICH, 5G uses PDCCH for UL data ACK [3GP21i]. `FANE` can still corrupt the NACK by the same method but on a different channel. The second difference is that 5G allows a flexible time difference between data and HARQ response. `FANE` can still learn this time difference in the DCI information for the data transmission [3GP21h].

4.3.2.5 Step ④ A/B: Manipulate Ciphered Data

It is shown in [RKH19] that 5G/4G encryption is malleable: the ciphering is done by XOR-ing the plaintext with the stream cipher key. If certain bits are known, the attacker can manipulate them by applying an XOR mask to the ciphertext, which is equivalent to applying the same mask on the receiver side. In this way, the attacker can morph the ciphered known bits.

Compensating for checksum `FANE` also needs to ensure the correctness of IP header checksum, which is calculated by summing up all 16-bit words using one's complement addition. Similar to [RKH19], we adopt the idea of canceling out checksum change with other known fields. In a UL packet sent from the device, the DSCP/ECN field (octet 1) is always 0, and TTL field (octet 8) is fixed (e.g. default to 64 on Android). On DL, the attacker can set the 16-bit identification field to a predetermined value, which originates from attacker-controlled malicious servers. In both cases, `FANE` has enough fields to compensate checksum. After the address manipulation, the attacker calculates the change in the checksum and modifies the known fields to cancel out the change.

4.3.2.6 Step ⑤ A/B: Send Manipulated Data

The attacker directly injects the manipulated packet into the channel at the correct timing as retransmission. The attacker needs to re-encode the manipulated data into PHY subframes, using the correct redundancy version (RV) for retransmission, so that the receiving side can recognize and accept the "retransmission". For the UL attack, as the victim device's retransmission is suppressed in step ③, the attacker can directly send on PUSCH without being interfered with. For DL, the BS might schedule the RBs for other users. Thus, the attacker needs to generate its own DCI and retransmission in PDCCH and PDSCH. The attacker can send strong enough signal to overshadow the authentic signal from the BS [YBS19].

4.4 Leverage the Forged Data-Plane Signaling

In this section, we introduce the details for `CDS` data-plane signaling attacks, as shown in Table 4.1. Their damages range from breaking the data access (such as limiting the throughput) to privacy breach (such as localizing a victim device). They leverage the `CDS` forgery techniques in the previous section. We present two major categories: single-protocol attacks (§4.4.1) and cross-layer attacks (§4.4.2).

Table 4.1: Overview of data-plane signaling message attacks and their damages.

Category	Attack Damage	Message(s)	Impacted Protocol(s)	Direction
Single-Protocol	Radio resource draining	Buffer Status Report (MAC)	MAC	UL
	Prolonged packet delivery	DRX Command (MAC)	MAC	DL
	Flexible throughput limiting	Power Headroom (MAC)	MAC	UL
Cross-Layer	Device localization	Time Advance, RACH (MAC)	MAC & RRC	UL/DL
	Packet delivery loop	RLC Control (RLC)	RLC & MAC	UL/DL
	Connection reset	AS RAI (MAC)	MAC & RRC	UL
	Multicast Disabling	SC-PTM Stop Indication (MAC)	MAC & RRC	DL

4.4.1 Single-Protocol Attacks

Single-protocol attacks aim at breaking one protocol in either victim device or network-side eNB. An attacker can launch them by forging a single message that targets the protocol to be attacked. For each attack, we will introduce the message that an attacker can leverage and the concrete steps of the attack.

4.4.1.1 Radio Resource Draining

In this attack, the attacker drains the UL resource in a short period, during which no device is capable of sending UL data. The attacker achieves so by forging a Buffer Status Report (BSR). This message forces the eNB to schedule excessive resource to the victim device. Given the limited bandwidth in C-IoT, other devices will be denied from sending UL data.

Buffer Status Report A C-IoT device can send a UL BSR in MAC. It includes how much UL data is waiting in the device buffer. The BSR is an index, which maps to a range of the pending data size [3GP21e]. The eNB will process it and schedule sufficient UL grants for the buffered UL data.

Attack Details The attacker forges a BSR that indicates the victim device has much UL data to be sent. It can use any positive index (e.g. 30) in the forged BSR, which indicates the pending buffer in the BSR (could be as large as MBs), as specified in the standard [3GP21e]. The eNB subsequently grants sufficient resources to this device. The attacker can repeatedly forge BSR messages to consume more radio resources.

Attack Damages The eNB will assign grants that are sufficient for the device to send all UL data. The amount of wasted resource is thus the same as the requested amount in the BSR. Studies show that an eNB tends to assign sufficient resource for a single device before serving the others [BBM19]. Therefore, when the network assigns all resources to the target device, other devices in the same network suffer from no access to radio resource. This is especially aggravated by the fact that C-IoT networks have limited bandwidth compared to broadband networks. The attacker can forge multiple BSR messages and disable the network for seconds.

4.4.1.2 Prolonged Packet Delivery

The attacker can leverage forged data-plane signaling to prolong a packet's delivery, without tempering the connection or forcing a DoS on the victim device. It turns the device into Discontinuous Reception (DRX) OFF state (sleep mode) by leveraging a MAC message called DRX Command. The device cannot receive data until the sleep mode ends.

C-IoT DRX C-IoT devices adopt DRX mechanism to save energy. Specifically, an eNB configures certain "ON" and "OFF" state periodically. The eNB only sends DL data during the next ON state. Since the device expects no data during DRX OFF, it turns off data reception and thus saves energy. The eNB forwards the related parameters, such as the duration of ON period and periodicity of a cycle, to C-IoT devices.

DRX Command During DRX ON, the eNB can ask a device to terminate DRX ON early, if it expects no more DL data in this period. An eNB does so by sending a MAC control element DRX command to the device. When a device receives a DRX command, it enters the DRX OFF immediately. This further saves energy for power-constraint C-IoT devices.

Attack Details The attacker first detects DRX ON state by eavesdropping on the PDCCH and decodes DCI. The device is in the ON state if a DL grant is found. The attacker thus forges a DRX command using that DL grant and the device will falsely change its state to DRX OFF.

Attack Damage When the eNB attempts to send more DL data during this DRX ON duration, the device cannot receive the data immediately. Instead, it has to wait for the next DRX ON state. In normal C-IoT networks, the latency could be hundreds of milliseconds. If the C-IoT device enables extended DRX (eDRX), the DRX OFF period could be even longer. This further prolongs the latency to seconds.

4.4.1.3 Flexible Throughput Limiting

The attacker leverages the data-plane signaling, Power headroom (PHR) messages, to limit the data throughput of the device. This attack is flexible and stealthy: Instead of persistently blocking the device's access (like the attack in §4.4.1.1), the throughput drop results from channel fluctuation from the victim's perspective.

Power headroom PHR is an uplink control message that indicates whether the device has extra transmission power to send UL data with higher throughput. Its value equals to device's max transmission power minus the current PUSCH Power. If the value is positive, the device has extra power for higher throughput. Otherwise, the current power consumption exceeds the device's capability. Upon receiving PHR, the eNB adjusts the UL scheduling accordingly.

Attack Details The attacker forges a PHR message to eNB. In this message, the included PHR indicates negative power headroom. The attacker uses value (0..10) as the content, which means the current PUSCH power exceeds max available transmission power [3GP21e]. This convinces the eNB that the device is suffering from high power so it needs to schedule data with smaller MCS. The eNB thus reduces scheduled UL throughput to meet the (fake) power requirements.

Note that the device might later report a PHR: when it uses the limited throughput to send data, it can notify the eNB that it has sufficient power to support higher throughput. The attacker can monitor PUSCH and observe this PHR. When the attacker detects it, the attack can be launched again.

Attack Damage The attack appears as a normal channel condition fluctuation. For the eNB, it thinks that the victim device runs out of energy. From the device's perspective, network congestion is a potential reason for the throughput drop. Therefore, neither side can detect the attack. Besides, the attacker can launch the attack anytime. The 3GPP standard allows sending PHR at any frequency. When eNB increases the throughput, it can send an additional PHR.

4.4.2 Advanced Cross-Layer Attacks

In this section, we detail the `CDS` attacks that inflict cross-layer damages on C-IoT protocols. For each attack, we introduce the involved messages and the attack procedure.

4.4.2.1 Device Localization

This attack targets localizing the static victim C-IoT device within the same cell it resides in. The attacker can breach the privacy of a C-IoT device. To infer the device location, we leverage the cleartext field Timing Advance (TA) in the random access procedure.

Timing Advance TA is a DL MAC control element used to synchronize the UL and DL due to propagation delay from C-IoT device to eNB. A device needs to advance the timing of its UL transmission to use the assigned RB correctly. Therefore, eNB sends TA in MAC CE during random access (RACH) or whenever any correction is required during RRC connected state. TA includes a discrete timing offset T_A. Upon receiving the offset, the C-IoT device adjusts the UL timing accordingly.

T_A reflects the propagation delay from the eNB to the device. Since the delay is affected by the distance between the eNB and the device, it can be used to infer the location of the device. The distance $d = 3 \cdot 10^8 \cdot (T_A \cdot 8/30720000)$. Note that T_A is a positive integer from 0 to 63. Therefore, a T_A can locate the device in a circular ring. Its inner radius is d and the breadth equals to $3 \cdot 10^8 \cdot (1 \cdot 8/30720000) = 78m$. This is shown in Figure 4.3(a). The Non-line-of-sight (NLOS) condition might introduce a small difference between the propagation path and the real distance, but this error can rarely exceed one TA value. A device with the same distance to the eNB will be assigned with the same TA value in Line-of-sight (LOS) or

NLOS settings. It will not substantially decrease the accuracy, as the device is still in the potential area inferred from TA.

Acquire the location of the eNB To locate the device, the attacker needs to get the location of the eNB so that it can use TA to infer the device's location. An attacker cannot trivially get this information over-the-air. One approach is to leverage the open cell database, such as [Cel21]. It returns the geo-location of the eNB given a cell ID, which can be observed by the attacker. Another active approach is to inversely use TA. The attacker uses its GPS location and TA to infer the location of the eNB. This is shown in Figure 4.3(b). It moves around and portrays the possible location of the eNB. It then takes the intersection of the potential areas as the location of the eNB.

Infer the device's distance to multiple eNBs If the attacker knows the distances of a C-IoT device to multiple cells, the area of its potential location can be further reduced by taking the intersection. Figure 4.3(c) depicts the idea. Consequently, an attacker forces the C-IoT device to connect to multiple cells and eavesdrop on TA under each eNB. According to standard [3GP21c], when the UL data is not synchronized, the device will re-initiate RACH procedure. If the number of attempts exceeds the limit, it initiates RACH and RRC connection to another cell. Based on this behavior, the attacker repeatedly sends noises on the reference signals. This only needs transmission power comparable to the victim device [LJL16]. The victim device will be forced to connect to another cell. If the attacker is in the coverage area of this new cell, it camps on the new cell for eavesdropping TA and sending noises. If the new cell is from a different physical eNB, the attacker eavesdrops on the TA value. The attacker can repeat the procedure until it cannot connect to the new cell. It gets the victim's TA in different BSes to enhance the localization accuracy.

Attack Damage The device localization attack is effective against a device in the coverage of multiple cells. It is suitable for localizing a static or quasi-stationary device. The attacker can position the victim device without compromising any internal data, such as the victim's GPS information. In addition to a specific victim, a malicious attacker can also use this technique to locate all active local devices (e.g. alarms) in a cell.

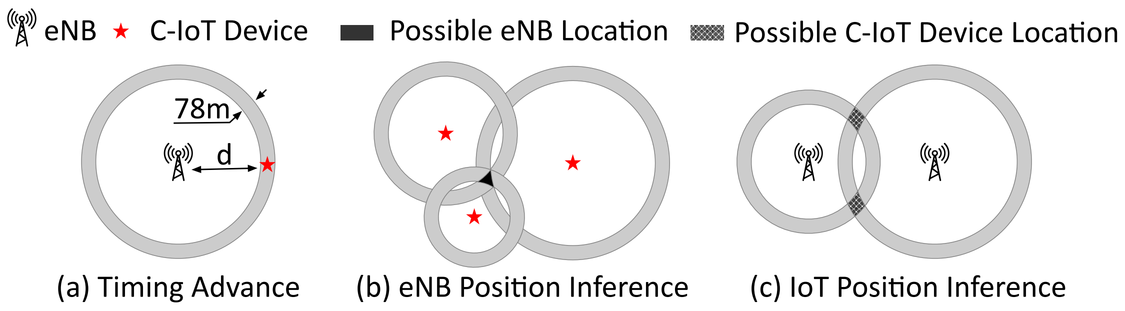

eNB
C-IoT Device
Possible eNB Location
Possible C-IoT Device Location
78m
d
(a) Timing Advance
(b) eNB Position Inference
(c) IoT Position Inference

Figure 4.3: Leverage TA to locate victim device/eNB.

4.4.2.2 Packet Delivery Loop

In this attack, the attacker forces the C-IoT device to repeatedly send the same packets instead of the new ones. The C-IoT device still has access to the radio resource but transmits repeated data. This forces the energy-constraint C-IoT devices to waste a large amount of power, quickly draining their battery. The attacker achieves so by sending RLC control messages.

RLC Control RLC Control is used by RLC protocol to ensure reliable transfer. In RLC AM (Acknowledged Mode, the most common mode used in C-IoT), C-IoT device and eNB will initiate RLC control messages to notify the other side whether the data reception is successful. This is triggered when a timer expires or the other side sends a polling bit. An ACK in RLC Control specifies the sequence numbers (SN) of the packets successfully transmitted. When some packet SNs are missing, RLC control includes an NACK that explicitly specifies the corrupted data. Note that RLC control is different from MAC ACK/NACK which is an unreliable indicator for quick recovery (<10ms), while RLC control is used to ensure reliable transfer. The receiver of an RLC control checks the content and acts accordingly. If an ACK is present, the receiver will clear the specified data in the buffer. If an NACK is present, the receiver of the NACK will read the content within the message and initiate the retransmission. The corrupted data packets will be forwarded to MAC.

Attack Details The attacker first eavesdrops on the RLC control exchange over-the-air by eavesdropping on PDCCH for DCI and decoding data channels. This allows the attacker to get the highest acknowledged SN (SN_m). It subsequently forges an NACK with a sent but unacknowledged SN (e.g., SN_m+2). Upon reception of this message, eNB/device retransmits the packets with the SN included in the forged RLC Control.

Attack Damage The RLC retransmission in MAC has a *higher priority* compared to the normal data transmission. The retransmission thus blocks the new data. Moreover, the attacker can repeatedly forge this message and cause the loop. The device thus loses UL access, while still logically connected to the eNB. Meanwhile, the constant retransmissions keep the device in DRX ON, quickly draining its limited power. In the extreme case where the attacker constantly forges messages, the device never enters DRX OFF.

4.4.2.3 Connection Reset

The attacker can reset the RRC connection by forging an AS RAI. This turns the device to IDLE state and might terminate any stateful connection (e.g. TCP).

Access Stratum Release Assistance Indication (AS RAI) AS RAI is a C-IoT-specific feature introduced in Release 16 [3GP21e]. A C-IoT device uses this message to notify the eNB whether it expects any subsequent data transfer with an indicator. The purpose is to save energy when the device has no more data transmission.

Attack Details The attacker forges an AS RAI control element with AS RAI value as `01`. The attacker embeds an RAI in a UL MAC control element called DCQR and AS RAI. In the message, the DCQR part can be any arbitrary content. The eNB falsely regards that no following data is expected from the C-IoT device.

Attack Damages The eNB falsely regards that no data is expected from the C-IoT device. Although AS RAI is a MAC layer message, it impacts RRC connection between the victim device and the eNB. A forged AS RAI indicating no data terminates the RRC connection with the victim device. After the attack, the victim might initiate a re-connection. The attacker can monitor the channel and repeatedly forge AS RAI attacks for consistent connection reset.

4.4.2.4 Multicast Disabling

In this attack, the adversary sends a signaling message to release the device from multicast data transmission. The forged signaling message is SC-PTM Stop Indication. It is a control element in MAC layer, but can trigger cross-layer damages that release the multicast session.

Multicast in C-IoT In the C-IoT network, the device and network can set up a multicast session. This scheme is called Single Cell Point To Multipoint, or SC-PTM. The network can then assign a G-RNTI identifier to every device within this group. With the G-RNTI, the devices can receive the DCI from PDCCH and decode the multicast data.

SC-PTM Stop Indication SC-PTM Stop Indication is a C-IoT-specific MAC layer CE introduced in 3GPP [3GP21e]. The eNB may transmit the SC-PTM Stop Indication to a multicast group that indicates the transmission in the multicast channel with a G-RNTI is stopped.

Attack Details The attacker first acquires the G-RNTI by enumerating the possible values in PDCCH transmission. If decoding the DCI with a G-RNTI results in correct CRC, the attacker infers the correct value. It then forges an SC-PTM Stop Indication message to the victim device encoded with the G-RNTI.

Attack Damages Upon receiving the message, the victim device will send the notification to higher layer to terminate the multicast (MBMS) session. The device will no longer be able to receive messages from the registered group, although it is still active. Certain devices that only enable multicast transmission will fail to receive any data and might even release the connection. An attacker with sufficient power can affect multiple devices if the forged message is correctly decoded by them.

4.5 Leverage the Forged Data-Plane Packets

The key requirement for forging a data packet is that the original content (before encryption) is known. Only this way, the attacker can flip the bits and forge the intended data. We

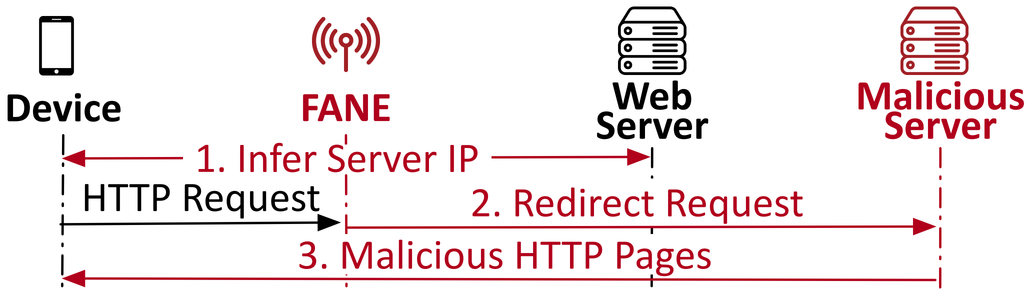


Figure 4.4: Attack against HTTP Web access.

target the IP header, whose fields can be potentially known for specific packets. In §4.5.1 and §4.5.2, we target the destination address of HTTP packets address which is known. In §4.5.3, we use `FANE` to launch an inference attack on packet IP address.

4.5.1 Attack on HTTP Web Access in 5G/4G

We first describe a cross-layer attack against HTTP-based mobile Web access in 5G/4G networks.

Attack This attack enables the adversary to intercept, modify and forward any HTTP request in an HTTP session, thus exhibiting a concrete form of MitM threat. It does not exploit DNS query and resolution, but directly operates on the HTTP requests for an ongoing or new HTTP session.

The attack procedure is illustrated in Figure 4.4. In Step 1, the encrypted HTTP request is intercepted by `FANE`. The ciphered destination IP address is inferred using the component attack of IP address inference. `FANE` replaces the destination IP address with the IP address of the malicious Web server. In Step 2, the twisted HTTP request is redirected to the malicious Web server, which returns any malicious content the attacker wishes to forge in Step 3.

The attack targets mobile Web access based on HTTP only, where Internet HTTP traffic has decreased to below 30% in traffic volume [w3t21a]. It also has a limitation. If mobile carrier deploys stateful firewall and NAT, the attack might not work with an ongoing HTTP session. However, it still works for a new HTTP session by intercepting and redirecting the

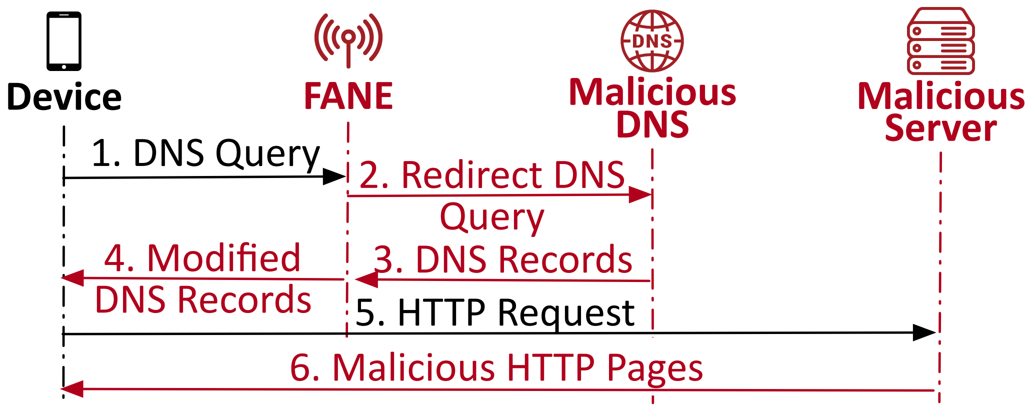


Figure 4.5: Attack against HTTPS based Web access.

TCP connection setup to the malicious Web server.

4.5.2 Attack on HTTPS Web Access in 5G/4G

We next present a more sophisticated attack that exploits 5G/4G vulnerabilities to break the HTTPS-based Web access.

Requirement HTTPS can further provide end-to-end protection against MitM attacks. As a secure extension of HTTP [Res00], HTTPS encrypts all data transfers using Transport Layer Security (TLS). HTTPS requires a trusted certification authority (CA) to sign server digital certificates. When visiting a Website via HTTPS, a user queries CA and verifies server authenticity. HTTPS thus prevents MitM attackers, which intercept and modify authentic data transfer.

Even if a Web server deploys HTTPS, a user can still send insecure HTTP requests to it before using HTTPS. HTTP Strict Transport Security (HSTS) [HJB12] is thus introduced. A Web server may configure HSTS for its clients. Any browser that sees this flag will add the Website to its HSTS list. The browser will always visit this Web page via HTTPS until HSTS expires. The browser also preloads HSTS for a list of Websites. However, statistics [w3t21b] show that HSTS is used by 18.1% of all Websites. This thus leaves 81.9% of HTTPS-enabled Websites being vulnerable.

Attack This attack leverages `FANE` and the vulnerable HSTS deployment. It tricks the victim device to directly connect with the malicious Web server by manipulating the DNS query, when initiating the HTTPS session.

The attack procedure has 6 steps, as shown in Figure 4.5. In Step 1, `FANE` intercepts the DNS query from the victim device to initiate the HTTPS session. In Step 2, `FANE` modifies the "dst IP address" in the DNS query, and redirects the forged query to the malicious DNS server controlled by the attacker. In Step 3, the server responds with a fake DNS reply that has the name resolution pointing to the IP address of the malicious Web server. In Step 4, this reply is intercepted by `FANE`, which modifies the "src IP address" in the DNS reply so that it appears to come back from the authentic DNS server, and is delivered to the victim device. In Step 5, the victim establishes its connection with the malicious Web server, which mandates the use of HTTP instead of HTTPS and supplies malicious content to the victim in Step 6.

Applicability Our attack works for a variety of scenarios. First, it is against HTTPS, but is also effective upon HTTP. Moreover, even though an authentic Web server deploys HTTPS, the initial request to it could still be in HTTP. The malicious server subsequently forces the Web session to use HTTP; this is known as SSL stripping attack [Mar09].

Second, although HSTS ensures that the Web session always uses HTTPS to a certain website, less than 20% of the domains enable HSTS to date [w3t21b]. Moreover, HSTS may expire in the browser, unless the browser statically preloads HSTS for this domain (only 10k domains in Chrome [Goo21]).

Third, the attack remains effective with stateful firewalls and NAT. The dst IP address in the UL DNS request (step ②) and the src IP address in the response (step ③) are consistent. Therefore, state checking at firewalls cannot detect it.

4.5.3 IP Inference Attack

With `FANE`, we next describe an attack, which infers the runtime IP address from the ciphered packets in 5G/4G, thus breaching privacy of the victim or the communicating server.

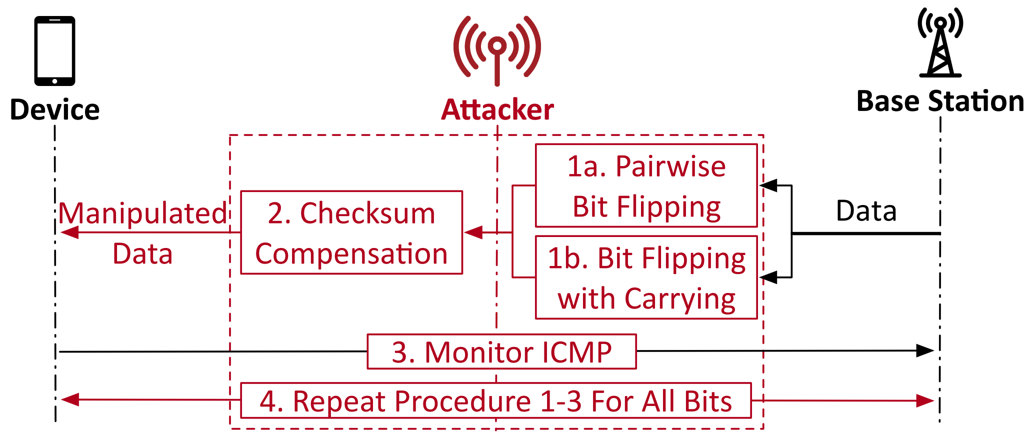

Device
Attacker
Base Station
1a. Pairwise Bit Flipping
1b. Bit Flipping with Carrying
Data
2. Checksum Compensation
Manipulated Data
3. Monitor ICMP
4. Repeat Procedure 1-3 For All Bits

Figure 4.6: Procedure of IP inference attack.

4.5.3.1 Issues and Approaches

Idea to infer a single bit The key idea is to guess one or more unknown bits by flipping them while learning whether the guess is correct or not by using the IP checksum as the indicator. Suppose we flip two ciphered bits with the same offset at different 16-bit words within the IP header. This amounts to flipping both bits in plaintext after deciphering. If they differ in original plaintext, with one of them being 0 and the other being 1, flipping both will not affect the IP checksum. It will be processed at the receiver. Otherwise, the packet is discarded at the receiver because of checksum failure before being processed by the IP protocol. If one bit has a known value, we can infer the other bit if correct IP checksum is observed. We next address two concrete issues.

How to know if the checksum fails? The attack can only decide the value of an unknown bit based on whether the packet is discarded by the network or not. However, this information is not observable by the attack node or the malicious server. Therefore, the attacker needs to find a way to make the effect directly observable.

Approach To make the effect visible to the attacker, we further modify certain fields in the IP header to trigger an ICMP error message upon successful reception, still by bit flipping, while keeping the IP header checksum unchanged. As ICMP message is generated *after* checksum [Bak95], the attack node can indirectly observe whether checksum fails at the receiver by observing the existence of ICMP response.

How to handle bits without a pairing bit of known value? Unfortunately, not all bits in the IP address field have a corresponding known bit of the same offset that can be used in the aforementioned method. Many bits within the IP header are unknown, while some other bits are known but cannot be changed (e.g. the version field). Therefore, we need to find an alternative inference method for those bits.

Approach We can utilize "carry" or "borrow" in checksum computation to infer an unknown address bit with adjacent known bits. This method is based on the following observation: flipping two bits of offset i from 0 to 1 yields the same effect as if flipping a single bit of offset $(i + 1) \bmod 16$ from 0 to 1; the same applies to flipping from 1 to 0.

4.5.3.2 IP Inference Attack Details

The attack workflow is shown in Figure 4.6: ① For a ciphered DL packet, the attacker flips one bit within source IP address and another known bit in the header. This step has two possible branches depending on bit offset: A for pairwise bit flipping, and B for flipping bits with carrying. ② The attacker modifies the protocol field and compensates checksum at other fields to keep the IP header checksum unchanged. ③ The attacker monitors the UL channels and checks if an "ICMP protocol unreachable" message is present. It thus determines the modified bit in the source address. ④ Repeat the procedure on those packets with the same IP source address, until all address bits are inferred.

Step ① A: Pairwise Bit Flipping The adversary flips some bits to help decide those unknown bits in the source IP address of DL packets. The easiest form is to use a pair of bits with the same offset in different 16-bit fields of the IP header, one being in the source address, the other being a bit with known value but changed without affecting delivery.

An example is the highest bit of the 3-bit flags field reserved in IPv4. This bit is always 0 in IP packets, but setting it to 1 does not affect delivery [Bak95]. We flip this bit and another bit with the same offset in the source IP address field. If the latter is 1, the IP header checksum remains unchanged after both bits are flipped; if it is 0, the header checksum would change and the IP packet becomes invalid.

Step ① B: Flipping Bits with Carrying This step provides an alternative for the unknown bits without a corresponding known bit that can be used. We can infer all other bits starting from the bit pairs inferred with known bits in the last step. It works as follows.

For the example of reserved flag bit, we know the original reserved flag (which is a 0) and two bits with the same offset 15 within the source IP address. Given these bits, there should be at least two bits with identical 0 or 1. We flip these identical two bits and another bit of next offset (0) within the source IP address. If the original value of the former pair and the latter differ, their effects on the header checksum cancel out; otherwise, the actual header checksum changes and the IP packet is invalid. Combined with ICMP response, we can infer both bits from offset 0 with known bits of offset 15.

The same technique can be applied onward for the next offsets. For example, if the two bits of offset 0 are identical, we can directly apply the method to infer two bits of offset 1. Otherwise, we can first convert it to the previous case with checksum-preserving transformation, where we flip two previous bits (offset 15) with identical value, and one bit on offset 0 with the opposite value. We apply it to the next offset. This can be done in a cascaded fashion for consecutive different bit pairs. We thus infer all bits in the src address.

Step ②&③: Trigger and Recognize ICMP packets The DL packet with changed source IP address is further modified to trigger predictable ICMP responses at the receiver, when the IP header checksum is correct. This is done by modifying the "protocol" field in the IP header. The receiver sends an ICMP "Protocol Unreachable" message, upon receiving an IP packet with a protocol ID it does not handle [RKH20b]. For common TCP packets with the protocol field being 6, we change it to 5 (Stream Protocol, no longer used in practice). The change to the IP header checksum is compensated by adding 1 to the ECN field in the IP header, which is observed to be always 0 in our traces.

Step ④: Repeat bit process. Since the attacker repeats bit inference for every bit to learn the IP address, it is important to find DL packets with the same source IP. The attacker may use TCP retransmission. Since inferring a bit in IP results in transmission failure, the sender retransmits upon TCP timeout [PAC00].

CHAPTER 5

Defend Against Forgery Attacks with Meta-Info Protection

In this chapter, we introduce the first countermeasure philosophy. Given the opportunity to redesign protocol and security measures, we propose a proactive solution that can defend against forgery attacks, including `CDS` and `FANE`. In §5.1, we introduce the idea of how to protect the data-plane without incurring the overhead from forced data integrity protection. This is done by securing the necessary information used to deliver each data. In §5.2, we introduce the solution idea to protect meta-info DCI for protecting data plane. §5.3 details the solution and §5.4 proves its security even against an advanced attacker who is aware of the scheme.

5.1 Protect Meta-Info, Not Data Messages

5.1.1 Disadvantages of Conventional Approaches and a New Way Out

The conventional way to protect data-plane authenticity is through per-packet integrity protection. It can be done either end-to-end (e.g., using HTTPS or TLS at sender and receiver) or at the radio link (in the PDCP layer). The sender side can encrypt and integrity protect every application or PDCP packet with a stream cipher key. The receiving side can generate the same key to decrypt and integrity check the message.

However, this approach suffers from severe overhead caused by integrity protection. Although 5G standards enable integrity protection for all data rates, it can be disabled for certain devices due to high overhead. In addition, the current scheme overlooks data-plane signaling messages in 5G link layers, which cannot be protected with higher layer protections on data packets or with reasonable overhead.

The fundamental overhead lies in the trade-off between performance and security. To ensure reasonable performance, data protection is not enforced. However, we argue that, to prevent against data forgery attack, we can protect *necessary* information instead of data itself. Without getting the info, the attacker will fail to initiate a successful forgery.

Meta-Info Protection In this dissertation, we explore a novel protection approach. We depart from the conventional end-to-end security approaches or PDCP-layer data packet integrity protection against forgery attacks. We study a new scheme that protects meta-info to prevent data-plane forgery. The meta-info is *strictly regulated by the network*. Even if the attacker does not follow the standardized data transfer procedure, it cannot forge the data-plane packets or signaling without knowing the meta-info. Protecting it ensures that an attacker cannot forge *any* malicious data.

Moreover, this solution approach has the potential to incur small overhead. The meta-info is smaller and much more lightweight compared to data traffic in 5G, as we will detail later in this section. Therefore, we attempt to design a solution that leverages this feature to incur lower overhead than conventional approaches.

5.1.2 Exploring Possible Solution Choices of Meta-info Protection

We summarize the available meta-info on 5G data-plane protocols in Table 5.1.

Meta-info in PDCP/RLC PDCP contains meta-info of sequence numbers. These numbers are used as parameters to generate security keystream. Sender and receiver exchange such info carried in the PDCP header for encryption and integrity protection. The RLC header contains segmentation information for the receiving side to reassemble the packets. As PDCP packets can be segmented and concatenated, the receiver requires the RLC meta-info to correctly recover the original PDCP (and thus IP) packets.

If an attacker cannot acquire correct PDCP or RLC meta-info, it will fail to forge PDCP data packet. Such meta-info is vulnerable due to being sent in cleartext [RKH19]. However,

Table 5.1: List of meta-info in 5G data-plane protocols and its relationship with data (both signaling and packets).

Protocol	Meta-Info	Description	Relation to Data
PDCP	Sequence Number	Incremental identifier for each data packet	Each data packet uses it to generate keystream for encryption/decryption and/or integrity check
RLC	Segment Info	Used to reassemble segments at receiver	Receiver uses it to reconstruct PDCP (and thus IP) packets
MAC	DCI	Resource assignment and coding modulation/parameters	Resource blocks, MCS, and modulation for each data delivery; assigned by gNB
PHY	C-RNTI	A PHY identifier	Any data needs to be precoded with correct C-RNTI for the receiver to identify correct sender/recipient

such PDCP/RLC meta-info is not necessary for data-plane delivery. Consequently, protecting such meta-info does not secure lower layer messages. For instance, a MAC CE can be forged without going through PDCP or RLC [TDZ21].

MAC/PHY meta-info is necessary for each data delivery Every message delivery, including data-plane signaling, data packets, or control signaling, must go through PHY and MAC layers to be successfully encoded and sent. Therefore, even a non-standard-compliant data forgery attack must pass all checks at PHY and MAC to be correctly decoded and accepted. To this end, an attacker must set up meta-info correctly at MAC and PHY for transmissions. This is a necessary condition to forge either data-plane signaling messages (from MAC and RLC) or data packets (from PDCP and IP).

Do we need to protect all meta-info? Protecting meta-info in all protocols used in data transfer offers one option. This will block any attempt to forge data packets or data-plane signaling. However, protecting *every* meta-info is unnecessary. The RLC and PDCP meta-info would not be used in every delivery; protecting such meta-info still allows attacks such as `CDS` to forge data-plane signaling. For MAC and PHY, each meta-info is necessary for data delivery. As long as one is protected, an attacker cannot succeed in the forgery. Securing both PHY and MAC meta-info would incur high overhead.

We thus aim to find the better option between PHY and MAC meta-info for data-plane protection. To this end, we discuss and study the features of PHY layer C-RNTI and MAC layer DCI. We check whether each is viable without unrealistic 5G design changes or high overhead.

5.1.2.1 Protect Physical Layer ID (C-RNTI)

If the attacker is prevented from accessing the C-RNTI, it will not be capable of forging data with correct physical ID. As we discussed in §2.3, device authentication on the control plane does not ensure data authenticity on the data plane, as the data-plane physical layer ID is transmitted over the air in cleartext and does not change with time.

A straightforward solution is to protect step ① by encrypting the C-RNTI delivery, so that the attacker fails to decrypt and learn C-RNTI. However, this solution cannot sufficiently protect this meta-info. Recall that C-RNTI stays constant after the assignment until a re-assignment. Even if the C-RNTI is unknown to the attacker during its initial assignment, it can infer the value by recording the signals, enumerating the possible C-RNTI values, and checking which one is the correct assignment for the victim UE. The possible value of C-RNTI is limited (<65k); although gNB might update the C-RNTI assignment, the attacker still can quickly infer the C-RNTI before it is changed (which can be minutes).

Consequently, a secure solution for C-RNTI protection requires constantly changing its value and ensures no reuse. This means that for every data precoded with C-RNTI, the device updates its C-RNTI. The update procedure needs to be secure; this guarantees that a C-RNTI inferred from the previous transmission will be invalidated for any future forgery. Hence, even a smart attacker who is aware of the scheme cannot break C-RNTI protection.

Impracticability of C-RNTI Protection Unfortunately, it is extremely difficult, if not impossible, to protect C-RNTI with per-delivery update while keeping its normal functionality. One solution approach is for each UE to update C-RNTI locally; however, 5G requires each device to own a distinct C-RNTI, so UE/gNB can use it to identify each connected device. A distributed solution cannot guarantee free-of-collision without centralized coordination [DAS20]. Another solution is a synchronized update scheme, where the gNB allocates distinct values to the connected devices for each delivery. However, C-RNTI assignment can fail, which forces the UE and gNB to wait for a retransmission. During this time period, the device cannot send or receive any data without a valid C-RNTI, since the old one could be already inferred by the attacker. This greatly limits the throughput for 5G.

5.1.2.2 Protecting DCI is a Valid Choice

If an attacker cannot get either scheduling or modulation, it cannot forge the data plane data or signaling messages. 5G has a strictly-regulated access network, where both information pieces are carried in DCI in PDCCH. That said, if DCI is kept secure in this downlink channel, the attacker will not be able to get the correct scheduling information or modulation without access to the encryption key.

DCI protection for meta-info security is feasible Encrypting DCI, which carries critical meta-info fields, is a sufficient solution to their security. Unlike C-RNTI which stays constant for multiple transmissions, gNB notifies DCI to UE for each data delivery (including transmission and retransmission) given 5G's strict access control. Hence, the encryption can be done without extra interchange between the gNB and device. The sender of DCI (aka gNB) can encrypt each DCI which prevents an attacker from eavesdropping on its value. The receiver can generate the same key for decryption. The solution thus requires consistent Keygen and additional encryption functionality at the MAC layer on both sides.

DCI does not demand integrity protection One key question is whether encryption on DCI suffices protection. *If DCI from gNB to device is forged by the attacker, can a smart attacker flip bits to alter its content?* We argue that, unlike data packets, integrity protection is not necessary for meta-info DCI. We first consider DCI containing uplink resource assignment. Even if the device could forge DCI and use its info to forge UL data packet or signaling, the data will be rejected with incorrect RB or modulation information that is known to the gNB. Similarly, authentic DL data will be not decoded properly if the corresponding DCI is manipulated by the attacker. If the attacker also forges data along with DCI forgery, it cannot learn the correct parameters to encode data without the capability to properly encrypt the forged DCI. Consequently, neither forging DCI for UL nor for DL results in any vulnerabilities that lead to data forgery attacks as discussed in the related works [RKH19, TDZ21]. The attack damage of sophisticated DCI manipulation is DoS at best, which can also be achieved by channel jamming. To achieve our security goal, DCI integrity protection is unnecessary.

DCI is much more lightweight than data packets We also show that, the overhead could be small for DCI protection due to its smaller size and lower frequency compared to that of the data packets, thus resulting in marginal overhead. This is especially important considering 5G's extremely high throughput. The reason is rooted in 5G's data aggregation approach. When multiple data packets or signaling messages wait in the buffer for resource assignment, gNB will schedule large data grants and attempt to clear the data in buffer in consecutive time slots. Consequently, the large chunk of data will be aggregated by RLC protocol into a single MAC block, in accordance with the assigned resource. The aggregated MAC block only requires one DCI. Meanwhile, each DCI is small, generally with a size of sub-10B regardless of type. Considering 5G's large bandwidth (up to 99MHz in sub-6GHz bands and 396MHz with mmWave) and enhanced encoding scheme (256QAM), 1 short DCI meta-info can carry a scheduling info for a MAC block that is concatenated from tens of PDCP packets of 1,500 Bytes. We confirm this by collecting logs from a 5G Pixel6 phone using MobileInsight 5.0 [LPY16, LPZ21] and analyzing the throughput between data packets (PDCP) and DCI. We compare them under different scenarios of uplink/downlink throughput. As shown in Figure 5.1, the DCI throughput is 1-2 magnitude

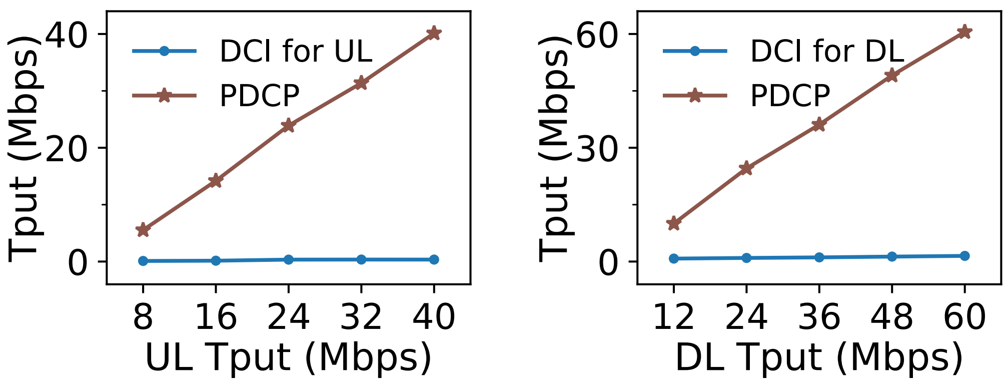


Figure 5.1: The throughput between DCI and PDCP data packets for uplink and downlink.

smaller than PDCP packets. Therefore, encrypting DCI meta-info is expected to incur much lower overhead compared to conventional per-message, data-centric protection.

5.2 DCI Protection: Our Approach and Challenges

5.2.1 Solution Overview

We design DP^2, a DCI protection scheme for data-plane forgery attacks. As a relatively unexplored solution idea, DCI protection has the potential of achieving both low overhead and high security given the features discussed in §5.1.2.1. DP^2 satisfies the following requirements to sufficiently protect against data-plane attacks:

1) Secure. DP^2 is secure so that the scheduling information cannot be inferred by the attacker. Encryption can prevent the attacker from learning the critical meta-info for data forgery. It is thus sufficient to protect either data-plane packets or signaling forgery attacks against 5G.

2) Lightweight. DP^2 only requires moderate overhead in comparison to integrity protecting and encrypting all data packets and signaling. This is due to the low frequency of DCI information compared to data packets and data-plane signaling. Besides, integrity protection is not necessary.

5.2.2 Challenge: Low Overhead and Latency Requirement

Challenges for DCI protection Although it is promising to protect DCI, our solution must further overcome the following technical challenges in designing viable DCI protection.

1) Is it possible to encrypt DCI with secure keystream? The encryption must use a variant keystream instead of a static key. Otherwise, an attacker can potentially infer the key from previous DCI and use it to decrypt future DCI encryptions. However, generating keystream requires consistent parameters for sender and receiver to produce a consistent key and these parameters must be different for each data delivery to avoid key reuse. Unlike data packets, DCI cannot be uniquely identified by PDCP meta-info sequence number. `DP`^{2} aim to generate such keys without adding extra fields in DCI or frequent cross-layer operations with higher layers.

2) Can we reduce the processing overhead to meet deadline? Although the frequency of DCI is considerably smaller, generating key and encryption DCI can still be the bottleneck. `DP`^{2} has to guarantee that it can be readily sent over-the-air before the time/frequency blocks (RB) scheduled in the DCI are expired. This is especially challenging for the gNB that is serving a large number of user devices. For a practical solution, `DP`^{2} further reduces the processing overhead of encryption DCI.

5.3 `DP`^{2}: Time-Frequency-Based Encryption on DCI

5.3.1 Leveraging Time and Frequency for Keystream Generation

Address Challenge 1: Time- and Frequency-Based Keystream Generation A keystream generation algorithm requires input parameter(s) to satisfy two requirements. First, the parameter needs to be available and synchronized for both sender and receiver, so that both sides can correctly encrypt and decrypt the data. Second, the parameters should not reoccur. Otherwise, the attacker can infer a generated key to reuse it later. In 5G, data packet and signaling message use distinct, increasing, and cleartext sequence numbers for

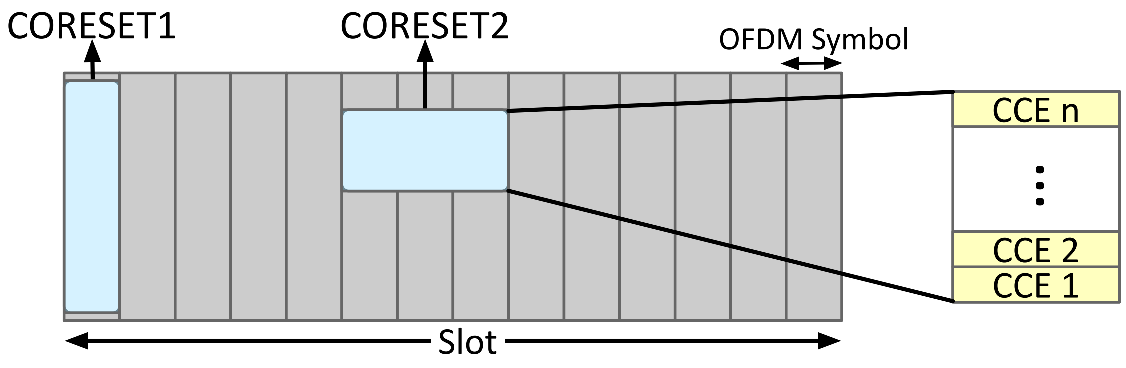


Figure 5.2: CORESET and CCE in a single time slot.

PDCP to generate keystream. Unlike data packets, MAC layer does not have such numbers for DCI information.

We leverage the fact that each DCI information is *strongly time- and frequency-correlated.* In the time domain, 5G splits the time discretely into frames of 10ms duration. A frame consists of 10 subframes of 1ms duration. A single subframe can have 2^{μ} slots depending on cell configurations, where $\mu \leq 4$ and varies based on cell configuration.

Each DCI is scheduled and sent in a specific time slot. The time slot can be indexed with the frame number and the slot number within a frame. The time clocks (frame and slot numbers) between UE and gNB are fully synchronized, with the propagation offset by timing advance during the random access procedure. Since the timers always tick forward, an attacker who infers a previous DCI key cannot use it for breaking future DCI transmission.

In the frequency domain, 5G splits the channel into multiple physical resources called Control Resource Sets (CORESET, detailed in [3GP21g]). A UE can be assigned at most 15 CORESETs in RRC message [3GP20e]. Each CORESET is divided into Control Channel Element (CCE), each with a unique index as shown in Figure 5.2. DCI can be transmitted in one or multiple CCEs[1]. On the device side, the UE takes a blind decoding approach. In each time slot, it enumerates all possible CCEs within CORESETS and check whether the DCI is for itself[2]. Therefore, the CORESET index and CCE index can identify a DCI in a

[1]Determined by the aggregation level config.

[2]The set of possible CCE index is called search space. It can be calculated with configs in broadcast &

time slot, which are known to both UE and gNB.

Consequently, each DCI can be uniquely identified with a time-frequency index. It satisfies both requirements of parameters for keystream derivation. 5G provides proven-secure, efficient keystream generation algorithms for data packets, called NEA [3GP22b]. We reuse the algorithm to generate keystream for $\mathtt{DP}^2$. NEA takes the following arguments. A 128-bit session key, a 32-bit count, a 5-bit bearer ID, a 1-bit direction, and the length of keystream are required. In $\mathtt{DP}^2$, we reuse the algorithm by replacing the arguments. The 128-bit session key is generated from K_{gNB} after mutual authentication. The 32-bit count is replaced with a 10-bit frame number (FN), an 8-bit slot number in the frame, a 2-bit CORESET ID, and a 12-bit CCE index. The bearer ID is set to all 0s and the direction bit is set to 1. The length is the size of the DCI.

5.3.2 Early Inference for Generating Keys

Address Challenge 2: Early Key Generation for DCI Encryption We could further accelerate the processing of stream key generation if we know the time, frequency, and type of a DCI in advance. By inferring the scheduling result in advance, the DCI protection stream cipher key can be pre-generated, which makes sure the processing can be finished without delaying its transmission.

In 5G MAC scheduling, early inference on scheduling information is indeed feasible, for both uplink and downlink. For downlink data transfer, as introduced in [BBM19], mobile network scheduling always empties a UE's DL buffer before moving to another one. For uplink data transfer, the gNB attempts to give sufficient grants to a user based on its BSR value [TZL21]. Therefore, for both UL and DL, multiple DCIs for consecutive time slots can be determined once after scheduling. The gNB can pre-generate keystream for all of them after scheduling, instead of waiting until the delivery of the DCI.

In addition, for lower overhead, we suggest gNB enables the new multi-PUSCH scheduling

RRC messages and C-RNTI. To find its DCI, a UE blindly tries all possible combinations of parameters in the search space and decodes them to check CRC with the assigned C-RNTI.

technique for the DCI. The recent standards [3GP20a] have supported time domain resource allocation for multi-PUSCH with a single DCI. The RRC layer sets up a specific table. In the table, the time domain resource assignment field in DCI format 0_1 (non-fallback version for PUSCH) can be mapped to up to 8 consecutive slots. gNB can use such mapping to assign resources with one DCI. It is compatible with gNB's scheduling policy to quickly clear a user's buffer. This could reduce the frequency of the DCI and thus the encryption overhead.

5.3.3 Other Solution Components

Other component: Secure key update The frame, slot number, and CORESET/CCE combination will reset after the frame number reaches the maximum value. Therefore, whenever this happens, the session key needs to be updated to prevent key reuse. We generate the new key by running key derivation function on the previous key. The key update procedure happens around every 10s. The low frequency of the update incurs negligible overhead.

Incremental deployment of DP^2 A gNB with DP^2 can serve both UE with DP^2 and legacy UE for incremental deployment. To achieve so, a DP^2-compatible UE can indicate its capability during RRC Connection setup. gNB confirms with UE and enables DCI protection. Otherwise, the data transmission falls back to the legacy mode.

5.3.4 Complete Procedure of DP^2

With the idea of time/frequency-based protection and pre-key generation, the complete procedure of DP^2 is shown in Figure 5.3.

Set-up Phase In the current 5G key hierarchy, a session key K_{gNB} is established during RRC connection setup. Both UE and gNB use this key to derive keystream for data-plane encryption and integrity protection. The derivation uses HMAC-SHA-256 in current standards [3GP21b, 3GP22b] as the key derivation function (KDF). It can securely generate a new key given an old key and a unique string S. We further generate a session key K_{DCI}

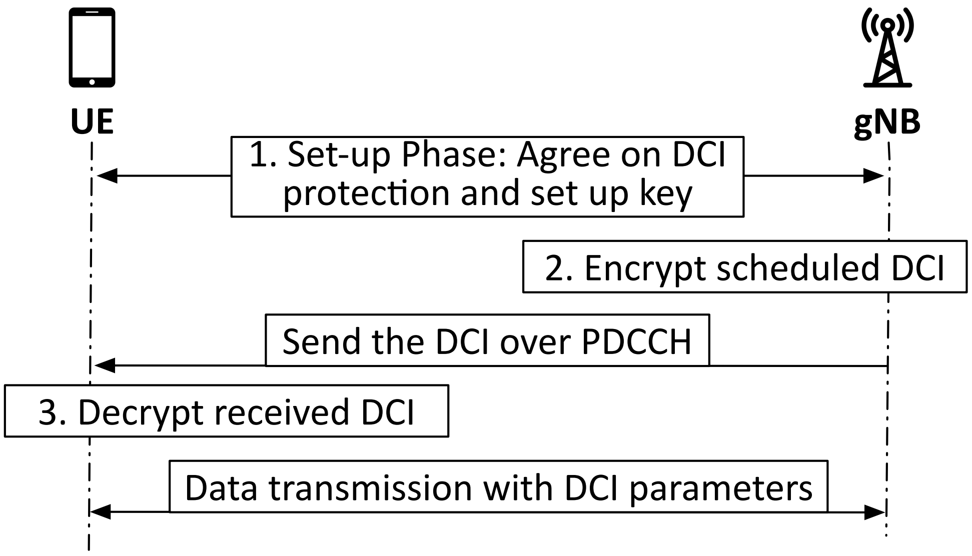

UE
gNB
1. Set-up Phase: Agree on DCI protection and set up key
2. Encrypt scheduled DCI
Send the DCI over PDCCH
3. Decrypt received DCI
Data transmission with DCI parameters

Figure 5.3: The procedure of DP^2.

with K_{gNB}.

$$S = FC_{\texttt{DP}^2} \tag{5.1}$$

$$K_{DCI} = KDF(K_{gNB}, S) \tag{5.2}$$

where $FC_{\texttt{DP}^2}$ is chosen from the reserved range of FC (i.e., unused for generating other keys). Since K_{gNB} is shared between the device and gNB, K_{DCI} is consistent on both sides.

Encryption at gNB The encryption is done by gNB side MAC. As introduced in §5.3.1, the keystream is derived from session key K_{DCI}, DCI length, frame/time slot numbers, and CORESET/CCE index. It is generated with the NEA KEYGEN function with the following formula.

$$COUNT = FrameNumber|TimeSlot|CORESET|CCE \tag{5.3}$$

$$BEARER = 0, DIREC = 1 \tag{5.4}$$

$$k = \text{KEYGEN}(K_{DCI}, COUNT, BEARER, DIREC, LEN) \tag{5.5}$$

$$DCI' = DCI \oplus k \tag{5.6}$$

Decryption at UE After the UE receives the DCI intended for it by checking the C-RNTI (same as the vanilla 5G), it generates the key k similar to the gNB side using Formula 5.3-5.5 and decodes the DCI with the following formula.

$$DCI = DCI' \oplus k \tag{5.7}$$

The key k will be identical to the encryption key, as all arguments are synchronized between two sides.

Secure key update The session key is refreshed when timers reset or handover occurs. A new K_{DCI} is derived from the old one K'_{DCI} as follows:

$$K_{DCI} = KDF(K'_{DCI}, S) \tag{5.8}$$

Since both sides have synchronized timers, they will update the keys at the same time to ensure a consistent key for encryption and decryption. When a handover occurs, the old gNB shall deliver the updated K_{DCI} to the new serving gNB.

5.4 Security Analysis for DP^2

In this section, we show that the attacker cannot forge data-plane packet or signaling correctly, with DP^2 applied.

A basic attacker cannot decrypt DCI DP^2 prevents the attacker from eavesdropping on the DCI. The DCI messages are encrypted by DP^2 with the same encryption algorithm used by data-plane packets. They thus have the same security level as data packet protection, so the attacker cannot infer the data without access to the keys [3GP22b]. Note that, even if the adversary does not follow standardized procedure, it still needs to send data with correct modulation and resource blocks. Therefore, it cannot bypass DCI meta-info to forge any message.

An advanced attacker cannot decode or forge a DCI by guessing We consider an advanced attacker who is aware of DP^2. Although it is not capable of decrypting the DCI content, the attacker can still attempt to launch the forgery attack. First, it is possible for it to infer the DCI fields used for data transmission by eavesdropping on the data delivery.

Table 5.2: Value space for each *necessary* field in DCI.

Field in DCI	Value Space	UL/DL
UL/DL identifier	2^1	Both
Frequency domain assignment	2^7 - 2^{16}	Both
Time domain assignment	2^4	Both
MCS	2^5	Both
HARQ Process	2^4	Both
Frequency Hopping Flag	2^1	UL
VRB-to-PRB mapping	2^1	DL
New data indicator	2^1	Both
Redundancy version	2^2	Both

However, the inferred DCI cannot be used for future transmission, as it could be different for each scheduled resource[3].

Second, the advanced attacker can choose to randomly guess meta-info and forge data. We show that the attacker only has a negligible rate of success in guessing the scheduling information. We list the critical fields for transmission in DCI and the possible values for each one in Table 5.2. To correctly forge the transmission, the attack needs to correctly guess every field, which has a probability of at most $1/2^{26}$. Hence, even if the attacker is aware of the DP^2 approach, it cannot launch any data-plane forgery with a reasonable success rate by guessing the correct RB and modulation for the forgery. In addition to the basic DCI format (called fallback), 5G introduces new DCI formats to support advanced features. For instance, a single DCI can indicate different MCS values for transmission in multiple time slots. In this case, an attacker will have even lower probability of guessing all fields.

DP^2 can help protect against DC attacks In the current 5G non-standalone (NSA) deployment stage, a device can connect to 4G and 5G cells simultaneously. This is called

[3]Hence, we suggest a DP^2-compatible gNB to schedule DCIs with variant MCS and RB assignment even for consecutive transmissions.

5G dual connectivity (DC). A scheduler decides from which cell a packet is sent. Therefore, new 5G security mechanisms, such as optional data packet integrity protection, do not apply to the data packets that go from the 4G cell(s). To prevent forgery attack that exposes 4G-specific vulnerabilities, a direct solution is to completely disable the use of 4G cell. However, in the current NSA deployment mode, control plane messages are carried in the main 4G eNB. Shutting down DC is thus not an option until the full rollout of 5G standalone (SA) [3GP22c].

DP^2 can also be deployed for a 4G cell in 5G DC mode, as 4G DCI carries similar meta-info with minor differences in several fields (e.g., CORESET$\rightarrow$Control Region). By protecting the meta-info, the attacker cannot target data or signaling message sent in the 4G cell.

Attacks not in the scope DP^2 aims to protect forgery attacks even for cleartext messages, but not a passive attacker that eavesdrops on (encrypted) data packets or data-plane signaling messages. Therefore, the sensitive information, such as data-plane packets, is still suggested to be encrypted as already in 5G/4G design. DP^2 can help mitigate forgery attacks in radio network, but not if a man-in-the-middle exists in the core network or Internet. Hence, end-to-end security protection is still preferable to complement our solution.

CHAPTER 6

Device-side, Rootless Attack Detection and Mitigation

In this chapter, we explore the other direction for countermeasure. Instead of changing standard, we aim at detecting forgery attacks and launching device-side mitigation. This will provide support for legacy devices immediately. We overview the solution scope and approach in §6.1. With the solution idea, §6.2 introduces how our solution `CellDAM` spots the potential attacker by inspecting data-plane signaling. The signal could be learned without root, thus requiring no extra privilege. Finally, §6.3 introduces a device-side mitigation approach and §6.4 analyzes the security of our solution.

6.1 Overview of the Solution Approach

6.1.1 Scope of the Detection

This chapter focuses exclusively on 5G *data-plane attacks* that actively forge data-plane packets or signaling messages. To the best of our knowledge, all previous works on 5G/4G attack detection focus on detecting undesired behavior on the control plane [HCM18,HEK19]. Although launching certain data-plane attacks relies on successful control-plane manipulation [EAW21], recent studies [TDZ21, YBS19] show that an advanced adversary can bypass such steps and directly attack the data plane. We will present a few example attacks. This work in the dissertation thus studies the viability of *directly* defending against such data-plane attacks. Some prior studies focus on detecting FBS on physical layer with traits such as signal signature; however, an FBS is not necessary for data forgery as shown in recent works [YBS19, TDZ21]. We do not consider *passive attacks*, such as fingerprinting attacks using the data-plane traffic [KRH19].

We aim to *detect and mitigate* such attacks *without standard changes*. Fixing the vulnerabilities with standard updates can eliminate some threats. However, it has two limitations that can be addressed by our complementary proposal. First, it requires software patches or hardware upgrades on both the infrastructure and the device, thus incurring high deployment and operation costs. Second, it often takes months or even years for full rollout [Vir12]. Some users may still suffer from attacks during the transition phase. Our approach offers an immediate remedy to the data-plane attacks.

We will design our solution *on the device side*. Unlike network-centric protection approaches [Vir12, HRO16b, Pos20, RKH16], our scheme works entirely on the device side without expensive infrastructure updates or hardware upgrades. Moreover, certain attacks can only be observed by the device, e.g., a forged message that is only decoded by the victim device [TDZ21]. This also makes a case for our device-based solution.

6.1.2 Threat Model for the Detection

The adversary seeks to incur various damages on the target victim device by *forging either user data packets or data-plane signaling messages*. We do not consider control-plane threats (such as an FBS-based IMSI-catcher). On the data plane, we do not limit the message that can be forged by the adversary. The forged message can be from a known attack or from an unreported attack. The user device runs its Internet/mobile apps, which involve bidirectional communications and data transfer with the base station.

The attacker could have the following capability: 1) The adversary connects to the same cell as the victim device. This is feasible with fingerprinting attacks [KRH19], social engineering to locate the target user [LZG14], or simply attacking a random nearby device; 2) The adversary may eavesdrop on the physical channels and transmit data on them; 3) The adversary can exploit fake base station (FBS) [RKH19, RKH20b] or overshadowing attacks [YBS19]. Although FBS is mitigated in 5G [3GP20b], it is still possible to launch certain variants of FBS, such as relay FBS as man-in-the-middle.

Note that, we do *not* consider an insider attack where the adversary can steal security keys stored in SIM/networks or even compromise a 5G base station (i.e., gNB). Detecting such attacks is beyond the scope of this dissertation.

6.1.3 Our Solution Approach and Challenges

Guideline 1: Verify what is right Rather than target specific attacks, we design and implement approaches that verify whether the runtime, data-plane operations follow the correct procedures stipulated by the 5G standards, and treat any undesired behaviors as suspicious. Our approach ensures the detection of a category of attacks that yield improper data-plane operations. Even if an attack has not been reported yet, it can be detected by our approach as long as it triggers undesired behavior.

However, it is nontrivial to monitor the 5G data plane. The 5G data plane is expected to reach gigabits per second in packet delivery. The traffic volume of data packets is thus several orders of magnitude higher than the control-plane signaling messages. We aim to design a lightweight security solution that can work under heavy data traffic.

Guideline 2: No extra privilege Checking data-plane attacks is impossible without accessing data-plane messages. Unfortunately, a common device-side, user-space application cannot access such cellular-specific information, unless extra privilege is granted. Mobile OS only provides control-plane basic info [APIb], such as cell ID or connection state. With root privilege granted, tools like MobileInsight [LPY16] or QXDM [Qua] can access Diag port and decode messages for detailed cellular logs. However, such extra privilege (e.g., root access) exposes new vulnerabilities [Lab17], thus degrading the system security strength. We aim to devise detection and mitigation schemes that *do not introduce new security loopholes.* Another option is to directly implement the solution inside the SIM or the firmware. However, a normal user or even developer will not have access unless collaborating with device vendors or mobile carriers.

In summary, our solution must address three issues:

Issue (1): How to detect data-plane undesired behavior with acceptable overhead?

Issue (2): How to detect undesired behavior in the user space without root privilege?

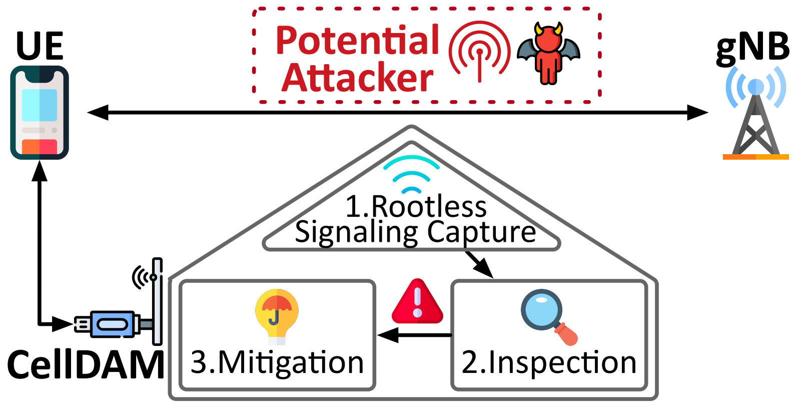


Figure 6.1: Envisioned procedure of `CellDAM`.

Issue (3): How to ensure user-space, rootless mitigation?

6.1.4 `CellDAM`: Rootless, Userspace Detection and Mitigation

We design `CellDAM`, a 5G data-plane inspection scheme without root privilege, as illustrated in Figure 6.1. It addresses all three aforementioned issues. `CellDAM` first captures all signaling messages from/to the protected UE in a separated node called `SecHub`. This bypasses the requirement of in-device extra privilege. `CellDAM` then inspects the data-plane signaling messages with lightweight, undesired behavior detection. Finally, `CellDAM` uses `SecHub` to mitigate the attack damage via cell switching. The detailed components are shown as follows.

Data-Plane inspection (§6.2.2) We first show that, inspecting data-plane signaling offers an effective way to detect data-plane attacks. The design effectively addresses Issue 1, as data-plane signaling is of smaller volume compared with the actual packets. We further propose a *cross-layer*, state-dependent model checking for attack detection. If the device spots an incoming signaling message that is undesired in the current state, it detects a potential attack.

Rootless signaling acquisition (§6.2.3) To address the issue of rootless detection, `CellDAM` incorporates a separate, companion node named `SecHub` for the protected device. The goal is to share a consistent view over the baseband. The node can capture over-the-air signaling messages by inferring the device ID and traffic pattern. The tracking can be done continuously even after a handover. It needs no extra privilege on the protected device, thus addressing Issue 2.

Device-side reaction (§6.3) Once the suspicious operations trigger an anomaly, we further activate the mitigation module that switches to other, potentially attack-free 5G channels. `CellDAM` can perform its channel switching without root access, thus addressing Issue 3. This is to leverage the standardized 5G/4G procedure and dense cell deployment. It forces a handover to a new cell using the designed signals from `SecHub`, without affecting other user devices.

6.2 Inspecting Data Plane for Detection

6.2.1 Monitor Data-Plane Signaling for Attack Detection

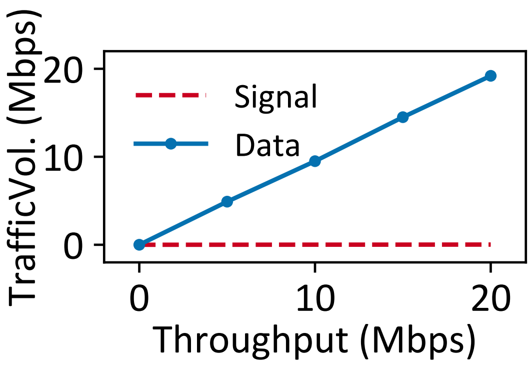
Data-plane signaling for attack detection We show that, monitoring data-plane signaling offers an effective method for detecting data-plane forgery, including both packet delivery and data signaling attacks. When a data-plane attack is underway, the data-plane signaling messages will deviate from their normal behavior. This applies to both data signaling forgery and packet manipulation attacks.

We first consider attacks of data packet manipulation. We start with downlink packet forgery. We use the examples in `CDS` and `FANE` to illustrate our finding. Contrary to conventional wisdom, forging a data-plane user packet will potentially incur undesired signaling messages. Consider both methodologies to forge downlink data. For the scheme with relay FBS, an attacker cannot directly learn the critical data-plane configurations, which are transmitted over the *encrypted* RRC messages. This has been discussed in [YBS19]. Consequently, the forgery will be sent in a normally impossible context, e.g., in the resource blocks when the device is in DRX (Discontinuous Reception) OFF (i.e., when the device is in sleep mode). For the shadow attack without FBS, it needs to forge DCI and data packets. The next DCI from the attacker could reach the device before the acknowledgment of the forged data, thus incurring an undesired behavior.

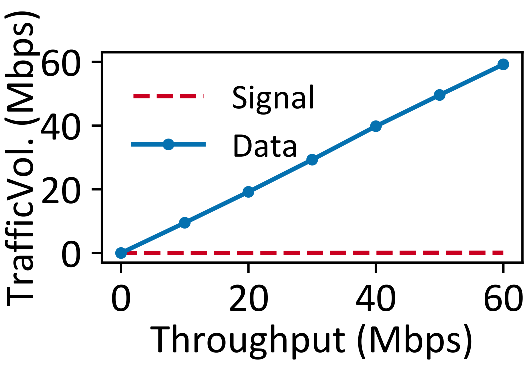
We next consider uplink data forgery. Note that uplink forgery alone might not be detectable from the device. However, a practical data-plane attack would forge both uplink and downlink packets to be effective. We also observe that almost all mobile apps involve both uplink and downlink data exchanges. Moreover, current 5G applications are usually built with TLS/HTTPS or DTLS (i.e., the secured versions of TCP/HTTPS and UDP). Consequently, a natural manipulation target is the IP header, where the attacker launches a redirection attack [RKH19]. We have 2 cases. If TCP is used, then the traffic is two-way. If UDP is used, most applications will also require downlink responses. The uplink attack will also manipulate the downlink packet, which is generated as the response to the uplink packet [RKH19]. Consequently, although an uplink forgery is not directly detected, attacks on data packets can still be detected at the device from its associated downlink packet forgery. In summary, inspecting data-plane signaling messages can effectively detect almost all attacks against data-plane operations.

We next consider data-plane signaling attacks. Since the forged signaling is unexpected, it will be received in the wrong/unexpected context. Consider the example attack with RLC Control. Suppose the attacker forges an RLC NACK to force retransmission on UL RLC packet ID k, which has already been received by gNB. The RLC will forward this packet to the MAC layer for false retransmission. However, if we cross-check the MAC and RLC protocols, the authentic RLC ACK is received before the successful delivery of the retransmission. This could not happen for normal operations. It thus triggers an undesired behavior on the device side; this is not currently checked by 5G implementations due to its cross-layer nature.

Lightweight data-plane signaling inspection Compared with data-plane packets, inspecting data-plane signaling is of much lower overhead. First, the size of each signaling message is much smaller than the actual data packet. The signaling messages (such as DCI, RLC Control) are at most several bytes long. Some PHY messages are merely 1-bit indicators. Second, 5G data delivery will transmit multiple IP data packets in a single data block (aggregated and segmented by the 5G RLC protocol). Therefore, for signaling messages that facilitate data delivery (e.g., DCI), only one such message is needed for the large block.



(a) Under Testbed (5G)



(b) Under Commercial (LTE)

Figure 6.2: The comparison of traffic volume per second for data packets vs. data-plane signaling messages.

We validate it by comparing the traffic volume between data packets and data-plane signaling under different traffic. We show results from operation traces in a commercial network and in our SDR-based testbed. Our testbed runs standard-compliant srsRAN 5G [SRS] and the details will be shown in §8.5. Since our testbed does not support features (including MIMO, carrier aggregation) for higher throughput, we also collect traces from commercial operators. As the current open-source 5G decoding tools (e.g., MobileInsight 5.0 [LPY16, LPZ21]) have not supported 5G data plane, we collect and analyze 4G data plane as a reference, considering that data-plane signaling design remains largely unchanged in the current 5G NSA [3GP20a, 3GP21a][1]. As shown in Figure 6.2, the processing of data-plane signaling is 1∼2 orders of magnitude lower than that of data packets.

Therefore, detecting attacks with data-plane signaling is of much smaller overhead than monitoring the entire data-plane packets. Prior work [FW19] has already shown an SDR-based system is capable of monitoring DCI messages for *all devices in a cell.* It is thus feasible for `CellDAM` to capture all data-plane signaling *for a single device* while performing inspections in real-time (detailed in §8.5). This is important considering 5G's high data rate.

[1]One major difference is that 5G cancels PHICH which is used for uplink data retransmission. Therefore, we do not include it for calculation.

6.2.2 Cross-Layer State-Dependent Model Validation

We now present how `CellDAM` inspects the data-plane signaling messages to detect undesired behavior. We follow our guideline of "verifying what is right" to model the device-side protocols and detect any undesired behavior. To this end, we devise validation schemes using the standardized data-plane delivery procedure to check each incoming/outgoing data-plane signaling message. These validation checks are *state-dependent.* For example, an RLC NACK for a packet before its MAC delivery and acknowledgement is an undesired behavior; it is not if already being handled by MAC.

To model and track the state, we use Deterministic Finite Automata (DFA), a common technique for state tracking in attack detection [HCM18, EAW21]. We build multiple DFAs for uplink and downlink communications, whose inputs for transition are *data-plane signaling messages* or their derived events. If the next message m passes all validation checks, the DFA moves to a new state; Otherwise, a potential attack is detected and we initiate the mitigation procedure.

Formally, we maintain n deterministic finite state machines $M = \{M_1, M_2, ..., M_n\}$. For each DFA $M_i, i = 1, 2, ..., n$, we denote it as a 5-tuple $(S_i, S_i^0, S_i^1, \Sigma, T_i)$, where S_i is a finite set of states with $S_i^0 \in S_i$ being the initial state and $S_i^1 \subseteq S_i$ being the accepted states, Σ is a finite set of input messages[2], $T_i : S_i \times \Sigma \to S_i$ is a transition function mapping the pairs of a state and a received message to the next state.

We build validation checks $V = \{V_1, V_2, ..., V_k\}$. Each V_i is associated with a DFA M_j, a state $S \in S_j$, and a message $m \in \Sigma_j$. Every time the DFA M_j with state S inspects a new message m, `CellDAM` runs the corresponding check(s). They map the signaling message m to 0 (fail) or 1 (succeed) given the current context, which is derived from past records or other DFAs. A potential attack is identified if one of the validation checks fails; Otherwise, M_j accepts message m and updates its state accordingly.

To construct the DFAs, we further address two design issues. First, the data plane does

[2]To make the problem tractable, we only consider the discussed data-plane signaling messages. We prioritize soundness over completeness.

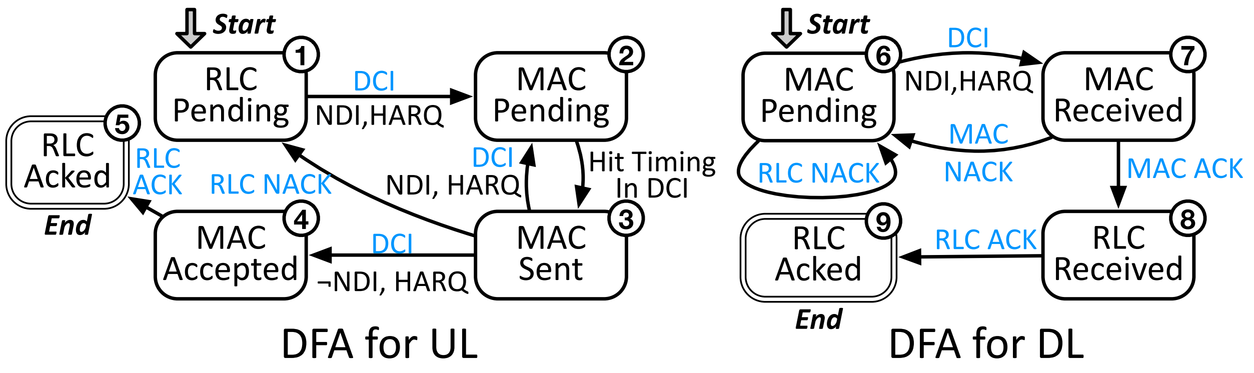


Figure 6.3: DFA for tracking states with data-plane signaling. See Appendix B for an extended version.

not have standardized DFA provided by the 5G specification. This differs from the control-plane protocols, which have standardized DFA based on control-plane signaling [3GP20e, 3GP22a]. Therefore, we leverage the *mandated, standardized data delivery procedure* to construct DFAs. We study 3GPP standards across all 5G-specific data-plane sublayers and manually create DFA for attack detection on the data plane. Second, maintaining single-layer DFAs will not work. Consider our example detection. The attack that forces RLC retransmissions cannot be detected if we curate the model for a single protocol only. The detection needs cross-layer considerations between MAC and RLC.

We utilize the data delivery procedure to construct the DFA as illustrated in Figure 6.3. We form cross-layer DFA for each RLC data packet with its necessary state transition at the MAC layer. We do not include PDCP, as it does not maintain state or buffer packets. For uplink, data transmission follows a scheduling-based feedback loop. The device first sends requests (Buffer Status Reports or BSR) to ask for resource grants until the packet is delivered by MAC. The MAC fast retransmission is notified by a new DCI with the same HARQ ID and NDI. The packet is considered delivered when the RLC ACK is received. For downlink, the data transmission follows the same procedure but without the request-grant loop, as the gNB initiates the transmission. The device sends the MAC feedback of ACK/NACK in PUCCH.

Based on the state, we perform a few validations on each incoming/outgoing signaling

Table 6.1: Validation checks performed by `CellDAM` based on state and message.

Check	State	Message	Validation Details
c_1	All	Any Message	The data-plane signaling shall be in the accepted list for each state. (Appendix B)
c_2	s_3, s_6	RLC NACK	It shall not be received after an RLC ACK with higher sequence number.
c_3	s_4, s_8	RLC ACK	An RLC ACK shall not be received before the packet is acknowledged in MAC.
c_4	s_7	MAC ACK/NACK	The ACK/NACK in PUCCH should be delivered at correct timing after DCI; If not, this is an indicator that the previous grant/data is received during DRX OFF.
c_5	s_1	DCI for UL Grant	There should not be large "free" grant when no request is sent or no data in buffer.

message. We show the list of validations in Table 6.1. First, all states will have a list of accepted messages. `CellDAM` checks whether the next message falls into the list. Any undesired message will be detected as a potential attack. The detailed list is shown in Appendix B. For c_2 to c_3, we are checking whether the RLC operations are consistent with the MAC layer for both uplink and downlink. For example, upon receiving RLC NACK, we check whether an early RLC ACK that has already acknowledged a packet is received. For c_4, we use the indicator of no ACK/NACK to detect a possible forged message received in DRX OFF. For c_5, `CellDAM` detects abnormal grants from gNB without any prior request. Note that a gNB can freely grant the device with some small grants, in the presence of uplink data. However, they are usually 100-200 Bytes long. Any larger grant incurs potentially a waste of resources. Therefore, it could be the outcome of a forged BSR or grant signal. If all checks pass, the DFAs are switched to the new state based on the signaling message.

We will show in §6.4 that, `CellDAM` can detect all known attacks adapted from prior work, as well as new unreported attacks; our model checking finds *any* incompatible or undesired forgery.

6.2.3 Capturing Signals for the Protected Device

As discussed in §6.1.3, we aim to apply our inspection solution without degrading its security strength. To address all privilege-related concerns, we design and deploy a separate node, `SecHub`, with wireless capability. The node is placed close to the protected device; it passively receives and decodes the data-plane signaling over the air. It can also communicate with the user-space security manager application at the protected device. The device and `SecHub` connect each other either with wire (a gadget that is attached to the device via USB) or wirelessly (i.e., Bluetooth).

`SecHub` protects itself by addressing security concerns in two ways. First, `SecHub` is available to end users without exposing any unnecessary interfaces, such as Internet access or wireless control. A user or an application will be unable to add/change/delete the `SecHub` software. `SecHub` is solely used for `CellDAM` detection and mitigation purposes. A passcode is set to access its functionalities. Second, `SecHub` communicates securely with the protected device. The user installs an application in the protected device using the certificate that comes with the `SecHub`. The device thus mutually authenticates with `SecHub` and encrypts all traffic between them. Only nonsensitive information is exchanged over this channel. Consequently, an attacker will be unable to easily use or compromise `SecHub`, eavesdrop on its channel, or break its functions.

The non-in-device design of `SecHub` has raised new issues. Although 5G data-plane signaling messages are not encrypted, the node must have the capability to identify and decode such messages. We first design the rootless inference to acquire all configurations needed for signaling capture (§6.2.3.1). Moreover, supporting continuous operation is challenging because the configuration may change upon user mobility or gNB updates. We design and deploy a continuous tracking scheme in `SecHub` for `CellDAM` (§6.2.3.2).

6.2.3.1 Rootless Signaling Messages Capture

For rootless signaling capture, several device configurations are needed for `SecHub` to record all data-plane signaling messages: (1) The carrier frequencies for downlink and uplink; (2) The physical cell ID (PCI) that indicates the physical-layer identity of the cell; (3) The Cell Radio Network Temporary Identifier (C-RNTI) that distinguishes the target device from other devices connected to the current cell. With all the above configurations, `SecHub` can select the corresponding frequency, camp on the target cell, and capture data-plane signaling for the protected device over the air.

However, not all these configurations could be acquired from the protected device without root access. Android provides APIs for applications to obtain the current band and PCI [APIb]. In contrast, C-RNTI can only be extracted from the victim device with root privilege. Without C-RNTI, `SecHub` cannot recognize which traffic is for the protected device. The question thus becomes: *How to acquire C-RNTI of the target device without root access?*

We use a novel, collaborated traffic fingerprinting scheme to infer the C-RNTI of the target device. The idea is, we generate specific traffic patterns on the target device, with `SecHub` being aware of the pattern in advance. It can thus observe the channel and identify the target device's C-RNTI by analyzing low-layer signaling. Our approach takes three concrete steps.

Traffic Pattern Coordination First, `SecHub` randomly generates a traffic interval and sends it to the target device through the wired or wireless channel. This interval is used as the fingerprint for the target device. The traffic interval triggers a unique pattern for the data-plane signalings for fast inference. `SecHub` leverages this shared traffic pattern to recognize the target device in later analysis.

Trace Collection Second, the device creates traffic (e.g., small UDP packets) with the acquired interval. The traffic generation is performed by the application and does not require root privilege. gNB will assign grants for the device to deliver data. At the same time, `SecHub` monitors all the subcarriers in the target cell and tries to decode the C-RNTI from all grants with all possible positions in the band. This is necessary as grants do not always locate on the same subcarrier in the band (for reducing inter-cell interference). `SecHub` records the decoded C-RNTIs with corresponding time slots for inference.

C-RNTI Inference Finally, `SecHub` aggregates the grants for each C-RNTI decoded from the collected trace. `SecHub` first ranks the C-RNTIs by grant numbers and filters the top 10% C-RNTIs as the candidates. The time intervals between consecutive grants are calculated and compared with the negotiated interval for all the candidates. The grants for the fingerprinting traffic will show the same interval. Although there may be background traffic from the target device, the grants triggered by the fingerprinting traffic still show the periodic pattern and could be filtered from the device's grant traces. By ranking the ratio of matched intervals with the total interval number, the top C-RNTI will be selected as the target's C-RNTI. To ensure robustness, `SecHub` performs the procedures twice and checks if the inferred C-RNTI values match. If the candidates do not match, `SecHub` will perform the inference again until a candidate is selected.

Combining the frequency and PCI of the target device from the Android API and the C-RNTI inferred from the collaborated traffic fingerprinting, the `SecHub` could successfully camp on the cell and capture the downlink/uplink messages. More importantly, the entire procedure does not require root access at the device.

6.2.3.2 Continuous Tracking

`SecHub` needs to get configurations in real-time to provide continuous protection. However, upon user mobility, the configurations could be updated within an encrypted RRC signaling message after the device connects to a new gNB. It is also possible that gNB updates the C-RNTI for the device upon RRC state changes without user mobility. Tracking the up-to-date configurations for the target device is critical.

To address these issues, `CellDAM` first develops a security manager application on the target device to track the possible configuration changes due to mobility. The application leverages the existing API and does not require any root access. When the device moves to a new gNB, the combination of PCI and band of the target device will be updated. The application could track the updates with OS-API and notify the `SecHub` to start a new round of C-RNTI inference.

`CellDAM` further leverages the C-RNTI updating scheme in the cellular deployment to retain the same C-RNTI when the device stays on the same gNB. Our study on commercial devices and operators shows that, the current gNB updates RNTI when the device transits from the RRC-Idle state to the RRC-Connected state. The gNB recycles the C-RNTI from the idle devices and reuses them for other devices. To prevent state transition, we trigger a lightweight ping in the background inside the security manager application. The ping traffic only requires low traffic volume to keep the device in RRC-Connected. A device in the RRC-Connected state can still go to DRX OFF state periodically, and the power saving is little affected. Our experiments show that the 1s interval ping could keep the C-RNTI unchanged for a long time. We validate it on 216 cells from three major US carriers. In all tests, the C-RNTI remains unchanged for at least 30 minutes with our light background traffic. To tolerate unexpected updates, `SecHub` also triggers the C-RNTI inference in §6.2.3.1 every 10 minutes to validate that the current configuration is up-to-date. In summary, `CellDAM` enables sustained protection for the target device with continuous tracking.

6.3 Device-Centric Mitigation with Handover

As we have introduced in §6.1.1, the 5G standard updates can fundamentally protect the victim from forgery attacks, but they are time-consuming and require months or even years to be deployed in practice. Moreover, such solutions can be incompatible with legacy devices. Instead, we design device-centric mitigation to provide a quick remedy readily available for existing devices. We leverage the existing protocols, dense 5G cell deployment, and `SecHub` to help the victim dodge the attacker without root access.

6.3.1 "Quick Dodge" with Band Switching

We observe that, the attacker must camp on the cell that serves the victim device and forge messages in the current band to launch the attack, regardless of using an FBS or an attack node. Therefore, the victim could quickly escape from attacks by switching to another frequency band (i.e., a different 5G cell). However, current in-device band switching requires root privilege on the mobile devices [DLH20].

We design standard-compliant and rootless mitigation with a "quick dodge" scheme. We leverage the handover procedure in 5G standards to enable rootless band switching. The handover procedure migrates a device from one cell to another. The UE measures the signal quality by metrics of Reference Signals Received Power (RSRP) and Reference Signal Received Quality (RSRQ). When the experienced signal quality of the serving cell is worse than the thresholds configured by gNB, the UE sends reports with measurements of the source and neighbor cells. The serving cell makes the decision and starts the handover to the best target cell.

We follow the same design guideline as for the detector, and propose to induce channel switching from the `SecHub`. The idea is as follows. `SecHub` reduces the channel quality perceived by the victim device upon detecting attacks; this will subsequently trigger a measurement report and a handover. With the handover, the victim could escape from the serving cell under attacks. We next introduce the detailed design to enable rootless band switching.

6.3.2 Trigger Handover with `SecHub`

The victim will handover to another cell when the perceived channel quality is degraded. `SecHub` thus injects noises on the frequency band of the current serving cell to reduce RSRQ in the measurement. We select RSRQ rather than RSRP, because the latter is measured based on the reference signal power and is hard to be reduced by a third-party node. However, simply sending noises cannot work. First, we envision `SecHub` to be a small portable device with limited power on a par with HackRF [One] or USRP [Rad]. It will not possess enough transmit power to inject noises over the entire 5G band, thus unable to downgrade the measurement to trigger handover. Second, injecting noises blindly within the entire band affects other nearby users. This might incur unexpected performance degradation or channel switching at other devices and raise legal concerns.

Precise reference signal downgrade We design adaptive signal degradation to ensure low-overhead band switching. Instead of the entire channel, `SecHub` only copes with the reference signal in 5G. The device measures the reference signal to monitor the signal quality regardless of PHY techniques (e.g., MIMO, carrier aggregation, dual connectivity, etc). The reference signal only exists in specific subcarriers and time slots. `SecHub` calculates the positions of the reference signal based on the current physical band according to the 5G standards [3GP21g]. By morphing the reference signals only, `SecHub` downgrades the victim's signal measurements of the current frequency band without much overhead.

Targeted switching The solution should not affect other users. `SecHub` adaptively controls its power upon triggering the handover. Previous 5G measurements show that -20dB RSRQ is enough to trigger the handover in more than 98% of the cases [XZZ20]. `SecHub` adaptively derives the minimal power so that RSRQ drops to -20dB, thus triggering handover. The minimal power (P_m) is computed based on the current RSRP, RSRQ, and the number of Physical Resource Blocks (N). We detail the calculation of minimal power to trigger the targeted switching. The current RSRQ is derived by:

$$RSRQ = \frac{N \times RSRP}{RSSI}$$

To reduce the RSRQ to -20dB, the minimal power (P_m) needed by `SecHub` follows:

$$\frac{N \times RSRP}{RSSI + P_m} = -20dB = \frac{1}{100}$$

Thus, the P_m could be derived by:

$$P_m = N \times RSRP \times (100 - \frac{1}{RSRQ})$$

All needed information can be acquired from the victim by the OS API (e.g., Android [APIb]) without root privilege. Furthermore, the power density is inversely proportional to the square of the distance from the antenna [Cal]. Assume `SecHub` is located close to the device (<0.1m). The RSRQ drop at the 1m distance is smaller than 1dB, which could be neglected by other devices. Only the victim device perceives a notable RSRQ drop and triggers its handover.

6.4 Security Analysis

`CellDAM` works for all known attacks `CellDAM` can detect all known data-plane signaling and packet attacks listed in Table 6.2 (A1-A5). They are adapted from those reported in 4G or Cellular IoT. Each attack violates a certain context in packet delivery and fails to pass all checks. The details of attacks and how `CellDAM` detects each are provided in Appendix A. For A1, the authentic RLC ACK might arrive after the forgery and before the forced retransmission, where an undesired behavior is triggered. For A2, the attacker forces the device into the DRX OFF even in the presence of new data. The device thus receives a DCI during DRX OFF, detected by its lack of ACK/NACK. For A3, the device will observe unsolicited grant without pending UL data. Although `CellDAM` has no access to data buffer, it can check if the corresponding uplink transfer contains no user data. For A4, the DCI from the relay FBS can arrive in the DRX OFF state and be detected similar to A2. For A5, the forged DCI from the adversary can conflict with the DCI with the same HARQ ID.

New unreported attacks can also be detected `CellDAM` can also detect unreported attacks on the data plane. As we mentioned in §6.1.1, our solution seeks to *verify what is right*, any potential attack that breaks the standardized delivery procedure will be detected. We run the validation and DFA to uncover new feasible data-plane attacks. For each state, message, and validation tuple, we check if a forged message that fails the validation can be from the adversary to incur damages. Consequently, we uncover three new unreported data-plane attacks. Their damages include reliable transfer violation, data collision for DoS, and delayed data delivery. We list these attacks and how `CellDAM` can detect them in Table 6.2 (A6-A8) and elaborate on their attack conditions in Appendix A. Since they fail the check, all these new attacks will be detected by `CellDAM`.

`CellDAM` **does not expose additional security loopholes** `CellDAM` achieves our goal of attack detection and mitigation without incurring new vulnerabilities. The separate node `SecHub` does not need to access or change hardware or firmware on either the network or the device. It also incurs no extra transmission except for 1) the wired or wireless communication with the in-device app that is secured; 2) signals that can only be detected by the protected device when the mitigation starts. As we introduced in §6.2, an attacker cannot access or break the functionality of `SecHub` due to its security measures. It will be eventually built in a gadget that can directly connect to the device via USB-C. Meanwhile, the in-device app does not need root privilege. It only accepts encrypted instructions to transmit light traffic and detects handover (i.e., cell change) with OS APIs.

Table 6.2: List of known (A1–A5) and unreported (A6–A8) data-plane attacks and how they trigger undesired messages. See Appendix A for details of each attack.

#	Attack	Forged Message	New?	`CellDAM`	Undesired Behavior	Check
A1	Packet Delivery Loop	RLC Control NACK	Adapted from [TDZ21]	✓	The RLC ACK is received right after a NACK.	c_1, c_2
A2	Prolonged Packet Delivery	DRX Command	Adapted from [TDZ21]	✓	Message received with pending transmission; DCI during DRX OFF.	c_1, c_4
A3	Radio Resource Draining	Buffer Status Report	Adapted from [TDZ21]	✓	Grant is received when no data is pending.	c_5
A4	DL Data Manipulation w/ Relay FBS	Data packet	Adapted from [RKH19, RKH20b]	✓	The forged packet received during DRX OFF.	c_4
A5	DL Data Forgery w/ Retransmission	Data packet	Inspired by [TDZ21, YBS19]	✓	Forged DCI for forged data received in wrong context.	c_1
A6	Break Reliable Transfer	RLC Control ACK	Yes	✓	RLC packet is NACKed after being already ACKed.	c_3
A7	Data Collision	DCI UL Grant	Yes	✓	Data sent in unauthorized resource blocks will not be acknowledged.	c_5
A8	Delayed Transfer	MAC ACK	Yes	✓	The sender MAC falsely regards ACK and triggers RLC retransmission.	c_1

CHAPTER 7

Optimizing 5G/4G without Affecting Security

In the previous chapters, we show that better security will require compromising performance, as the security mechanisms will incur extra overhead. The reverse statement is often true in most cases. Solutions to optimizing access network [LPY16, LYP17, TLL18] often need cellular-level information, such as signaling, critical parameters, etc. Acquiring them from the application layer will need extra privileges, such as firmware access or root privilege. This way, tools like QXDM [Qua] or MobileInsight [LPY16] can function as expected for applications. The basic idea is shown in Figure 7.1.

In this chapter, we invalidate this common wisdom. Even without exposing extra privilege, an attacker can still optimize the application performance. The key idea is that, certain configurations can be inferred from the application layer with specifically-designed traffic pattern. With the inferred parameters, many lower-layer performance metrics (i.e., latency) can be inferred and diagnosed on the application layer. A solution can thus take advantage of this information for optimization.

In the rest of this chapter, we will use latency optimization as an example. We focus on optimizing access network latency for emerging, latency-sensitive applications (§7.1). We break down the latency components in §7.2. Afterwards, we present `LRP`, a rootless solution that can reduce the RAN latency without root (§7.3). It relies on a new inference scheme to learn the cellular-specific parameters (§7.4). The acquired cellular-level info is used for proper optimization to satisfy 5G/4G latency requirements. We discuss the effectiveness of the solution under different scenarios in §7.5.

We note that, all the latency analysis is done on 4G LTE because 1) latency analysis tools on 5G is not accessible; 2) current 5G NSA still uses a similar data delivery procedure

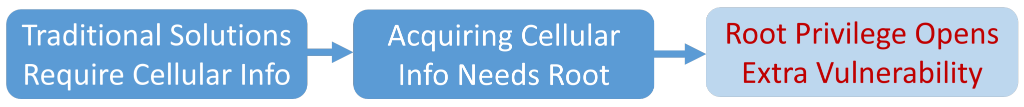

Traditional Solutions Require Cellular Info
Acquiring Cellular Info Needs Root
Root Privilege Opens Extra Vulnerability

Figure 7.1: Traditional solution approaches might expose new security threats.

as in 4G and thus produces the same latency components. The discussion and necessary adaptation for 5G is discussed in this chapter.

7.1 Optimizing Latency-Sensitive Mobile Applications

We exemplify some representative latency-sensitive applications over the mobile networks.

Mobile VR A mobile virtual reality (VR) app typically involves 3D scenes and associated graphical engines [Clo20, LCC15, BBC17, LHC19]. Standalone VR headsets such as Google Daydream [Day] render 3D scenes locally. However, due to limited computation resources and high power consumption on mobile devices, high-quality VR applications typically need the edge/cloud servers to offload the rendering task [SGJ19]. In this client-server scheme, the mobile headsets or pads provide sensory/control data, while the server renders the 3D scene in the form of graphical frames. The server coordinates multiple devices, renders the VR graphical frames based on the device's input, and constructs the appropriate 3D scene for each given device.

• *Showcase VR prototype:* Following the above paradigm, we have built an example VR game with Unity 3D engine [Rep] on Android phones to study LTE latency. It has three modules: the controller at the device, the camera controller at the server, and the streaming component. The Android controller app acquires the device rotation data from the gyroscope sensor to control the in-game camera rotation. The GPS location is fed into the VR game so that the virtual character moves with the player's location updates. Upon receiving the player's sensory data, the camera controller at the server processes them and makes corresponding position and rotation movements for the virtual camera. We implement the streaming module with open-sourced libraries Unity Render Streaming [Str] and WebRTC for Unity [Uni]. With the streaming module, the camera view is rendered and streamed back in 60FPS to the player with WebRTC. Players open the camera stream with the Web browser on the phone to get the real-time camera view.

Mobile sensing Smartphones today are equipped with multiple sensors: accelerometer, gyroscope, camera, to name a few. Many mobile sensing apps collect sensory data and upload them at runtime to the cloud for processing. For example, a localization app sends the GPS data to the cloud for realtime navigation. All such sensing apps are latency sensitive.

Mobile gaming In multi-player mobile games, the device acts as a controller that collects user motion, while the remote server processes the game logic. The server further provides proper synchronization and coordination among players. Moreover, pure cloud-based gaming (with rendering being processed in the cloud) is also trendy [LHC19]. It is a new gaming paradigm being pushed by companies [PUB20].

Cloud/edge-assisted machine learning Mobile apps with machine learning features (e.g., image/object recognition or speech understanding [CRD15, GJB17]) also pose latency requirements. Network latency becomes a bottleneck for smart assistants, such as Alexa [Ama20] and Siri [Sir20]. Users may tolerate at most 200ms response time, while deep learning based local transcriptions take only 10ms [CR19].

Networking usage patterns by these mobile apps All the above representative mobile apps involve *frequent and regular uplink* data transfer. The mobile VR, sensing, and gaming [Waz20, Ube20] applications collect data from device sensors and upload them to the server for subsequent actions. These sensors typically produce small data *periodically.* The user can only configure the sampling periodicity through the API provided by the mobile OS [Goo20]. The machine learning based apps also have predictable traffic. They typically perform local computations with predictable latency before an uplink data transfer. For example, face recognition apps process a video frame locally using a fixed-sized neural network (NN). A user can gauge the delay based on the NN size. Emerging robotic or drone-based applications perform local tasks for a certain duration (e.g., scanning the surrounding environment for a few seconds [ADL19]) before uploading the result. Such apps

Table 7.1: LTE latency (ms) for two mobile apps.

App	Latency	AT&T	T-Mobile	Verizon	Sprint
PUBG	UL Net	10.7	9.9	10.0	17.7
	DL Net	5.0	5.0	5.0	5.0
	UL/Total	68.2%	66.4%	66.7%	78.0%
VR	UL Net	N/A [1]	18.4	23.8	N/A
	DL Net	N/A	8.5	10.6	N/A
	UL/Total	N/A	68.4%	69.2%	N/A

also exhibit uplink traffic that can be accurately predicted.

7.2 Latency Analysis

In this section, we empirically analyze where the application-perceived LTE latency stems from. We address two issues:

- *How large can LTE uplink latency be over operational 4G networks?* We use measurements to quantify it in §7.2.1.
- *Why is the uplink latency prohibitively high over LTE?* We break down this latency into multiple elements. We quantify their impact, identify root causes, and share insights in §7.2.2.

7.2.1 Measuring Latency in 4G

We quantify the uplink latency over operational LTE networks via measurements and trace analysis.

[1]VR cannot run on AT&T and Sprint, since their firewalls block the traffic.

Methodology We analyze the traces from our showcase VR game and another popular mobile application PUBG Mobile [PUB20]. Our VR application uploads user motion packets (∼60Bytes) and receives 60FPS, 5Mbps downlink video stream. The downlink data packets are sent from the deployed server to the device over LTE. PUBG is a mobile game with frequent uplink data (∼40ms interval) and downlink responses. Both uplink and downlink packets are small (<100Bytes). The latency due to server processing is less than 1ms. The mobile devices (a Pixel 2 and a Pixel XL) run the apps. We collect both app logs and LTE signaling traces via MobileInsight [LPY16]. We carry out our experiments over four US mobile carriers from 12/2019 to 09/2020. The tests cover static, low-mobility (∼1m/s), and high-mobility (∼30mph) cases, with varying signal strength (-120∼-80dBm).

Results We first measure the LTE uplink latency. We monitor the device buffer and compute the latency for each data packet. This information is available in the MobileInsight message "LTE MAC UL Buffer Status Internal". Despite small packet size, the uplink latency turns out to be non-negligible, as shown in Table 7.1. For all four carriers, the uplink latency (UL NET) ranges from 9.9-17.7ms for PUBG and 18.4-23.8ms for VR. These latency values might not meet the requirements of a number of latency-sensitive apps [Abr14].

Who is the latency bottleneck? We further discover that, instead of downlink, the uplink latency poses as a major component in overall latency. We compute the downlink latency from logs of "MAC DL Transport Block" in MobileInsight. The results (DL NET) are in Table 7.1. We see that, uplink latency accounts for 66.4-78.0% in PUBG and 68.4-69.2% in VR. Surprisingly, even for the downlink-heavy VR app, uplink latency still contributes to a large portion of the overall latency. Recent techniques (e.g., MIMO and carrier aggregation) and 5G further reduce the DL latency with faster PHY designs. In contrast, as we will see later, the scheduling design employed for the uplink will likely be retained in 5G. As a result, we will focus on the uplink latency in this dissertation.

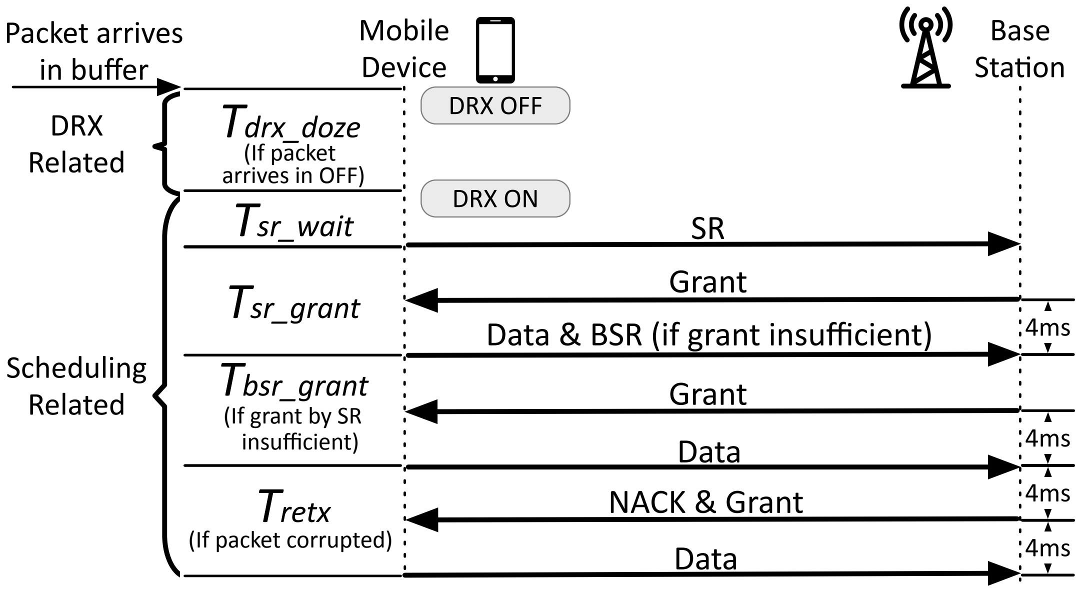


Figure 7.2: LTE uplink procedure & latency elements.

7.2.2 Why Long Latency: Breakdown Analysis

We next analyze the root causes for long network latency in 4G LTE. We identify various latency elements for the LTE uplink latency by analyzing the 3GPP standards [3GP19c, 3GP19b].

We breakdown the uplink latency as shown in Figure 7.2. The average number of each latency element is shown in Table 7.2. We can observe that, the major uplink latency bottlenecks are T_{drx_doze}, T_{sr_grant}, and T_{sr_wait}, while T_{bsr_grant} and T_{retx} are one magnitude smaller compared to other elements. We will see how each latency element acts and why it poses or does not pose as the latency bottleneck to our applications.

7.2.2.1 DRX Doze Latency.

Power-Saving Mode through DRX

The DRX state transition is shown in Figure 2.3. In the Long/Short cycle state, if any downlink data is received during the ON period, the device enters the CRX state and starts the drx-InactivityTimer. Upon sending an uplink data, the device initiates an SR

Table 7.2: Measured latency elements for VR application. T_{drx_doze} is the average value when present.

Latency (ms)	AT&T	T-Mobile	Verizon	Sprint
T_{drx_doze}	29.7	31.9	28.3	29.2
T_{sr_wait}	4.4	4.4	4.6	9.0
T_{sr_grant}	8.2	8.5	8.0	10.1
T_{bsr_grant}	0.03	0.00	0.03	0.16
T_{retx}	0.17	0.14	0.32	0.72

request. It then switches to the CRX state as well. If the device receives downlink data or initiates another SR request, the timer restarts. The short DRX state is entered once the drx-InactivityTimer expires. In this state, the device enters long DRX after the number of drxShortCycleTimer short cycles. All such involved timer parameters are negotiated between the device and the BS during connection setup through RRC.

How downlink DRX incurs long uplink latency DRX is designated for power saving over *downlink* transmissions. It should not block any uplink transfer. In fact, the 3GPP specification [3GP19c] stipulates that, upon the uplink sending an SR, downlink DRX should enter the CRX state as if receiving a downlink data packet. However, we found that this is not the case in practice. A new data packet refuses to invoke an SR if the device is in the doze mode. Instead, it continues to doze for a while (the time is denoted as T_{drx_doze}). It then waits for an SR slot to initiate the SR, while migrating the device to the CRX state. Table 7.2 shows that, T_{drx_doze} is 28.3-31.9ms on average in the four carriers. The maximum latency is 59ms with the 90th percentile being 42ms.

Note that the DRX doze latency is different from the known downlink packet delay due to waiting for DRX ON state. 3GPP [3GP19b, 3GP19c] does not mandate to prepare for SR at the DRX state. Although this latency element is not standardized, it is common for vendors as they use DRX doze to save energy. The DRX-induced doze timer is hinted

Table 7.3: **Critical LTE parameters for uplink latency.**

Parameters		AT&T	T-Mobile	Verizon	Sprint
T_{sr_grant}	8ms	96.6%	96.5%	98.8%	0
	10ms	0	0.2%	0.1%	98.1%
	others	3.4%	3.3%	1.2%	1.9%
$T_{sr\_periodicity}$	10ms	94.0%	98.1%	92.3%	11.9%
	20ms	6.0%	1.9%	0	48.9%
	40ms	0	0	7.7%	39.%
$T_{inactivity\_timer}$	200ms	100.0%	99.5%	99.6%	84.5%
	others	0	0.5%	0.4%	15.5%

in Qualcomm patents [YC17], where the device defers its SR during DRX OFF for energy savings. We also indirectly validate this behavior in a ZTE Z820 with Mediatek Chipset. For packets with an interval of 1 second, the measured average RTT is 35ms longer than that of packets with a small interval.

Insight: A packet keeps the device at the CRX state for $T_{inactivity\_timer}$. The idea is to reduce the DRX doze latency by sending a dummy message in advance. This way, the device is kept in the ON period before data arrival. The data packet can thus be sent without deferring until the doze period ends.

7.2.2.2 Scheduling Latency.

Uplink/downlink scheduling in LTE: In LTE, the uplink and downlink data transfers take different approaches:

• *Uplink data transfer over LTE* The uplink data transmission is through PUSCH (Physical Uplink Shared Channel). All data transmissions are regulated by the BS, which allocates resource blocks (RBs) for the actual transfer. An RB is the smallest unit allocated for a device.

In the scheduling-based LTE design, uplink data cannot be immediately sent out before the device is granted resource. This is done via the request and grant mechanism. Specifically, the device sends an SR (Scheduling Request) through PUCCH (Physical Uplink Control Channel). SR is a signaling message notifying new data arrival at the mobile device. Moreover, an SR signal cannot be sent at any time instantly. It can only be sent during certain subframes (called SR occasions). The periodicity of SR occasions is notified by the BS during connection setup. Upon receiving an SR, the BS returns an uplink Grant (i.e., grant) to the device. A grant specifies what RBs and modulation the device could use *4ms later*. The number of RBs in response to an SR depends on the BS configuration, since SR is just a message stating "device has data to send" without specifying the amount.

• *Downlink data transfer over LTE* LTE still uses the scheduling-based operations for its downlink. However, BS directly allocates RBs for each device upon data arrival, since BS knows what data to transmit to which device.

How scheduling incurs long latency The device also suffers from its uplink scheduling latency. It must wait for an SR occasion before receiving a grant from the BS to upload its data packet. The latency element, denoted as T_{sr_wait}, is thus affected by the periodicity of an SR occasion $T_{sr\_periodicity}$. The device then waits for a grant, which the device could use 4ms later. The latency from sending the SR to sending the data packet is denoted as T_{sr_grant}. The two elements of scheduling are shown in Figure 7.2. We measure them in Table 7.2. The SR waiting latency T_{sr_wait} is 4.4ms for AT&T, 4.4ms for T-Mobile, 4.6ms for Verizon, and 9.0ms for Sprint. Sprint has the largest T_{sr_wait} because it has the longest SR cycle. T_{sr_grant} is 8.2ms, 8.5ms, 8.0ms, and 10.1ms for the four carriers. The accumulative latency is denoted as $T_{scheduling} = T_{sr\_wait} + T_{sr\_grant}$.

Insight: This scheduling latency can be reduced. If a grant is pending at the device, a new arriving data packet can use it for transfer. Therefore, we may use a dummy message to request for a grant in advance, so that the data packet can use this grant for actual transfer without delay.

7.2.2.3 Other Latency Elements.

Buffer status report (BSR) SR is an indicator that informs the BS of new pending data, without specifying *how much*. When the packet that triggers SR is large, the initial grant might be insufficient. The device then sends a BSR (Buffer Status Report) together with the data packets in the scheduled RBs. Unlike SR, a BSR includes the info on how much data still remains in the device buffer. Upon receiving the BSR, the BS will process it and respond with sufficient grants for the buffered uplink data.

We note BSR's impact on uplink latency is negligible for most applications in §7.1. The latency between a BSR and the time to use the grant (denoted as T_{bsr_grant}) is illustrated in Figure 7.2. Conceptually, it is the request processing time + 4ms, similarly to T_{sr_grant} ($\approx 10ms$). However, it equals to 0 when the initial grant is sufficient. The measurement results are in Table 7.2. The BSR latency is less than 1ms on average for four US carriers. This is because a base station usually provides a large grant (>100B) sufficient for our apps in response to SR,

Retransmission in LTE An uplink data packet might be corrupted during transfer. Upon receiving a corrupted packet, the BS notifies the device by sending a NACK and a grant. The device uses the grant to retransmit the corrupted data. Similar to BSR latency, the retransmission has limited impact on the uplink latency for apps in §7.1. The ReTx latency for uplink data packet is fixed at 8ms if needed [3GP19b] and 0 otherwise. We denote this latency as T_{retx} and the procedure is shown in Figure 7.2. Among all data packets, 2.1% in AT&T, 1.7% in T-Mobile, 4.0% in Verizon, and 9.1% in Sprint perceive ReTx latency. Less than 1ms latency is incurred on average, shown in Table 7.2. Unlike downlink with up to 10% retransmissions [TLL18], uplink packets are small and less prone to corruption.

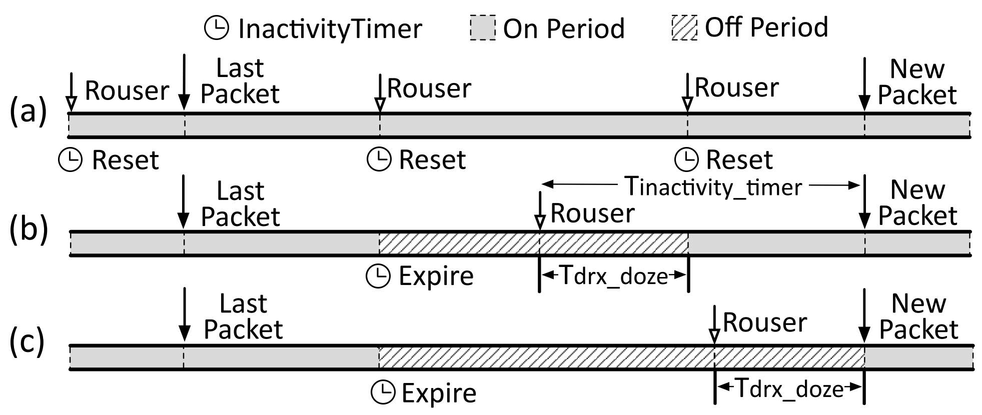


Figure 7.3: Component solution to DRX doze latency.

7.3 LRP: Latency Reduction for Each Component

7.3.1 Energy-Efficient DRX Doze Elimination

To reduce the DRX doze latency in §7.2.2.1, LRP should ensure the device is in ON period when a data packet arrives at the device buffer. As an application layer solution, LRP cannot directly switch the device to the CRX state that needs firmware modification. Instead, it sends a dummy packet (*rouser*) *before* the data packet's arrival.

Despite being straightforward at the first glance, a *rouser* is only effective if being sent at the right time. An imprudent *rouser* can either incur unacceptable energy waste or cannot help reduce latency. Therefore, timing control is crucial to balancing latency and energy cost. We first discuss some naive solutions with limitations, and then present our design.

Naive timing control One naive solution is to keep DRX at CRX state at all times by frequently sending *rouser*s. As shown in Figure 7.3(a), this can be achieved by sending a *rouser* every $T_{inactivity\_timer}$. Unfortunately, this results in unacceptable energy waste, as the device never enters the doze mode.

A better choice is to send a *rouser* with the time in advance, denoted as t_r, being set to $t_r = T_{inactivity\_timer}$ (Figure 7.3(b)). On one hand, as the packet keeps the ON period for $T_{inactivity\_timer}$ after dozing, $t_r = T_{inactivity\_timer}$ ensures that the data packet enters the buffer during the ON period. On the other hand, this saves power compared to the first naive choice,

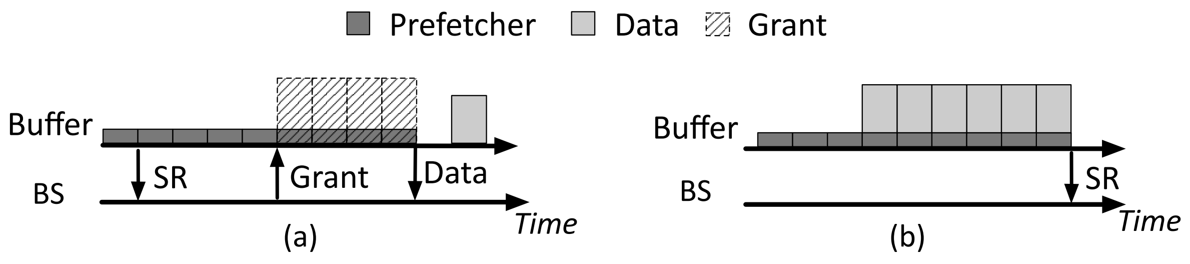


Figure 7.4: Impact of *prefetcher* Timing.

since the extra ON period is capped at $T_{inactivity\_timer}$ for each packet at most. However, extra energy consumption is still incurred. Since $T_{inactivity\_timer}$ (∼200ms) is typically much larger than T_{drx_doze} (∼30ms) in reality, the ON period between wakeup from the doze mode and the data packet is unnecessary.

LRP's approach LRP prioritizes latency over marginal energy waste with proper timing control. Instead of frequent *rouser*s in naive solutions, LRP only sends a *rouser* for the time T_{drx_doze} in advance. We thus keep updating the maximum T_{drx_doze}, denoted as $T_{drx\_doze\_max}$. The timing to send the *rouser* is $t_r = T_{drx\_doze\_max}$. If the device enters the ON period during doze, i.e. $t_r > T_{drx\_doze}$, the *rouser* finishes dozing before the data packet arrives, thus eliminating the doze latency for the data packet. It is also likely that T_{drx_doze} for a *rouser* exceeds t_r. In this case, the packet enters the buffer and endures the dozing latency together with the *rouser*. Although the doze latency is not eliminated, the *rouser* reduces it by t_r.

7.3.2 Resource-Efficient Proactive Scheduling

LRP next seeks to mask the round trips of the scheduling in §7.2.2.2 for the mobile app. The idea is to send a scheduling request (SR) *before* the arrival of the data, so that the data does not need to wait for the radio grants. As an application-layer solution, LRP cannot directly trigger the SR early (which requires modifying the firmware). Instead, it requests a grant from the BS in advance by sending a dummy message, named *prefetcher*. This is feasible since the grant is not tied with the packet that requests it. Moreover, since the BS responds to each SR regardless of the pending data size, a small dummy message can receive a grant that allows for much-larger-size transmission than itself, thus sufficing to accommodate the followup data packet transfer in a single transmission.

Similar to the DRX doze elimination in §7.3.1, an effective *prefetcher* also needs accurate timing control. As shown in Figure 7.4, imprudent timing can offset the latency reduction, and/or waste radio resources. We next discuss both naive solutions in Figure 7.4, and then show `LRP`'s approach.

Naive timing control A too early *prefetcher* might result in both resource waste and prolonged latency as shown in Figure 7.4(a). The *prefetcher* is sent too early so that the timing to use the returned grant is already passed when the data packet arrives. The resource is thus wasted, while the data packet misses the opportunity to reduce its scheduling latency.

Similarly, a late *prefetcher* could also miss the opportunity to reduce the scheduling latency for the data packet, as shown in Figure 7.4(b). If the *prefetcher* is sent too late after a potential SR that could reduce latency, the data packet might have to wait for scheduling latency as if no *prefetcher* is issued. In the worst case for both early and late *prefetcher*, it may result in missed latency savings up to $T_{sr\_periodicity} + T_{sr\_grant}$.

`LRP`'s approach `LRP` aims at reducing the scheduling latency at marginal radio resource cost. Let a *prefetcher* be sent t_p before the data packet. The parameter t_p must meet two requirements. First, we should ensure $t_p \geq T_{sr\_grant}$. Note that, an SR can only request a grant to be used at T_{sr_grant} after the SR. Therefore, $t_p \geq T_{sr\_grant}$ guarantees that the SR is sent only if it helps to reduce the scheduling latency for the data packet. Second, we must ensure $t_p \leq T_{sr\_grant}$. This is to let the requested grant be used to transmit the data packet. No resource waste or premature SR is incurred.

Consequently, our timing design is to set the time advance as $t_p = T_{sr\_grant}$, which meets both requirements. Note that T_{sr_grant} is typically constant for a BS, being the accumulative latency of SR processing latency + 4ms, where 4ms is a standardized parameter in [3GP19b]. In our experiment, more than 96.5% of T_{sr_grant} is identical under a BS regardless of the carrier. If T_{sr_grant} changes after handover to a new BS, we update T_{sr_grant} immediately. Even if T_{sr_grant} may vary, our solution is no worse than the current practice.

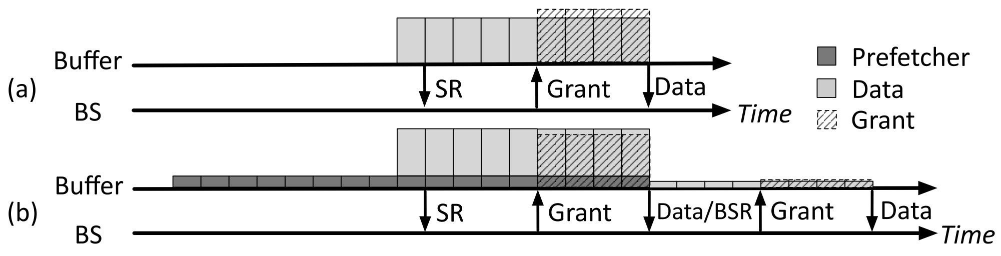


Figure 7.5: Corner case: a *prefetcher* increases latency.

Impact of the data packet size A *prefetcher* helps reduce scheduling latency if the data packet size ≤ grant - *prefetcher* size, which is common in reality as >99% of initial grants in our experiments exceed 100B in all operators, while the uplink sensory data is smaller than half of that. Therefore, a *prefetcher* initiates an SR, and gets a returned grant that suffices for the data packet to be sent with the *prefetcher*.

However, a corner case arises when the grant in response to SR is enough for the data packet, but not for a *prefetcher* + the data packet. As shown in Figure 7.5(b), the device could only send the *prefetcher* and a portion of the data packet. A BSR further requests a grant for the remaining data. The data packet thus suffers extra BSR latency compared to the case without *prefetcher* (Figure 7.5(a)). In the worst-case scenario, this latency increases by T_{sr_grant} (∼8ms). We discuss the probability of this case in Appendix C. However, even in this corner case, the worst case happens only when the data and the *prefetcher* arrive in the same SR period, with probability T_{sr_grant}/T_{sr_wait}. For other conditions in the corner case, the latency is the same as vanilla LTE.

7.3.3 Handling the Conflicts for Low Latency

LRP further resolves several conflicts for overall latency reduction. Figure 7.6 illustrates the workflow of LRP. Let $T_{interval}$ be the time interval between the last and the next expected packet. LRP thus reduces various latency elements. It handles improper interplay between latency elements, and between dummy and data packets.

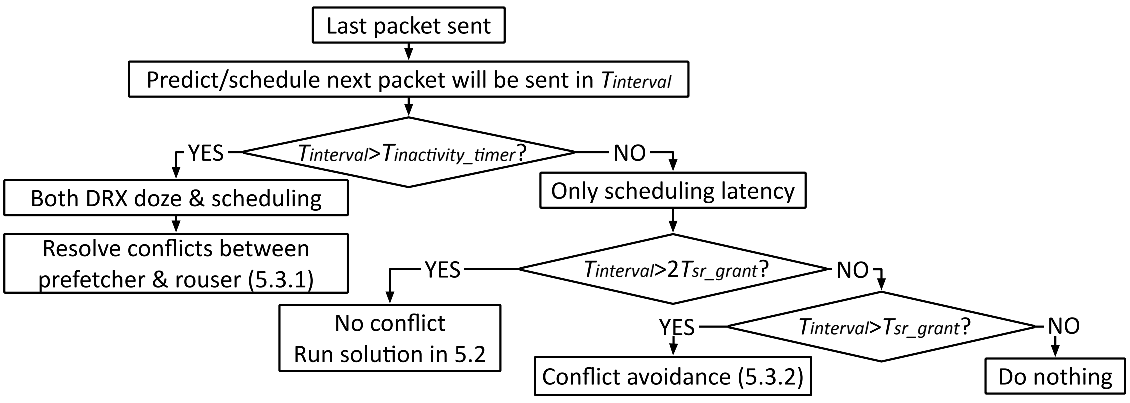

Last packet sent
Predict/schedule next packet will be sent in $T_{interval}$
$T_{interval}>T_{inactivity\_timer}$?
YES
NO
Both DRX doze & scheduling
Resolve conflicts between prefetcher & rouser (5.3.1)
Only scheduling latency
$T_{interval}>2T_{sr\_grant}$?
YES
NO
No conflict Run solution in 5.2
$T_{interval}>T_{sr\_grant}$?
YES
NO
Conflict avoidance (5.3.2)
Do nothing

Figure 7.6: The workflow of conflict handling in LRP.

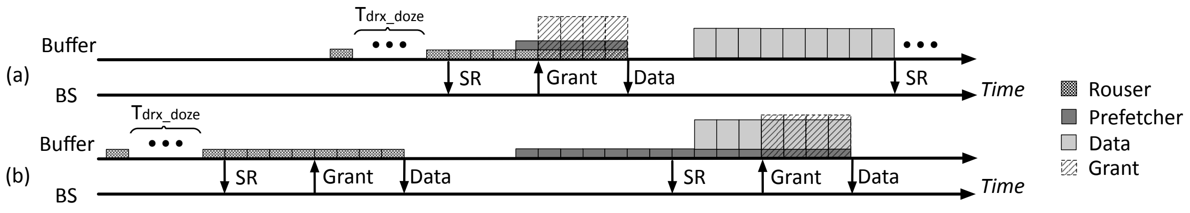


Figure 7.7: Improper timing control causes conflicts between components.

7.3.3.1 Conflict Resolution Between Latency Elements

LRP issues two types of dummy packets for latency reduction: *rouser*s for DRX-induced doze latency, and *prefetcher*s for scheduling latency. Figure 7.7 illustrates their conflicts. A *rouser* itself is a dummy message that needs to be sent before a *prefetcher*. Once turning the device to DRX ON, it asks for the grant, which could carry both *rouser* and *prefetcher*. Therefore,

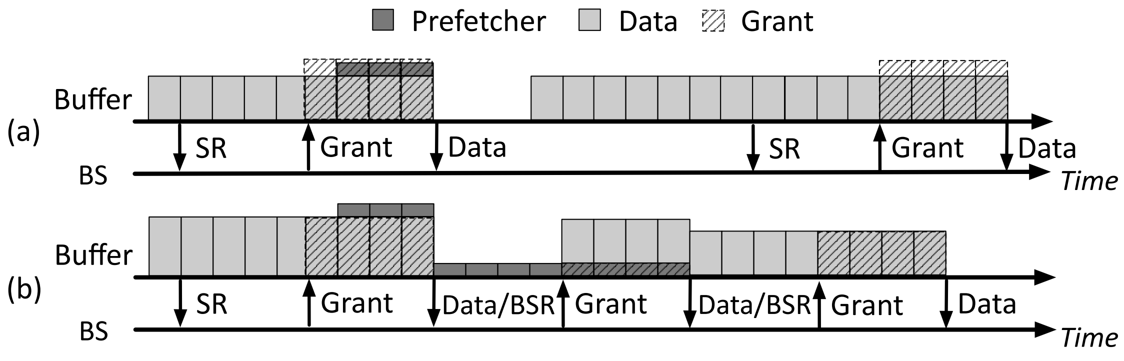


Figure 7.8: Conflicts: dummy messages & data.

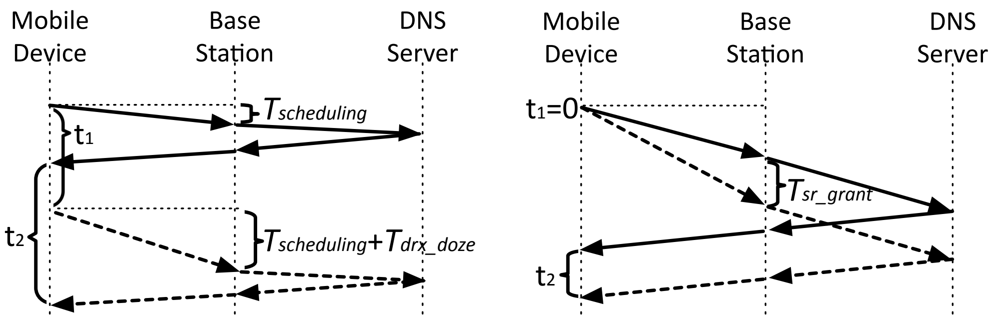
the *prefetcher* is sent by the grant requested by the *rouser*. The grant-induced scheduling latency is not reduced at all. The latency penalty can be as large as $T_{sr\_periodicity} + T_{sr\_grant}$ compared to no-conflict case in Figure 7.8.

To resolve this conflict, we refine the timing control to ensure both dummy packets' effectiveness. Specifically, we should make sure a *rouser* is sent when a *prefetcher* hits the device buffer, so that the *prefetcher* can take effect and reduce the scheduling latency. A *rouser* takes at most $T_{sr\_periodicity} + T_{sr\_grant}$ to be sent out as a dummy message and a *prefetcher* needs to be sent T_{sr_grant} before the data packet. Therefore, we adapt the timer from $t_p = T_{drx\_doze\_max}$ to $T_{drx\_doze\_max} + T_{sr\_periodicity} + 2T_{sr\_grant}$ to ensure a *rouser* is sent before a *prefetcher*. The *rouser* thus endures $T_{drx\_doze\_max}$ that guarantees the doze is completed and then sent out.

7.3.3.2 Conflict Avoidance Between Dummy and Data

The next conflict arises between LRP's dummy packets and the last legitimate data packet. If a *rouser* conflicts with the last packet, this does not pose an issue: the *rouser* can still help the device to remain in the ON period for $T_{inactivity\_timer}$. We thus only discuss where a *prefetcher* intervenes with the last packet. We show how LRP adapts this for latency reduction.

There are two instances when a *prefetcher* arrives in the buffer *before* the last data packet being completely sent out, shown in Figure 7.7 and Figure 7.8. In case (a), the *prefetcher* does not provide any latency reduction. The grant for the last packet has enough room to carry the *prefetcher*, which will be sent together. There is no *prefetcher*-requesting grant for the next data packet. In case (b), a *prefetcher* may increase the latency. The grant for the last data packet cannot accommodate the piggy-backed transmission of the *prefetcher*. A BSR request is thus triggered by the device to request for more grants. Since BSR specifies the size for the dummy message *prefetcher*, the returned grant does not suffice to transmit the data packet. This subsequently invokes another round of BSR-grant operations. The data packet might suffer from extra BSR latency.



(a) T_{drx_doze} (b) T_{sr_grant}

Figure 7.9: Inferring parameters at application layer.

To avoid the conflicts, LRP adjusts the timing of a *prefetcher*. It leaves enough time for the last packet to complete its transmission before the *prefetcher*. Recall that the theoretical maximum uplink latency that the last packet would experience after optimization is T_{sr_grant}. The dummy *prefetcher* is then sent at least T_{sr_grant} after the application sent its last packet. Specifically, if the time gap (between the last packet arrival and the next packet arrival) is larger than $2T_{sr\_grant}$, we send a *prefetcher* T_{sr_grant} before the next packet. This is the timing we designed in §7.3.2; it will not break the above condition. Otherwise, we send a *prefetcher* T_{sr_grant} after the last packet. This choice will reduce less latency compared to the timing in §7.3.2 without conflicts. However, we avoid the cases where a conflict negatively affects the latency.

7.4 Rootless Inference of Critical Parameters

As shown in §7.3.1–7.3.3, LRP relies on knowing certain LTE parameters for latency reduction. Obtaining such parameters through the root privilege can definitely work. However, such an approach limits the applicability of LRP. To let LRP work with *every* commodity device, we seek to infer these parameters at the application layer. Note that existing tools typically require system privilege (e.g., MobileInsight [LPY16]) or additional hardware (e.g., QXDM [Qua]).

To infer these critical LTE timers, LRP exploits packet pairs for probing. Figure 7.9 shows the general procedure. `LRP` sends two adjacent probing requests and records their interval t_1. Upon receiving the responses to both packets, `LRP` compares the responses' intervals t_2 with t_1, and estimates the corresponding timers. This approach is based on the premise that, the difference between t_1 and t_2 mainly arises from the different uplink LTE latency experienced by two packets. This premise largely holds in practice, because latency fluctuations from the base station are much larger than those in the core network or servers[2]. Compared with the conventional packet-pair technique, `LRP` customizes probing packets with the LTE domain knowledge for accurate inference.

Inferring DRX-related parameters To reduce DRX doze latency, `LRP` should know $T_{drx\_doze\_max}$ (§7.3.1). Recall that the DRX doze latency is only present when the packet interval is large. The idea is to let t_1 be large enough so that the first response packet cannot keep the second request in DRX ON. The second packet in the pair experiences UL DRX doze latency, while the first does not as we immediately start next pair after one is done. We can thus use the interval difference $t_2 - t_1$ to infer DRX doze latency. Considering that two requests can also be different in terms of scheduling latency, we repeat the pair for 10 times and take the interval difference average. Figure 7.9(a) illustrates this procedure.

One caveat is that we need to know how large t_1 is so that the second request suffers from DRX doze latency. We increase the interval t_1 gradually until a certain spike appears in measured RTT for the second request, caused by DRX doze latency ($\approx$30ms as shown in §7.2). The time interval between the first response and the second request that triggers such spike infers $T_{inactivity\_timer}$. We can thus infer $T_{drx\_doze} = t_2 - t_1$. We take the max in multiple rounds as $T_{drx\_doze\_max}$.

Inferring scheduling-related parameters `LRP` needs T_{sr_grant} to reduce the scheduling latency (§7.3.2). Figure 7.9(b) shows how `LRP` infers it. We let t_1 as 0 by sending both requests together. As we just showed, the grant is sufficient for a single request packet. We increase the size of the second request so that the grant will not be sufficient for both request

[2]We have validated this premise in operational LTE. We send a pair of DNS requests at $t_1 = 0$. A UL grant suffices to send both requests; they arrive at the BS simultaneously. t_2 is solely affected by the core network and DL. The results show that $t_2 < 1ms$ for >99% responses.

packets. According to the scheme, the first request experiences only scheduling latency while the second experiences the same scheduling latency plus T_{bsr_grant}, which equals to T_{sr_grant} under a same BS. We thus can derive `LRP` optimization parameter T_{sr_grant} from the measured t_2 as $T_{sr\_grant} = t_2$.

7.5 Miscellaneous Issues for `LRP`

Applicability to 5G In principle, `LRP` is applicable to 5G, which has three usage cases. Enhanced Mobile Broadband (eMBB) extends 4G technology. Massive Machine Type Communication (mMTC) is for cellular IoT, whose design is based on LTE-M and NB-IoT [AAP17]. The scheduling mechanisms and handover logic of both modes largely remain unchanged [3GP20e, 3GP20d]. Consequently, `LRP` can still diagnose latency components and reduce latency for these 5G scenarios. Ultra Reliable Low Latency Communications (URLLC) targets low-latency communication. Although not fully standardized, the potential grant-free scheduling might partially achieve `LRP`'s latency reduction for scheduling latency. However, `LRP`'s latency diagnosis, DRX doze latency reduction, and handover prediction will still help URLLC applications.

Energy Analysis `LRP` incurs extra energy overhead from four sources. First, transmitting a *rouser* incurs a longer ON period. The time to send a *rouser* can be as long as $T_{sr\_periodicity} + T_{sr\_grant}$. It incurs $T_{sr\_periodicity}/2 + T_{sr\_grant}$ on average. Second, T_{drx_doze} is not predictable so we select $T_{drx\_doze\_max}$ to prioritize latency over energy. The extra ON period is $\delta = T_{drx\_doze\_max} - T_{drx\_doze}$ for each *rouser*. If the packet arrives during DRX OFF, $T_{drx\_doze\_max}$ equals to T_{drx_doze} and $\delta = 0$. Otherwise, $T_{drx\_doze} = 0$ and $\delta = T_{drx\_doze\_max}$. The expectation of δ is thus $p_{on} \cdot T_{drx\_doze\_max}$, where p_{on} is the probability of a packet arriving during DRX ON period. If we ignore background traffic and assume the packet arrives in the buffer at a random time, p_{on} = onDurationTimer / DRX cycle. Third, an early *rouser* (due to inaccurate estimation) also causes a longer ON period. Denote ϵ as the estimation error. When the *rouser* arrives during DRX OFF, the extra ON period is ϵ. Otherwise, ϵ incurs no extra ON period. Finally, sending extra small messages incurs extra energy waste.

Impact on the spectrum efficiency For every data packet, we define its spectrum efficiency $SE = \frac{\text{sizeof (data packet)}}{\text{sizeof (Total UL resource granted)}}$. When a data packet suffers from doze latency and LRP sends a *rouser*, it reduces SE by half: the *rouser* and the *prefetcher* initiate two grants, while the legacy LTE only requests for one. The extra grant occupies ≈2 RB in commercial networks. LRP trades-off SE for low latency. When LRP sends a *prefetcher* only, two scenarios arise. In the normal case, the grant from SR can carry both the data packet and a *prefetcher*. Therefore, LRP requests no extra grant and SE is the same as the legacy LTE. In the corner case discussed in §7.3.2, the BS allocates at most sizeof (*prefetcher*) extra grant. One extra RB is thus wasted, since a single RB is sufficient to carry a *prefetcher*. SE is reduced by $\frac{\text{sizeof } (prefetcher)}{\text{sizeof } (prefetcher + \text{grant from SR})}$. This value multiplying the probability of the corner case (see Appendix C) yields the expectation of SE reduction.

Impact of background traffic LRP still reduces latency in the presence of background traffic. No matter whether the background packet is sent before a *rouser* or between a *rouser* and a data packet, the *rouser* will keep the data packet at the DRX ON state, thus eliminating the DRX doze latency. On scheduling latency, if the background traffic is sent after the data packet, it does not affect the *prefetcher*. If the background traffic is in between, the *prefetcher* reduces its latency, which indirectly reduces latency for the real data packet. It still does not increase the latency compared to legacy LTE without LRP. When the background traffic is sent before a *prefetcher*, it will be sent out through BSR before the data packet in the worst case, equivalent to no optimization.

What if the uplink traffic is not strictly regular? While mobile sensors produce regular data packets, the actual uplink data packets might not be strictly periodic. This can be caused by mobile OS overhead, prediction inaccuracy, or sensor periodicity variance. LRP still guarantees no worse latency than legacy LTE, and saves LTE latency in most scenarios. We show the following Theorem 7.5.1 and prove it in Appendix D.

Theorem 7.5.1. For the data packet that should have arrived at $T_{interval}$ but actually arrives at T, LRP does not incur extra latency compared with the legacy 4G LTE.

LRP **for non-regular traffic** For those ML/AI apps of §7.1 with irregular but predictable uplink traffic, `LRP` works equally well. For others with irregular yet unpredictable uplink traffic, we do not recommend `LRP` for such apps. If users intend to use our APIs, latency reduction cannot be ensured.

CHAPTER 8

Building SDR-based 5G/4G Systems for Validation

This chapter elaborates on our implementation and evaluation of the proposed attacks and countermeasures. We first introduce two software-defined radio (SDR) testbeds we build to evaluate out designs (§8.1). They are respectively built for broadband devices and IoT devices. In the remaining of the chapter, we elaborate on the implementation and evaluation of each solution component: `CDS` (§8.2), `FANE` (§8.3), `DP`^{2} (§8.4), `CellDAM` (§8.5), and `LRP` (§8.6).

8.1 Building Standard-Compliant 5G/4G Testbeds

To evaluate the attacks and solutions, we develop our own software-defined radio (SDR) testbeds. Such systems require an RF device, such as a USRP [Rad] board to receive and transmit data. All computation and processing logic are done on a server, which is connected to the USRP. SDR systems have two advantages. First, the system is affordable with general-purpose server machine and USRP board, It does not require specialized 5G/4G base station or core network hardware boxes. Second, for any new solution that requires changing protocol stack (e.g., `DP`^{2}), SDR-based systems can quickly prototype them for evaluations. Updating software stack is sufficient while no hardware changes are necessary.

8.1.1 Software-Defined, Intelligent Testbed for 5G/4G Broadband

We first build an open-source, standard-compliant software-defined radio 5G/4G system named Flora [Flo]. The overview is shown in Figure 8.1. It is based on open-source SDR systems OpenAirInterface [Ope18] and srsRAN [SRS]. Its salient features include in-network

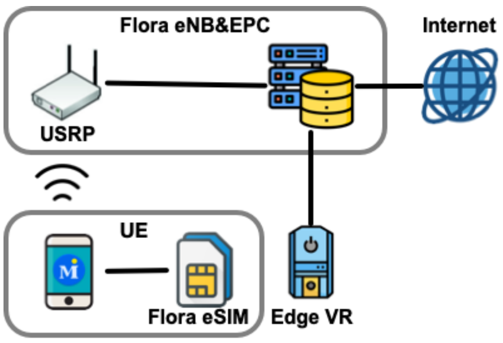


Figure 8.1: Overview of Flora testbed.

analytics, run-time intelligence, and new feature support (carrier aggregation, handover, etc.). Flora does not depend on any specialized hardware (SIM, chipset, or infrastructure) and works with commercial-off-the-shelf devices. A commercial-off-the-shelf (COTS) device can connect to the base station and core networks with a customized SIM card. We can implement and integrate our attacks and defenses in our Flora testbed. We will use it to confirm the feasibility of our design, and to help understand the practical issues in implementation and deployment.

8.1.2 Building an NB-IoT Testbed for IoT Devices

We present Sonica [Son], our open-source NB-IoT prototyping platform. It provides an implementation of eNB and core network that supports commonly deployed Release 14 of NB-IoT standard, and is able to interact with other commercial off-the-shelf (COTS) components without special configuration specific to certain hardware model. Having a prototyping platform can be critical in NB-IoT research as it can help understand the new technology and fully explore it. There have been SDR-based systems [Ama21, CCH20] that claim support for NB-IoT. However, the Amarisoft system [Ama21] is closed source, and the attempt in OpenAirInterface [CCH20] implements Release 13 NB-IoT, which is not compatible with the current Release 14 or future releases. Meanwhile, Sonica provides a flexible framework to construct custom NB-IoT nodes for analysis.

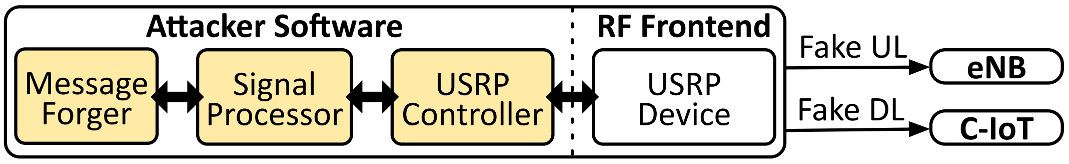


Figure 8.2: The implementation of the CDS attacker node.

8.2 Evaluation of CDS

8.2.1 Implementing CDS

The overall implementation is presented in Figure 8.2. USRP acts as an RF frontend. The software part contains three components: a USRP Controller, a Signal Processor, and a Message Forger. The USRP Controller communicates with the USRP. It provides interfaces for the Signal Processor to send and receive wireless C-IoT signals. Meanwhile, it does correction on signal according to hardware (detailed below). The Signal Processor carries out PHY processing, decoding messages from the eavesdropped signal while encoding forged messages into wireless signals. We also allow the Signal Processor to send noises to corrupt a specific channel. The Message Forger implements a simplified NB-IoT protocol stack (Up to RLC). It receives demodulated data from Signal Processor and crafts the fake signaling. To forge a signaling message, it passes the data to the Signal Processor.

USRP Controller The USRP Controller uses the API from the RF frontend to assist Signal Processor in eavesdropping on and sending wireless signals. Besides, it ensures correct timing and frequency synchronizations. For timing, USRP Controller aligns itself with the synchronization signals (NPSS and NSSS) from eNB. For frequency, it tackles two critical roadblocks: *Carrier frequency offset* (CFO) and *Sampling frequency offset* (SFO). The frequency offsets can cause failure in attack if not properly handled. While a GPS-Disciplined Oscillator can be applied to minimize the difference, we adopt a more cost-effective approach. The USRP Controller estimates the CFO and SFO between itself and the eNB with synchronization signals. Then it carries out the correction on transmitting (after it gets signals from the Signal Processor) and receiving (before passing signals to the Signal Processor). For CFO, the attacker applies a phase rotation on the signals. While for SFO, the attacker inserts or removes samples to compensate when the difference in sampling accumulates to a certain degree.

Signal Processor The Signal Processor is mainly responsible for PHY layer processing. Our implementation extends the signal processing logic from srsRAN [GGS16]. Concretely, we realize the modulation of NB-IoT data (in both UL and DL) and noise signals. The processing follows 3GPP standards [3GP20c, 3GP19b] and supports different modulation schemes. It decodes relevant messages of interest (e.g. DCI or data-plane signaling messages intended for the victim) and forwards them to Message Forger. When the Message Forger generates the attack traffic (forged signaling or noise) the Signal Processor generates the corresponding signals according to the request.

Message Forger Based on the decoding results from the Signal Processor, the Message Forger crafts fake data-plane signaling messages. The victim will receive the forged message instead of the original authentic one because of the stronger signal strength of the attacker. The fake messages follow MAC/RLC standards [3GP21e,3GP21f]. They appear to be legitimate messages but are specially designed to inflict certain damages on the device or network. Forged messages are then forwarded messages to Signal Processor for modulation.

8.2.2 Testbed for Evaluation

Our testbed is shown in Figure 8.3. A Surface Pro 3 with Intel i7-4650U processor and 8G RAM runs the attacker software. It implements the attack logic and controls the forged signaling on Ubuntu 20.04. The attacker software connects to a USRP B210 as its RF frontend.

The IoT device is an STM32L496 LTE IoT Cellular-to-Cloud Discovery Pack [STM21]. To connect to an operational network, it uses a SIM card from operator X and registers on X's NB-IoT networks. To connect to our Sonica testbed network, we insert a sysmoUSIM SIM card [sys16] and register its info on our server.

The attacker node targets both operational and testbed NB-IoT. We do not have access

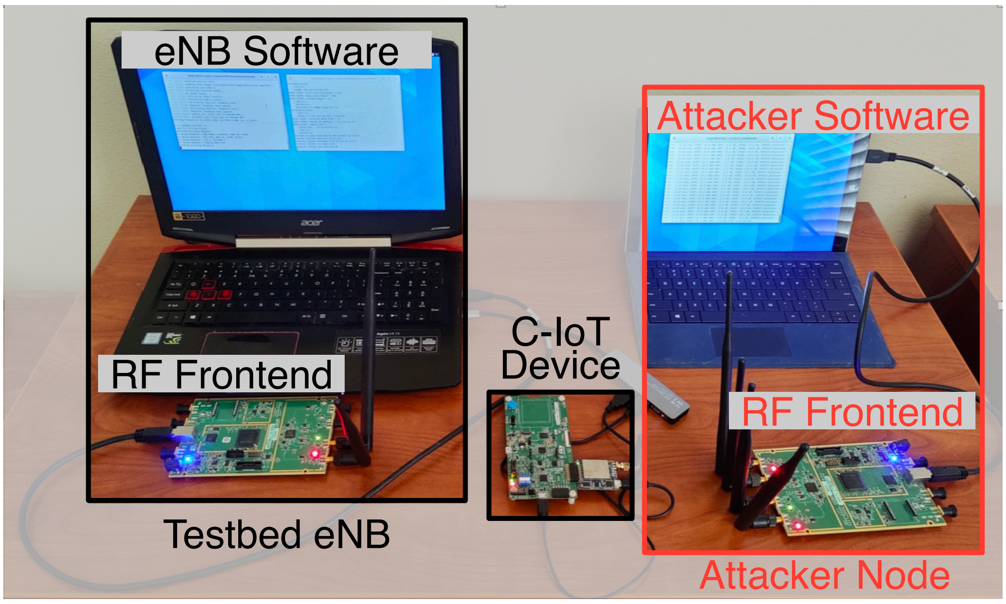
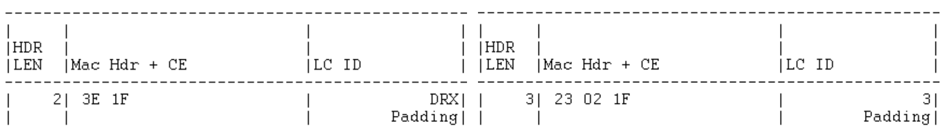


Figure 8.3: The testbed for attack validation.

to a Cat-M testbed, but the implementation could be adapted with similar components. For DL attacks, since the damage will only inflict damage on the victim, we validate the attacks with an operational network. As some UL attacks affect operational networks and users, we customize a private NB-IoT network. To set up an NB-IoT eNB, an Acer laptop with i7-7700HQ CPU and 16G RAM runs OpenAirInterface [Ope18] for eNB processing logic. It is connected to another USRP B210 for sending and receiving wireless signals.

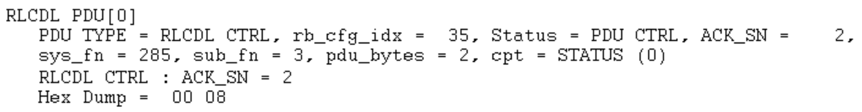
Attacker's Location Due to the limitation of the SDR-based attacker and testbed, we place the attacker node close to the victim to ensure sufficient relative power. We test our data forgery with two different locations. In location A, the attacker is 5 meters away from the victim in the same room. In location B, the attacker is separated by a wall in a different room from the victim. They are 15m away from each other. The testbed is placed in an indoor building on the third floor with NLOS condition. The settings are adapted from the previous work as in [YBS19].

Power Requirement To evaluate power requirement for successful data forgery, we adjust the USRP's transmission power at the attacker with different levels of relative power. We have tested with relative power levels of 3dB, 5dB, and 7dB.



```
--------------------------------------------- ----------------------------------------------
|     |                        |          | |     |                        |              |
|HDR  |                        |          | |HDR  |                        |              |
|LEN  |Mac Hdr + CE            |LC ID     | |LEN  |Mac Hdr + CE            |LC ID         |
--------------------------------------------- ----------------------------------------------
|    2| 3E 1F                  |       DRX| |    3| 23 02 1F               |             3|
|     |                        |   Padding| |     |                        |       Padding|
```

(a) MAC header of forged messages
Left: DRX MAC Command, Right: RLC Control ACK



```
RLCDL PDU[0]
   PDU TYPE = RLCDL CTRL, rb_cfg_idx =  35, Status = PDU CTRL, ACK_SN =    2,
   sys_fn = 285, sub_fn = 3, pdu_bytes = 2, cpt = STATUS (0)
   RLCDL CTRL : ACK_SN = 2
   Hex Dump =  00 08
```

(b) Payload of the forged RLC Control ACK above

Figure 8.4: MobileInsight log of forged messages.

Ethical Considerations Our attacker node is carefully managed during experiments. The DL signals only reach a few meters; we ensure no device other than our victim NB-IoT board is affected. We validate the UL attacks at our testbed, without affecting the operational networks. The attacker node cannot access any other C-IoT device. Mobile operators are also notified of our findings.

8.2.3 Evaluate Forgery Success Rate

We first validate a fake message can be successfully decoded and processed for both UL and DL. For DL, the attacker forges a MAC CE message and an RLC message. We demonstrate the victim device can receive both. The content of the forged MAC CE is `3E 1F 00 00 00`, where `3E` indicates a DRX command and `1F 00 00 00` are 4-byte paddings. On the NB-IoT board, we use the NB-IoT support in MobileInsight [LPY16] to collect fine-grained cellular logs. A MobileInsight log only includes messages that are *correctly decoded and processed*. As shown in Figure 8.4(a), the victim receives and accepts the forged MAC CE. The victim device correctly recognizes the DRX command in the forged message. We also test with message `23 02 1F 00 08`, where `23 02` means there is a 2-byte RLC message, `1F` is the padding, and `08` is the RLC control. As shown in Figure 8.4(b), the victim device accepts the RLC control message. It successfully recognizes the message `08` as an RLC ACK for packet SN 02. Similarly for UL, we send two messages to verify UL forge data: `3D 23 02 1F 00 00` (uplink BSR with padding) and `23 02 1F 00 08` for RLC control. In the eNB logs, both messages are decoded correctly.

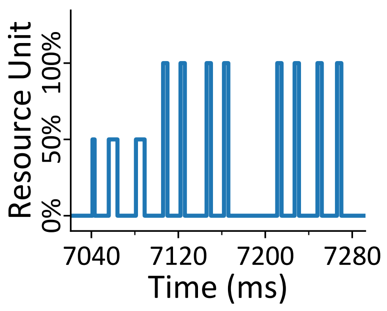
Table 8.1: Success rate of data forging.

Relative Power	3dB	5dB	7dB	Location A	Location B
DL	40.3%	75.8%	99.9%	99.1%	92.7%
DL (with DCI forgery)	32.1%	72.0%	99.8%	98.2%	90.0%
UL	41.2%	70.3%	99.8%	96.3%	94.1%

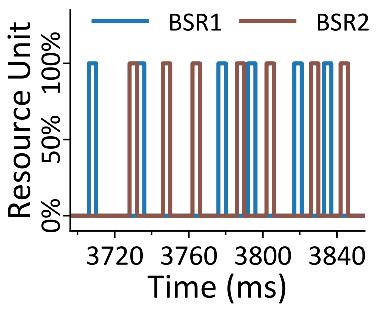
Message forging success rate We measure the success rate that the forged signaling is accepted. We present the obtained success rates in operational networks for DL forgery and at the testbed for UL forgery. For each setting, we forge 1000 data-plane signaling messages and count them in MobileInsight logs. The results are shown in Table 8.1. The legitimate data are fully blocked in all power levels greater than 3dB. Meanwhile, the success rate of decoding forged data is 40.3% at 3dB, 75.8% at 5 dB, and 99.9% at 7 dB for DL and 41.2% at 3dB, 70.3% at 5 dB, and 99.8% at 7 dB for UL. The success rate of DL forgery with fake DCI is 32.1% at 3dB, 72.0% at 5 dB, and 99.8% at 7 dB. This is lower compared to DL signaling message forgery with eavesdropped DCI, as there is a small probability of unsuccessful decoding of the forged DCI. For location-based testing, UL forgery has a high success rate with 96.3% in location A and 94.1% in location B. For DL, 99.1% and 92.7% (98.2% and 90.0% with forged DCI) of the signaling messages can be successfully decoded for location A and B, respectively. The attacker has a higher success rate at location A, where it is closer to the victim.

8.2.4 Validating Attacks with Data-Plane Signaling

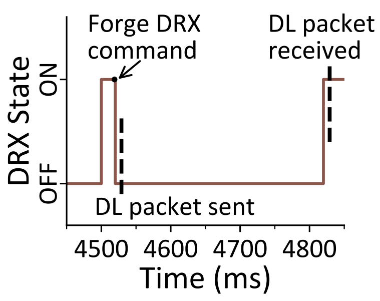
Next, we evaluate the damage and impact for each attack introduced in §4. The radio resource draining is validated in the testbed environment to avoid disrupting other devices in the cell. The flexible throughput limiting and connection reset attacks are also validated in our testbed because we cannot otherwise confirm the attack damages. The other attacks are validated in the operational network X. We place the attacker 5 meters away from the victim



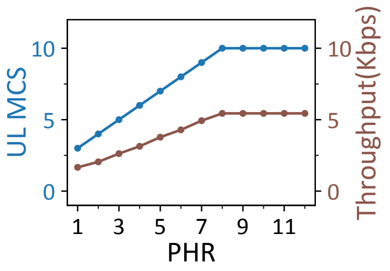
(a) The forged BSR wastes UL resource.



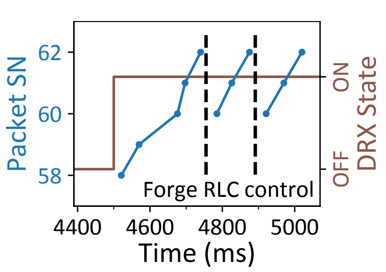
(b) Two forged BSRs drain UL resource.



(c) DRX command prolongs packet latency.



(d) PHR affects MCS and throughput.



(e) Forged RLC controls cause packet delivery loop.

Figure 8.5: Impact quantification of data-plane signaling attacks through experimental validation.

for each attack. For the device localization attack, the device is placed at three different, LOS outdoor locations. For all other attacks, the device is placed at an NLOS location in an indoor building on the third floor. Since we have validated the possibility of forging a data-plane signaling message with other settings, the attacks are also applicable in those scenarios.

Validate radio resource draining attack We demonstrate that a forged BSR can drain the radio resources. The fake BSR has a value of 31, which indicates 1KB UL data is pending. We measure the percentage of RU assigned to the fake BSR, divided by the total RU. Figure 8.5(a) illustrates how this ratio changes over time in response to a forged message. When the attack starts, the eNB schedules for this forged BSR for 200ms. During this period, the attacker completely occupies the channel for 32ms and uses 50% of the channel for 18ms. We also test how the eNB responds to two forged messages, shown in

Figure 8.5(b). Each BSR claims there is 1000B pending data. The attacker completely occupies the channel for 64ms during the next 180ms.

The forged BSR messages drain the resource for the other devices in the same cell. For a time period, the attacker occupies 100% UL resources. The damage could be even more severe if the attacker uses a larger BSR index or initiate frequent BSR messages. The attack applies to operational networks, as the eNBs that follow 3GPP standards should respond to BSR with sufficient grant. 4G Broadband can easily transfer 1KB UL data in one subframe [TZL21], however, it requires NB-IoT to consume most resources over hundreds of milliseconds.

Validate prolonged packet delivery attack We present an attack trace that demonstrates a DRX command can delay the DL data reception. As shown in Figure 8.5(c), the DRX ON state is supposed to finish at time 4540ms. However, a DRX command signaling message prematurely turns the device into DRX OFF state at time 4520ms. At time 4530ms, the eNB sends a DL packet. Since the victim is in the sleep mode (DRX OFF), it fails to receive the packet. The transmission cannot go through until the next DRX ON period at 4820ms. The latency is prolonged by more than 300 milliseconds in this experiment. This extra latency caused by the attacker is dominated by the periodicity of a DRX cycle time. In this operational network, the periodicity is 320ms. This value could be seconds in some operational networks.

Validate flexible throughput limiting attack We first validate the PHR attack messages are successfully accepted and decoded. Since existing codes for handling PHR in NB-IoT is incomplete, we use its broadband processing logic to derive the damage. When a PHR is received, the eNB adjusts the MCS for UL transfer. We draw the relationship between the MCS assignment and PHR in Figure 8.5(d). Assume the eNB assigns 3 resource units to the victim device, we also plot how PHR impacts the UL throughput. A forged PHR can reduce the throughput by 3.27x.

Validate device localization attack In this experiment, we demonstrate: 1) the attacker can locate the victim C-IoT device with TA, and 2) the inference is accurate as the

Table 8.2: Evaluation of Device Localization. Inferred area and error are average numbers in an area.

Area	Locations	# cells	Inferred area (m^2)	Error (m)
1	15	3	4475	119.4
2	10	3	9775	197.6
3	12	3	6900	99.1

real position of the device is close to the inferred area. We test three different areas in City A. The attacker first finds 3 local eNBs. It then forces the device to connect to them and eavesdrops on TA. The attacker succeeds in each attack within a minute. We record the user location with GPS as ground truth and compare it with the attack results, shown in Table 8.2. In each area, we localize the device as it moves to at least 10 different locations. The average area by inference is 4475-9775 m^2, and the device is 100-200m away from that result on average. To gauge the error, we sample the points in the inferred area and calculate the distance between the sampled spots and the device. The error stems from large TA coverage area, inaccurate TA incurred by interference, and imprecise eNB location, etc. The localization is less accurate compared to positioning techniques used for emergency calls in [Fis14,Bre15] (error of 50-100m). However, the attacker achieves so without actually connecting to or controlling the device.

Validate packet delivery loop We forge RLC control to force the device to repeatedly send the already accepted data. We show a trace from our experiment in Figure 8.5(e). At time 4740ms, the device sends packets with SN 62. Packets with SN 60-62 are not acknowledged. The attacker learns the SN and forges an RLC control with NACK at time 4754ms. The forged control message indicates packets 60-62 are not received at the eNB. The device thus retransmits them upon receiving the RLC control. The attacker repeats this message at time 4891ms. The packet sending forms a loop. Starting from 4500ms, the C-IoT device is kept in DRX ON period. The power-constraint C-IoT device thus consumes excessive energy for >500ms. The attacker can repeat the message to persistently drain the

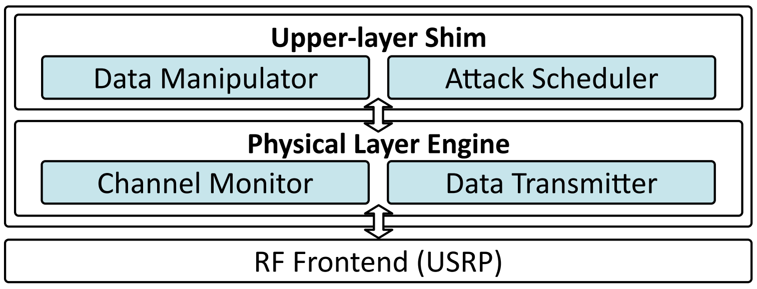


Figure 8.6: Implementation of FANE.

device's energy.

Validate connection reset attack We partially validate the attack. We first validate the eNB successfully decodes a forged RAI and the content is compliant with the standard. Unfortunately, we cannot directly observe the connection reset effect in our experiments, as both the operational network and our testbed run on release 14 and do not support release 16 message AS RAI. However, a standard-compliant eNB should terminate the connection based on the attack message.

Validate multicast disabling attack Similar to connection reset attack, this message, along with the multicast feature, is not supported by our COTS NB-IoT device. However, we partially validate the attack by confirming that the C-IoT device could successfully decode the SC-PTM Stop Indication signaling message. A standard-compliant C-IoT device should exit the multicast group based on the attack message, although the group is still active.

8.3 Evaluation of FANE

8.3.1 Implementing FANE

The overview of our implementation is shown in Figure 8.6. It contains two major parts: a Physical Layer Engine and an Upper-Layer Shim. The Physical Layer Engine communicates with the RF Frontend to monitor channels and transmit data for both UL and DL channels. The Upper-Layer Shim contains a simplified version of the 4G network protocol stack and attack-related logic.

The Physical Layer Engine camps to the target cell following a standardized procedure similar to a legitimate device. For each subframe (1ms interval), the Channel Monitor receives baseband signal from RF frontend, eavesdropping simultaneously on both UL and DL channels for control and data exchange between the victim device and the BS. Relevant control and data are handed to upper layer shim. Data Transmitter will send manipulated data for fake retransmission or inject attack signal notified by the Attack Scheduler.

Within the Upper-layer Shim, the Attack Scheduler inspects the control signals and decides the timing of sending noise or data. It keeps track of the necessary actions for multi-step attacks in a circular buffer. The attack sequence is recorded into the buffer at a triggering event, and each step is carried out according to the timing. The Data Manipulator is controlled by the Attack Scheduler. It receives commands issued from circular buffer. It handles the parsing and encapsulation of MAC/RLC/PDCP protocols. During the attack, it looks for the target data packet and applies manipulation.

The remainder of this section discusses two unique technical challenges to implementing Physical Layer Engine. Figure 8.7 illustrates its detailed handling logic.

Eavesdropping and Transmitting on UL/ DL Channels Our attack node needs to send and transmit data/noises on *both* UL and DL channels. This is different from either BS or device which only monitors channels in one direction and transmits in the other. This requirement poses challenges to both RF hardware, which needs to handle signal reception and transmission on UL and DL channels with different frequencies, and for physical layer processing.

Our solution is to implement the attack node based on carrier aggregation (CA) processing, where the DL and UL channels are regarded as "component carriers" and the former as the primary carrier for synchronization and alignment. This is shown in Figure 8.7. The PHY signaling processing mostly reuses and combines the existing logic for BS and device. However, some major changes are required because of the uniqueness of attack node. Most notably, we change the parameter "`rx_window_offset`" in BS handlers' FFT module. This is required to compensate for the difference in anchor of synchronization between BS and

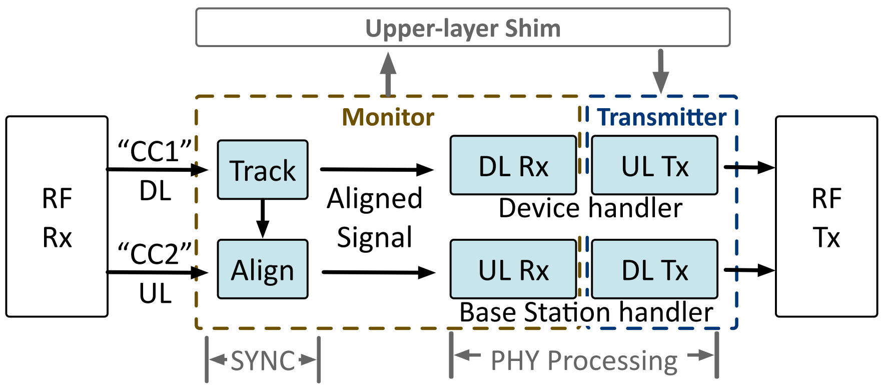


Figure 8.7: Details on Physical Layer Engine and its interaction with RF Frontend.

attacker. Besides, the timing to decode UL for a real BS is retrieved from its MAC layer scheduler. While in the attack node, the timing is obtained from Attack Scheduler by decoding UL grant.

Simultaneous Decoding and Corrupting An important attack step is to simultaneously decode (for attacker) and corrupt (for receiver) data. Although corrupting data at the receiver's side is not hard, we need to ensure the data can still be correctly decoded by the attacker for later manipulation. We use directional antennas for this purpose. They are capable of radiating greater power in specific directions. When sending noises, attacker can point the strongest direction (also known as the main lobe) to the receiving antenna of the BS or device, whose direction can be inferred from signals over-the-air. Meanwhile, the attacker puts its own receiving antenna in a direction least affected by the noises from directional antenna. In our testbed, the directional antenna provides around 25dB difference in signal strength between receivers from different directions. It can successfully achieve the goal of corrupting and decoding.

The attacker can also adopt other self-interference cancellation (SIC) techniques which are often used in in-band full-duplex wireless communication. The interference signal is fully known by the attacker and thus can be canceled out to a large degree. Existing systems [AE15,KFS18] are capable of providing more than 40dB separation between transmitter and receiver, which is enough for general usage in the wild.

Applicability to commercial networks Our FANE implementation can be readily adapted to attacking commercial networks. Our testing environment, which is the SDR-based testbed, is already standard-compliant and can serve off-the-shelf devices without modification. Only two main adaptations on hardware configurations are needed.

The first adaptation is to increase signal power to match the commercial BS. Corrupting the transmission requires reducing signal-to-noise ratio (SNR) in data channels. According to [LJL16], this only requires a comparable level of signal strength of victim devices. Thus, the requirement on signal strength and attacker positioning is much lower when compared to FBS attacks, which require orders of magnitude stronger signal to ensure the victim's stable connection [YBS19].

The other necessary adaptation for attacking commercial networks is to support MIMO. A pair of transceivers is needed for both UL and DL spatial streams. Our current FANE prototype with one USRP X300 (two transceivers) can only handle a single spatial stream. However, with more capable hardware (e.g., two synchronized USRPs with 4 transceivers), an attacker can implement MIMO decoding and encoding in Channel Monitor and Data Transmitter.

8.3.2 Building a Testbed for FANE

We construct a 4G LTE testbed (shown in Figure 8.8) for attack validation. A Surface Pro 3 runs processing for both BS and LTE core network using srsRAN [GGS16]. It connects to a USRP B210 for radio communication. The testbed is fully standard-compliant, a commodity device can connect to it and receive 4G data service without any software or hardware modification. We use a Pixel XL smartphone as the victim device, with a SIM card sysmoUSIM [sys16] registered on our core network. The attack node uses a USRP X300 with two CBX-120 daughterboards as its RF frontend. It is connected to a server with a 12-core Intel Xeon Silver 4214 CPU and 32GB RAM, running Ubuntu 18.04 as its OS. The server runs the implemented attack logic.

Ethical Considerations Our testbed is carefully controlled for its experiments, operating

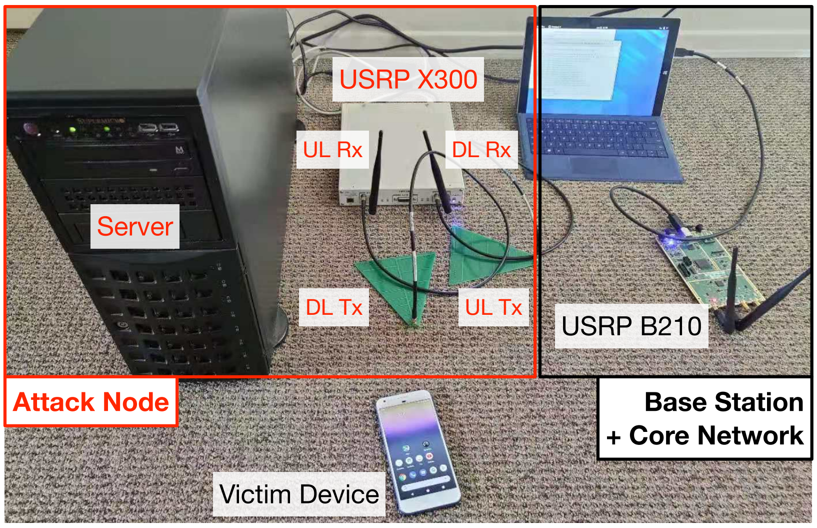


Figure 8.8: The testbed for attack verification.

on LTE Frequency Band 7 that remains unused by mobile carriers at our location. The radio signals only reach a few meters; no other device is affected or attempts to communicate with our testbed. Our malicious servers are not accessible to other users. Mobile operators are also being notified of our findings.

8.3.3 Forgery Success Rate

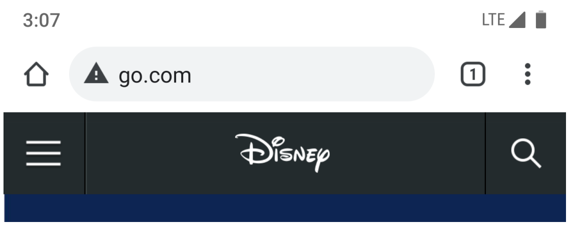
In this section, we assess the success rate of the forgery with `FANE` attack through experiments.

UL/DL data manipulation For each direction, the adversary targets a portion of ICMP echo (ping) packets between the victim device and a server, changing the TTL field for outbound or inbound packets. To compute the attack success rate, we collect IP traffic from our testbed (LTE core network and the victim device) and inspect packet content. The result is listed in Table 8.3. The physical-layer logs for the BS and the `FANE` are also used to analyze individual steps.

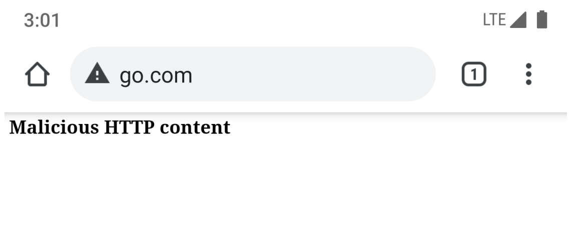
As shown in the table, UL attack has a high success rate. For 500 UL packets, 94.6% of them can be manipulated. The small failure rate is caused by failed decoding at the adversary or BS. The current attack scheme only manipulates the first retransmission. The

Table 8.3: Experiment results of UL/DL Attacks.

	# Packets	Corrupted	Decoded	Manipulated
UL	500	500 (100%)	491 (98.2%)	473 (94.6%)
DL	500	500 (100%)	487 (97.4%)	471 (94.2%)

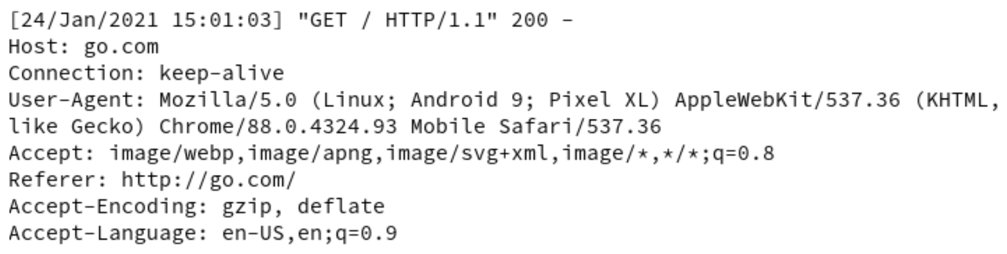


(a) Original content for go.com.



Malicious HTTP content

(b) `FANE` replaces the website.



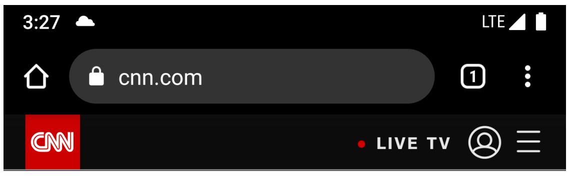
```
[24/Jan/2021 15:01:03] "GET / HTTP/1.1" 200 -
Host: go.com
Connection: keep-alive
User-Agent: Mozilla/5.0 (Linux; Android 9; Pixel XL) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/88.0.4324.93 Mobile Safari/537.36
Accept: image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Referer: http://go.com/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
```

(c) The malicious server receives the HTTP request.

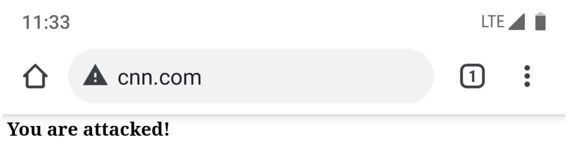
Figure 8.9: Attack validation on HTTP: Attacker intercepts an HTTP request and replaces website content.

success rate can be higher if the attacker enables multiple retransmissions upon decoding error.

DL manipulation also achieves a high success rate similar to the UL attack. For 500 DL packets, 94.2% of the packets can be successfully manipulated. Failed retransmission, along with some cases where `FANE` fails to corrupt some original DL transmission, causes unsuccessful manipulation.



(a) CNN website uses HTTPS.



You are attacked!

(b) FANE forces HTTP.

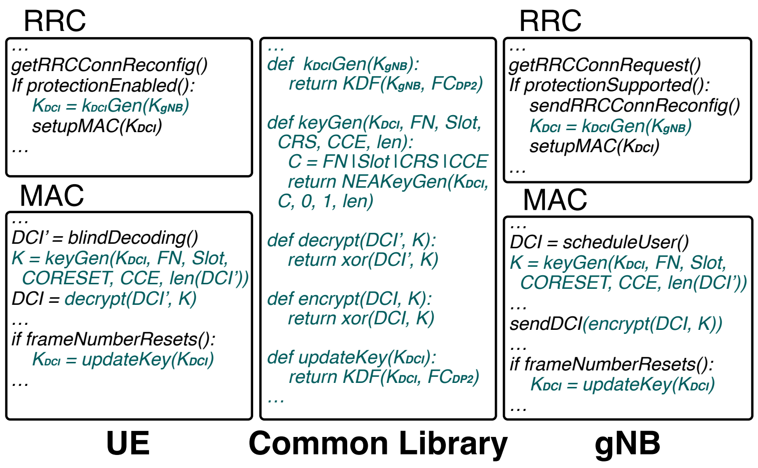
Figure 8.10: Attack validation: The adversary forces the use of HTTP and replaces the HTTPS website content.

8.3.4 Validating Data-Packet Attacks

HTTP Attacks Figure 8.9 illustrates a successful showcase of the attack. The adversary manages to infer the IP address of the authentic Web server, and redirects the HTTP request to the malicious Web server, which subsequently returns any content the adversary wishes the victim device to view. Despite a warning sign at the address bar, the browser at the victim device displays the fabricated content from the malicious Web server by default.

HTTPS Attacks Figure 8.10 illustrates a successful attack using the above procedures. Although the popular Web site cnn.com enables HTTPS, the adversary has forced the use of HTTP through the malicious Web server and redirect. The Web page is also replaced with any arbitrary content. This attack can be launched against any website without HSTS loaded into the browser (such as CNN, Fox News, CBS, ...). Despite a warning sign in the address bar, the browser still displays the insecure content by default.

IP Inference Attacks We also assess the IP Inference attack. In our experiment, we infer the IP address of the remote server based on periodic ICMP echo (ping) messages sent from the server with 400ms interval. This traffic is interleaved with other traffic. We use the packet size to single out the target messages. We let the adversary infer 20 distinct server IP addresses that are in different address spaces. Our experiment shows that the adversary correctly infers all of them. On average, the adversary only needs 33.85 attempts (step ①-③ in Figure 4.6) to infer an unknown IP address.



```
...
getRRCConnReconfig()
If protectionEnabled():
    KDCI = kDCIGen(KgNB)
    setupMAC(KDCI)
...
```

```
...
DCI' = blindDecoding()
K = keyGen(KDCI, FN, Slot,
  CORESET, CCE, len(DCI'))
DCI = decrypt(DCI', K)
...
if frameNumberResets():
    KDCI = updateKey(KDCI)
...
```

```
...
def  kDCIGen(KgNB):
    return KDF(KgNB, FCDP2)

def keyGen(KDCI, FN, Slot,
  CRS, CCE, len):
    C = FN|Slot|CRS|CCE
    return NEAKeyGen(KDCI,
  C, 0, 1, len)

def decrypt(DCI', K):
    return xor(DCI', K)

def encrypt(DCI, K):
    return xor(DCI, K)

def updateKey(KDCI):
    return KDF(KDCI, FCDP2)
...
```

```
...
getRRCConnRequest()
If protectionSupported():
    sendRRCConnReconfig()
    KDCI = kDCIGen(KgNB)
    setupMAC(KDCI)
...
```

```
...
DCI = scheduleUser()
K = keyGen(KDCI, FN, Slot,
  CORESET, CCE, len(DCI'))
...
sendDCI(encrypt(DCI, K))
...
if frameNumberResets():
    KDCI = updateKey(KDCI)
...
```

Figure 8.11: Implementation of DP^2 and its APIs.

8.4 Evaluation of DP^2

8.4.1 Implementation

We implement DP^2 as a C library. The implementation consists of three major components. First, we support DCI meta-info protection with DP^2. The library leverages the NEA algorithm implementation in srsRAN [GGS16] to generate the keystream. The inputs include the session key K_{DCI}, the synchronized time clocks (FN and slot), the frequency index (CORESET + CCE index), the length of the DCI, and the constant bearer ID and direction bit. The key generation follows our design in §5 and can be called immediately after scheduling. Two additional APIs are provided to encrypt and decrypt the DCI content with the keystream. Second, the library supports DCI key update when FN resets to prevent key reuse. We use the standardized KDF function on the previous key K'_{DCI} to update it. Finally, we modify the RRC protocol to setup the MAC layer scheduling protection. If the UE does not indicate that DP^2 is supported, the gNB fallbacks to the legacy operational mode. Otherwise, when RRC connection is established, both gNB and UE calculate the key K_{DCI} using K_{gNB}. RRC on both sides share the key with their MAC layers.

The entire library is lightweight as it reuses the existing 5G NEA algorithms. It consists of 156 lines of code. A gNB and UE can leverage the well-packaged APIs to quickly set

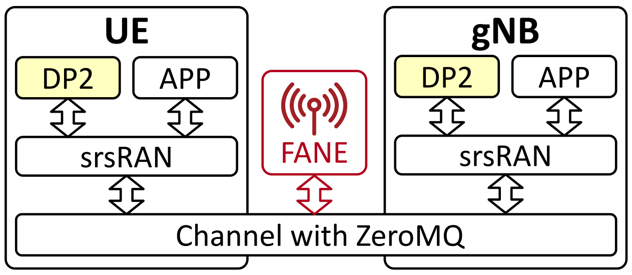


Figure 8.12: Testbed setup for attack and DP^2 protection.

up DP^2 for protecting data-plane. After scheduling, gNB MAC can make an API call to pre-generate the keystream for obfuscating the DCI information later. The pseudocode and example calls to our DP^2 library are shown in Figure 8.11.

8.4.2 Evaluation

Testbed Figure 8.12 illustrates our setup for 5G emulation. We evaluate both attacks and defense using Flora Testbed installing 5G SDR protocol stack srsRAN [GGS16]. It is a standard-compliant implementation of 5G RAN. The UE and gNB run as two processes on a server with 8-core 16-thread AMD Ryzen 7 5800X CPU and 32GB RAM. The server runs Ubuntu 20.04 as the OS. Due to licensing and legal concerns, we use ZeroMQ [Zer22] for emulating the 5G physical channels. It runs on an emulated physical channel with bandwidth of 10MHz. Note that the current srsRAN 5G only supports 10MHz, we also emulate a channel of 5MHz by limiting the gNB to only use half of the RBs when assigning wireless channel resources. Given that the protocol stack is 5G standard-compliant, our testbed could work with commercial off-the-shelf devices if we incorporate USRP boards as frontend and send/receive signals on physical wireless channels.

Ethical Considerations This testbed does not raise any ethical issues. The attack and protection are only visible in our own testbed, without affecting any real UE or gNB. This is because the testbed emulates wireless channel with ZeroMQ without real signals. We are working with our collaborating mobile operators and plan to submit proposals to 3GPP.

Table 8.4: Data throughput (Mbps) with and without DP^2. DP^2 does not affect data throughput in all scenarios.

Pkt Size	BW	UL Traffic		DL Traffic	
		Legacy	DP^2	Legacy	DP^2
1000 B	10 MHz	16.85	16.85	29.50	29.50
100 B	10 MHz	16.67	16.84	29.00	29.01
1000 B	5 MHz	12.49	12.47	14.27	14.27
100 B	5 MHz	12.54	12.44	14.29	14.27

8.4.2.1 Evaluate DP^2 Protection

We now evaluate how well DP^2 secures data plane with meta-info protection. We evaluate DP^2 from two aspects. First, we show its correctness, as meta-info encryption and decryption can be finished in the required time without affecting meta-info processing or data transmission. Second, we also measure and demonstrate that DP^2 incurs small overhead compared to other approaches for each packet/signaling message.

Correctness To validate the correctness of DP^2, we compare the throughput on the application layer of legacy 5G UE/gNB and our solution that incorporates DP^2. We run iPerf3 [Ipe] to saturate the radio link between gNB and the UE and measure the throughput. We test both uplink traffic (where we run iPerf3 client on UE and iPerf3 server on gNB) and downlink traffic (where we run iPerf3 client on gNB and iPerf3 server on UE).

Per our design, the processing on MAC layer DCI meta-info can be finished along with other existing operations before the deadline (i.e., within the current time slot), and thus do not affect the throughput at all. Therefore, the data transmission in the application layer shall not be affected by processing on meta-info in DP^2. For this purpose, we evaluate two different packet types: small packets of 100B and large packets of 1000B, respectively. For each packet size, we adjust the bandwidth to 10MHz and 5MHz for testing. We repeat each setting for 3 times and calculate the average numbers.

Table 8.5: Extra processing on each DCI with DP^2.

Pkt Size	BW	UL Traffic		DL Traffic	
		UE	gNB	UE	gNB
1000 B	10 MHz	0.77%	1.02%	0.64%	2.42%
100 B	10 MHz	1.02%	1.14%	0.90%	2.51%
1000 B	5 MHz	0.91%	1.28%	0.84%	2.51%
100 B	5 MHz	1.13%	1.48%	1.03%	2.57%

We validate the correctness of DCI protection. The decryption achieves a 100% success rate, regardless of retransmission, channel condition, or DCI type. This is due to the fact that both sides can use synchronized information for generating keys for encryption and decryption. In addition, the extra operation on meta-info does not affect data transmission. As shown in Table 8.4, the UL throughput is 16.85→16.85 Mbps for 1000B/10MHz, 16.67→16.84 Mbps for 100B/10MHz, 12.49→12.47 Mbps for 1000B/5MHz, and 12.54→12.44 Mbps for 1000B/5MHz. The DL throughput is 29.50→29.50 Mbps for 1000B/10MHz, 29.00→29.01 Mbps for 100B/10MHz, 14.27→14.27 Mbps for 1000B/5MHz, and 14.29→14.27 Mbps for 100B/5MHz. Note that downlink throughput is higher than uplink, as uplink requires extra request-to-scheduling time. The throughput is not affected for all scenarios, as each DCI can be protected within a single time slot without being delayed. UE and gNB can integrate DP^2 without affecting any data transmission.

Note that, the throughput achievable in our testbed is smaller compared to what is allowed by 5G. This is due to the lack of MIMO/carrier-aggregation support, the constrained processing capability, and the relatively narrower band used in srsRAN 5G and our testbed. In commercial networks, despite a higher throughput, the commercial gNB and device will have much higher computing capability than our software-based server implementation. Therefore, DP^2 shall still affect no data transmission in such environment.

Low overhead of DP^2 We then evaluate the overhead of our solution. We first evaluate the extra processing overhead for each DCI. We compare the processing overhead of DP^2 on each DCI processing with vanilla 5G without any extra protection. Different settings are tested in this experiment same as described in our correctness experiments. The results are shown in Table 8.5. In our experiments, $\mathtt{DP}^2$ only incurs 0.64-2.57% overhead on each DCI on average. The extra overhead for each DCI on the UE side is small, as the UE needs to perform the time-consuming blind-decoding procedure. Note that, the results do not include the costs that a UE finds no DCI through blind decoding, where the efforts are wasted. If we consider this overhead, the per-DCI extra processing for $\mathtt{DP}^2$ will be even smaller. Meanwhile, the overhead is slightly larger on gNB but still considered marginal with its strong capability. This small overhead explains why the throughput is not affected by the extra operations. Therefore, the current gNB can support $\mathtt{DP}^2$ without upgrading its hardware and can still satisfy the requirement for data plane security.

We next compare the solution overhead of meta-info protection on each data packet with other alternative solutions. For baseline purpose, we compare $\mathtt{DP}^2$ to three alternative solutions: 1) Legacy 5G without integrity protection; 2) 5G with optional data packet encryption; 3) 5G with enforced encryption and integrity protection on all data-plane packets and signaling messages. For evaluating the solution overhead, we compare both time and data overhead for processing each data packet. Note that the processing on DCI is of lower frequency than data packets or data-plane signaling, so we amortize the overhead of $\mathtt{DP}^2$ for each data packet.

We show the evaluation results of processing overhead from our experiments in Table 8.6. We let the processing time for our security design $\mathtt{DP}^2$ be the baseline of 1.0X and calculate the relative overhead of the alternative solutions in comparison. For 4 different traffic/packet settings with UL traffic, the overhead is 15.6-40.1$\times$ on UE and 16.4-35.1$\times$ on gNB for legacy 5G, 31.8-79.3$\times$ on UE and 33.0-79.6$\times$ on gNB for 5G data-plane encryption and integrity protection, and 32.0-79.7$\times$ on UE and 33.2-80.0$\times$ on gNB for encrypting and integrity protecting all data and signaling. On DL traffic testing, the overhead is 7.9-15.2$\times$ on UE and 7.6-14.8$\times$ on gNB for legacy 5G, 16.2-30.4$\times$ on UE and 15.3-29.5$\times$ on gNB for 5G data-plane encryption and integrity protection, and 16.4-30.5$\times$ on UE and 15.5-29.7$\times$ on gNB for encrypting and integrity protecting all data and signaling. Among all three alternative

Table 8.6: Processing and data overhead of DP^2 compared to alternative solutions. The overhead is amortized to every packet.

Setting	Solution	Uplink Traffic			Downlink Traffic		
		Processing		Data	Processing		Data
		UE	gNB		UE	gNB	
1000B/10MHz	DP^2	1.0×	1.0×	0 B	1.0×	1.0×	0 B
	Legacy	37.5×	35.1×	0 B	15.2×	12.7×	0 B
	Legacy + Pkt Integrity	75.7×	79.6×	4 B	30.4×	28.4×	4 B
	Pkt/Sig Enc+Integrity	76.0×	80.0×	4.15 B	30.5×	28.6×	4.19 B
100B/10MHz	DP^2	1.0×	1.0×	0 B	1.0×	1.0×	0 B
	Legacy	40.1×	31.1×	0 B	13.0×	14.8×	0 B
	Legacy + Pkt Integrity	79.3×	70.8×	4 B	26.8×	29.5×	4 B
	Pkt/Sig Enc+Integrity	79.7×	71.2×	4.14 B	30.0×	29.7×	4.19 B
1000B/5MHz	DP^2	1.0×	1.0×	0 B	1.0×	1.0×	0 B
	Legacy	15.6×	17.2×	0 B	7.9×	7.7×	0 B
	Legacy + Pkt Integrity	31.8×	33.8×	4 B	16.2×	15.5×	4 B
	Pkt/Sig Enc+Integrity	32.0×	34.1×	4.23 B	16.4×	15.7×	4.39 B
100B/5MHz	DP^2	1.0×	1.0×	0 B	1.0×	1.0×	0 B
	Legacy	18.6×	16.4×	0 B	9.1×	7.6×	0 B
	Legacy + Pkt Integrity	36.9×	33.0×	4 B	17.9×	15.3×	4 B
	Pkt/Sig Enc+Integrity	37.1×	33.2×	4.23 B	18.0×	15.5×	4.38 B

solutions, only protecting all data (data-plane signaling + packets) will prevent FANE and CDS attacks as DP^2. Yet, it consumes the highest overhead that makes it an unsuitable solution for the current 5G. Meanwhile, DP^2 only incurs small amortized overhead, given our design and the low frequency/short length of the DCI meta-info for encryption. We further note one thing from our experiments. DP^2 incurs more overhead in the downlink than in the uplink. Higher CPU load is needed for downlink data delivery due to higher throughput. Therefore, the processing of DCI in MAC is slower. Still, DP^2 can reduce processing overhead by up to 30.5×.

We compare the data overhead of DP^2 with alternative solutions in different settings, as shown in Table 8.6. Since DP^2 does not require integrity check on meta-info or any data, it

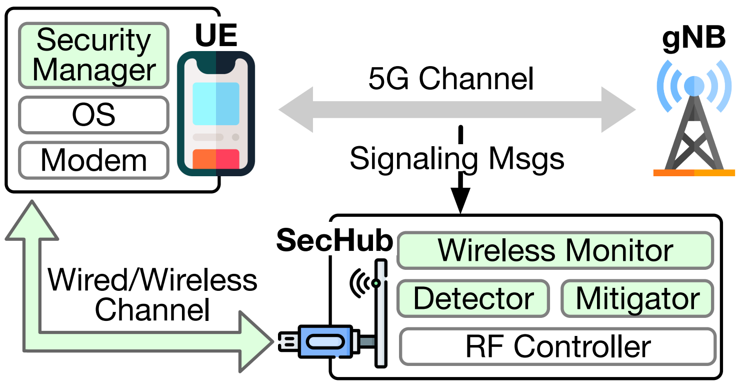


Figure 8.13: Implementation of `CellDAM`. Green blocks are `CellDAM` modules.

results in no data overhead per packet as the legacy solution without any integrity protection. Meanwhile, the solution that includes integrity protection on either data plane packets or signaling will incur high data overhead. If only data-plane packets are integrity protected, the extra data overhead is 4B per packet. If both packets and data-plane signaling are integrity protected, the extra data overhead could be 4.4 per packet. Consequently, $\mathtt{DP}^2$ is a solution that provides "implicit integrity protection" to prevent data-plane forgery attacks without incurring extra data overhead.

8.5 Evaluation of `CellDAM`

8.5.1 Implementation

We implement `CellDAM` as shown in Figure 8.13. `SecHub` performs attack detection and mitigation, and a security manager app on UE facilitates `SecHub` for rootless signaling capture. We next elaborate on each component.

Wireless Monitor A wireless monitor is deployed based on srsRAN [SRS] to perform the rootless signaling capture through the RF controller with the SDR devices. We implement it with 2,794 lines of C++ code. The monitor collaborates with the UE to camp on the target cell, infer the UE's C-RNTI, and simultaneously capture real-time messages on both the uplink and downlink channels. After the capture, it decodes the signaling messages accordingly and passes them to the attack detector.

Detector with state-dependent model checking We implement the state-dependent model checking attack detector with 1,252 lines of Python. The real-time traces from the wireless monitor will be fed into the detector for undesired behavior and potential attacks. If any consecutive signaling messages violate the cross-layer model checking, potential attacks will be reported by the detector. The detector will further notify the mitigator module to trigger the mitigation.

Mitigator We deploy the mitigator with 860 lines of Python code. After detecting any attacks, the mitigator triggers the victim handover. With the current signal conditions acquired from the security manager application from the victim, the mitigator calculates the minimum transmit power to trigger the UE handover with the SDR devices. It then notifies the RF controller to initiate signals targeting the victim UE. This will trigger handover to escape from the attacker's cell.

Security Manager App On the phone side, we deploy a security manager application with 1,264 lines of Java. The application monitors the current band, PCI, and signal conditions (RSRP and RSRQ) with the Android Telephony API [APIb] to facilitate the rootless signaling capture. It also generates a corresponding UDP traffic after receiving the traffic interval from `SecHub` for the collaborated traffic fingerprinting. It supports exchanging the data with `SecHub` through a wired (USB) or wireless (Bluetooth in the current implementation) connection leveraging Android APIs [APIa]. It supports the X.509 certificate to facilitate the mutual authentication and encryption between the app and `SecHub`. We also implement an equivalent application for srsUE running on user-space. The same set of information is extracted from the srsUE by hooking the current srsUE functions. Then the information is shared to the `SecHub` with the socket API.

8.5.2 Evaluation Setup

Testbed Setup We construct a testbed for experimental validation (Figure 8.14). The gNB and UE are built upon Flora with srsRAN using a 5G protocol stack. The physical layer encoding is still kept with 4G due to current hardware limitations. The gNB software

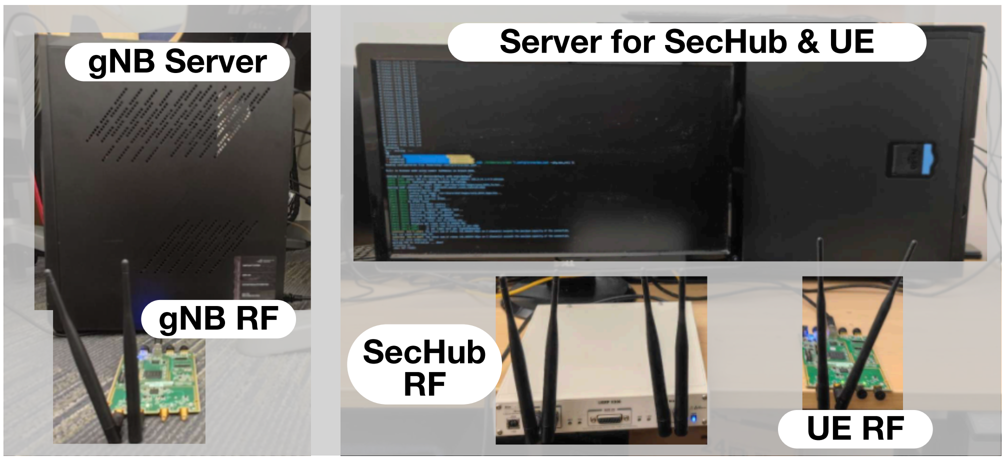


Figure 8.14: Testbed Setup for `CellDAM`.

is run on an i7-9700K PC with Ubuntu 20.04. The UE runs on an Intel Xeon Silver 4214 server running Ubuntu 20.04. Both use USRP B210 as their RF frontend, with the frequency set to an unlicensed 2.4GHz ISM band. `SecHub` is co-located on the same server as the UE and uses USRP X300 as the RF frontend. The security manager application runs on the same server as a user-space process and shares the information extracted from srsUE with `SecHub`. The mobile version of security manager application is tested on a Pixel 4a with Snapdragon 730 running Android 12.

Attack Reproduction All attacks listed in Table 6.2 are recreated within the testbed in order to evaluate `CellDAM`'s ability of attack detection and mitigation. According to §6.4, the attacks can be classified into two categories: with or without the usage of relay FBS. We support both types of attacks with partial software emulation on our testbed for controllability and reproducibility while still achieving effects equivalent to real attacks.

In attacks with relay FBS, we set up both the FBS and the real gNB in the testbed. We emulate the radio link between relay FBS and real gNB in software with ZeroMQ [Zer22] to avoid interference with the link between UE and relay FBS, which uses physical link with USRP.

On the other hand, the attacks without relay FBS rely on manipulation of the underlying physical channel. We emulate the attacker using a separate thread within the gNB and UE program, which has access to the transmitting and receiving signal buffer. Eavesdropping is realized with inspection of the receiving buffer, while the forging or corruption attacks are emulated by injecting encoded attack messages or noise to the transmitting buffer.

Ethical Considerations The evaluation does not raise any ethical issues. Our testbed is carefully controlled for experiments, operating on an unlicensed 2.4GHz ISM band. The experimentation is conducted within a 5MHz channel centered at 2.49GHz. We ensure no nearby device is using the frequency band. The radio signal emitted by the testbed only reaches a few meters, ensuring that no other device is affected or attempts to communicate with our testbed. Meanwhile, we are working with collaborating mobile operators about the discovered solutions and will open source `CellDAM`.

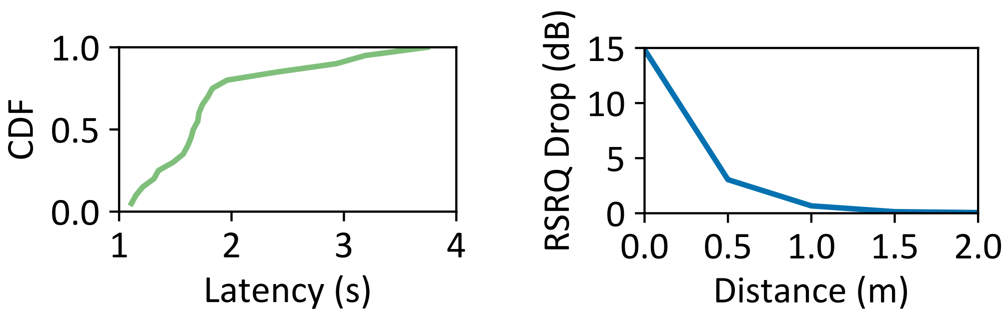
8.5.3 Evaluation Results

Evaluation of `CellDAM` Attack Detection In this subsection, we answer the question of whether `CellDAM` can detect all attacks displayed in Table 6.2. For this purpose, we inject attack messages according to the attacker procedure, and observe if the detector can initiate warnings as expected. We test the attack detection under two different traffic types: A lightweight traffic with ping and a heavy traffic with iPerf3 [Ipe] to saturate the channel. For each attack, we repeat 1,000 times under each scenario. We evaluate the results with three metrics: precision, recall, and F1 score.

Table 8.7 summarizes the precision, recall, and F1 score for the `CellDAM` detection. As shown in the table, the detection reaches a precision of 98.9%∼100% for different attack types. The high precision is achieved by the correctness of the DFA and verification, as the normal operations in legitimate 5G delivery will follow the correct procedure. Meanwhile, the recall is 70.5%∼100% for different attack detection. Note that, the relatively low recall is because a heavy-traffic scenario will extend the ON state for a device. The device can use other FBS detection methods [AF19, ZJZ18] to complement `CellDAM`. The recall is high for other attacks, as they will incur undesired data-plane signaling. This results in a high F1 score of 0.823∼0.999. The detection works well for both light and heavy traffic. This is because `CellDAM` only requires inspection on lightweight data-plane signaling messages.

Table 8.7: Effectiveness of attack detection with `CellDAM`.

Attack	A1	A2	A3	A4	A5	A6	A7	A8
Precision	0.999	1	0.996	0.989	1	1	0.996	0.999
Recall	1	0.989	1	0.705	0.976	1	1	1
F1	0.999	0.994	0.998	0.823	0.988	1	0.998	0.999



(a) CDF of the mitigation latency for "quick dodge". (b) Relationship between the distance and RSRQ drop.

Figure 8.15: Mitigation Performance.

We also measure the average detection latency for attack detection. The signaling messages could be captured and fed into the detector for real-time detection. The detection achieves an average latency of 28ms. Therefore, `CellDAM` can quickly spot potential adversaries and take action.

Evaluation of `CellDAM` Mitigation We evaluate the success rate for mitigation. When `SecHub` detects any attack, it will trigger the target device handover to escape from the attack. We enable the handover-related RRC messages and config in gNB and let the UE monitor the handover events. Note that our testbed does not support real handover to a different cell; if UE receives a handover command, we assume a successful handover. We evaluate the ratio of successful handover and the corresponding latency for each mitigation. In all 40 rounds of experiments, `SecHub` successfully triggers the UE handover. Figure 8.15(a) shows the CDF of the latency. The results show that the average mitigation latency is 1.85s.

Table 8.8: Success ratio of rootless signaling message capture under different traffic scenarios.

Traffic	Control		Data	
	PUCCH	PDCCH	PUSCH	PDSCH
Light	99.1%	100%	100%	99.7%
Heavy	98.7%	99.9%	98.1%	97.2%

90% of them could successfully handover within 3 seconds, and all handovers are triggered within 4 seconds.

In our design, signals from `SecHub` will have minimum impact on other devices. We show this point by measuring the perceived RSRQ drops at the UE at different distances from the `SecHub`. Figure 8.15(b) shows that the RSRQ drop at 1m and 2m are only 0.67dB and 0.06dB, respectively. With the controlled power of `SecHub`, our mitigation will only trigger handover on the close-by protected device, while not affecting other users or devices.

Evaluation of `SecHub` Rootless Capture We then present our microbenchmarks for inferring signaling messages. We measure the ratio of correctly captured signaling in both uplink and downlink to show the effectiveness of our rootless signaling capture. We record the traces of PUCCH/PDCCH (SR and DCI) and PDSCH/PUSCH (MAC CE and RLC Control) at the base station side as the ground truth. We capture the traces through the `SecHub` under different traffic scenarios. We use ping and iPerf3 for the light and heavy traffic scenarios, respectively. The ratio of correctly decoded signaling is calculated by comparing the traces captured on the `SecHub` and the ground truth.

Table 8.8 shows the success ratio for the rootless capture. For the control messages, 99.1% of PUCCH and 100% of PDCCH are successfully captured and decoded from the `SecHub` with the light traffic. For the heavy traffic, the `SecHub` still achieves a high success rate with 98.7% for PUCCH and 99.9% for the PDCCH. For data messages, `SecHub` successfully decodes all the PUSCH messages and 99.7% of PDSCH data under the light traffic scenario. 98.1% of the PUSCH and 97.2% of the PDSCH traffic is successfully captured and decoded for the heavy traffic scenario.

A high success rate is possible with accurate C-RNTI inference. We quantify the success rate with the collaborated traffic fingerprinting. Since the inference can be done without actively sending signals over the air, we perform the verification on both the commercial network and our testbed. The ground truth of C-RNTI can be acquired in gNB (by checking logs) and in COTS UE (by using MobileInsight [LPY16]). Every time `SecHub` collects 5s of traces for the inference after the traffic pattern coordination with the target UE. We perform 120 rounds of experiments with 60 rounds on the testbed and 60 rounds on the commercial network.

On the testbed, `SecHub` correctly infers the C-RNTI in all 60 rounds, achieving a 100% success rate. On the commercial network, with the increased device number, `SecHub` successfully infers 98.3% of the C-RNTI. We further measure the overheads caused by the background ping in the continuous tracking. The result shows that `CellDAM` involves 0.14 KBps traffic overhead, which is marginal on the target device. The results show that `CellDAM` could continuously perform the monitoring during the mobility, and protect the victim without any root privilege.

8.6 Implement and Evaluate `LRP`

8.6.1 Implementing `LRP`

We implement `LRP` as a standalone user-space daemon with Android NDK. A similar implementation is also feasible for iOS. Figure 8.16 shows its key components, including a latency manager for latency reduction with conflict resolution in §7.3.1–7.3.3, an inference engine that offers key parameters for `LRP` based on the solutions in §7.4, and a set of APIs for latency-sensitive applications. To use `LRP`, a latency-sensitive mobile app requests `LRP` service using its APIs as detailed below. At runtime, `LRP` first detects if the device connects to a new base station by checking the change of serving cell ID. Upon cell changes, `LRP` starts to infer the key LTE parameters for this new cell. Once the key parameters are obtained, `LRP`

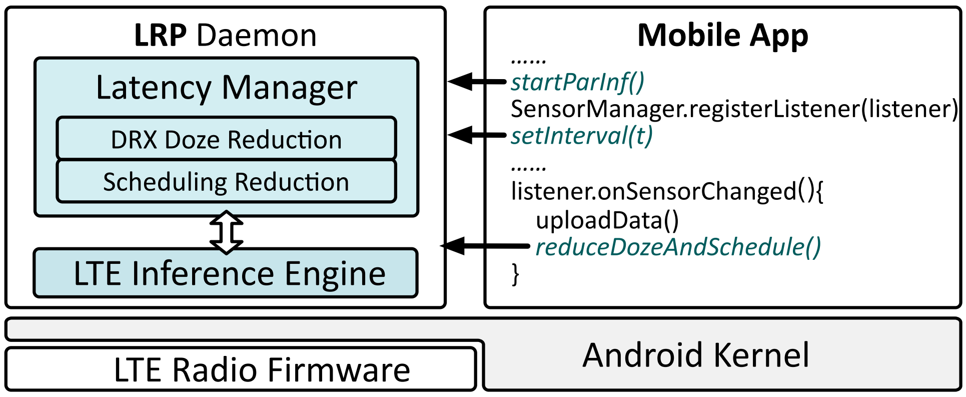


Figure 8.16: Implementation of `LRP` in Android.

initiates its latency manager to reduce DRX doze latency in §7.3.1 and scheduling latency in §7.3.2, and resolves the conflicts in §7.3.3.

APIs `LRP` provides easy-to-use application-layer APIs for mobile application developers. Figure 8.16 showcases these APIs with a mobile VR application. The app first calls startParInf() so that `LRP` daemon starts and infers the LTE parameters relevant to latency reduction components. The daemon detects possible parameter changes (say, upon handover) and re-runs the inference procedure whenever necessary. As our VR application uploads periodic sensory data packets, it calls setInterval(t) to inform `LRP` such periodicity. Whenever a data packet is sent, the application calls reduceDozeAndSchedule() for `LRP` to reduce latency for the next packet.

Latency manager It realizes the latency reduction in §7.3.1–7.3.2 and conflict resolution in §7.3.3. A practical issue to realize them is to optimize the dummy packet's construction and delivery for low cost. Both *prefetcher* and *rouser* messages in `LRP` should be as small as possible so that extra data overhead is minimized. In addition, a smaller *prefetcher* will decrease the likelihood of the corner case discussed in §7.3.2. The smallest packet we could generate in the Android device without root is an ICMP ping packet with IP header only via system command. Our implementation issues only one small ICMP packet to the local gateway in LTE that serves the users.

LTE inference engine It infers the key LTE parameters for `LRP`'s latency reduction based on the approaches in §7.4. We use DNS requests/responses as probing packets, which have low deployment cost (by using LTE's readily-available DNS servers) and higher accuracy (compared to other probing packets delivered with low priority such as ICMP). For DNS servers, LTE assigns its own in-network DNS server when the device attaches to it, which provides fast and stable service. We use such DNS servers for our experiment.

Moreover, we note that simply running the inference in §7.4 may be inaccurate in practice, since it is sensitive to the noises from background traffic, vendor-specific base station behaviors, and server load. To this end, we optimize our implementation to mitigate these noises and improve the inference accuracy. Specifically, we add a few filters to get rid of the noises. For instance, when measuring the scheduling-related parameters, we know that T_{rtt} should be greater than 4ms in reality, therefore, if the packet response pair is received within 4ms, we ignore this round of experiment.

We assess how LRP improves the overall latencies and QoEs for emergent mobile applications, evaluate the effectiveness of solution components in LRP, and quantify LRP's overhead.

Experimental setup We run LRP on Google Pixel, Pixel 2, Pixel XL, and Pixel 5. We quantify the latency reduction in both US and China over AT&T, Verizon, T-Mobile, Sprint, and China Mobile. The evaluation covers 375 unique cells. We repeat the tests in static, walking (∼1m/s), and driving (∼30mph) scenarios. We do experiments mostly in metropolitan areas while driving tests cover rural areas as well. The radio signal strength varies from -120 to -80dBm, covering good (>-90dBm), fair ([-105, -90dBm]), and bad (<-105dBm) conditions. To quantify LRP's latency reduction, we use MobileInsight [LPY16] to extract the ground truth of fine-grained per-packet latency breakdown from the chipset.

To gauge LRP's impact on the network side, we build a USRP-based testbed. A server with Intel i7-9700k CPU and 32G RAM runs srsRAN [GGS16] for the functions of core network and BS processing. A USRP B210 connects to the server and provides wireless access for the devices. We plug sysmoUSIM [sys16] into the test phones, and register them.

Table 8.9: Uplink network latency (ms) reduction by LRP in evaluations with four apps. η=(Legacy-LRP)/LRP.

App		AT&T			Verizon			T-Mobile			Sprint			China Mobile		
		Leg	LRP	η	Leg.	LRP	η	Leg.	LRP	η	Leg.	LRP	η	Leg.	LRP	η
Mobile VR	Med.	N/A	N/A	N/A	12.0	8.0	0.5×	11.0	6.0	0.8×	N/A	N/A	N/A	N/A	N/A	N/A
	95%	N/A	N/A	N/A	28.0	15.0	0.9×	40.0	14.0	1.9×	N/A	N/A	N/A	N/A	N/A	N/A
Gaming	Med.	10.0	6.0	0.7×	9.0	7.0	0.3×	9.0	7.0	0.3×	17.0	11.0	0.5×	4.0	3.0	0.3×
	95%	17.0	15.0	0.1×	15.0	15.0	0×	15.0	15.0	0×	27.0	21.0	0.3×	10.0	5.0	1.0×
Localization	Med.	38.0	5.0	6.6×	50.0	14.0	2.6×	42.0	5.0	7.4×	30.5	14.0	1.2×	11.0	3.5	2.1×
	95%	46.0	14.0	2.3×	59.0	23.0	1.6×	48.0	10.0	3.8×	61.7	25.8	1.4×	22.0	6.0	2.7×
Object Detection	Med.	23.0	7.0	2.3×	38.0	9.0	3.2×	33.0	5.0	5.6×	30.0	15.0	1.0×	14.0	6.0	1.3×
	95%	47.8	16.0	2.0×	51.0	15.3	2.3×	45.0	10.0	3.5×	59.0	27.5	1.1×	22.0	17.0	0.3×

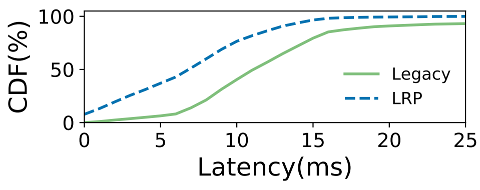
NOTE: Mobile VR is evaluated under Verizon and T-Mobile only. Other operators' firewalls block the VR traffic. Leg: Legacy LTE. Med: Median.

8.6.2 Overall Benefits for the Applications

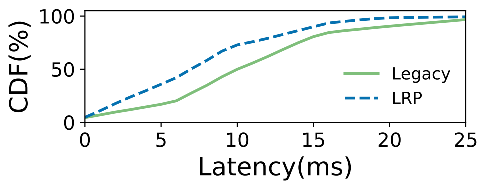
We showcase LRP's latency reduction and QoE improvements with four representative emergent mobile applications:

○ *Mobile VR.* we have built an example VR game with Unity 3D engine [Rep] on Android phones to study LTE latency. It has three modules: the controller at the device, the camera controller at the server, and the streaming component. The Android controller app acquires the device rotation data from the gyroscope sensor to control the in-game camera rotation. The GPS location is fed into the VR game so that the virtual character moves with the player's location updates. Upon receiving the player's sensory data, the camera controller at the server processes them and makes corresponding position and rotation movements for the virtual camera. We implement the streaming module with open-sourced libraries Unity Render Streaming [Str] and WebRTC for Unity [Uni]. With the streaming module, the camera view is rendered and streamed back in 60FPS to the player with WebRTC. Players open the camera stream with the Web browser on the phone to get the real-time camera view. We measure the latency of the sensor data and control data, and use it to gauge how our design reacts to VR games.

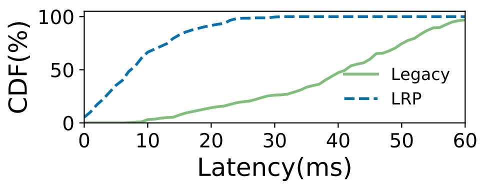
○ *Localization.* We write an Android app that uploads the periodic GPS location status to the cloud via the Android API [Goo20]. We encode each location update in 22 bytes and send it to the cloud every second.



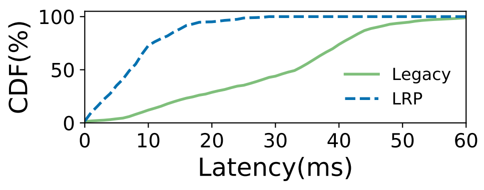
(a) Mobile VR



(b) Gaming



(c) Localization



(d) Object Detection

Figure 8.17: LTE latency with and without `LRP` in representative apps.

◦ *Object recognition.* We prototype an object recognition app using MobileNetV2 [SHZ18], a phone-based deep learning model. The app processes camera frames and uploads the recognition result to the cloud. The typical inference time is 250ms.

◦ *Gaming.* We evaluate its latency by replaying the traces from PUBG Mobile [PUB20], one of the most popular multi-player online mobile games. Since PUBG traffic is not strictly regular, we use it to demonstrate the effectiveness of `LRP` for non-regular traffic. We use the traffic emulator to send data packets based on the trace.

Overall LTE latency reduction Table 8.9 and Figure 8.17 show `LRP`'s latency reduction for these apps in static settings with fair-good signal strength; other scenarios have similar results as detailed in §8.6.3. On average, `LRP` achieves 4-5ms (0.5-0.8×) latency reduction in mobile VR, 8-37ms (1.2-7.4×) reduction in localization, 8-29ms (1.0-5.6×) reduction in object detection, and 1-6ms (0.3-0.7×) reduction in gaming for all 5 LTE carriers. Our breakdown analysis further shows these apps suffer from different latency bottlenecks. For the localization and object detection, the majority of data packets suffer from both DRX doze and scheduling latency. For the VR and gaming with more frequent packets, the scheduling latency is the major latency bottleneck. `LRP` can reduce both bottleneck latencies and thus benefit all these applications.

QoE improvement To showcase the impact of LRP on the mobile VR, we conduct a user study with 10 participants to evaluate the subjective experiences of using VR with/without LRP. Figure 8.18 shows the average Mean Opinion Score (MOS) on three aspects: graphical visual quality, responsiveness, and overall experience. Participants rate 1 (Bad) to 5 (Excellent) on these three aspects of the VR game with constant head position changes. The results show that LRP can improve the visual quality by 8% (3.1→4.0), responsiveness by 63% (2.4→3.9), and overall experience by 46% (2.4→3.5).

5G latency reduction We evaluate how LRP reduces 5G latency under AT&T 5G network. Since we do not have access to its fine-grained data-plane traces, we measure RTT at the application layer. LRP reduces RTT by 4.6ms for Gaming, 20.5ms for Localization, and 19.8ms for Object Detection. The results are similar to the latency reduction in AT&T 4G.

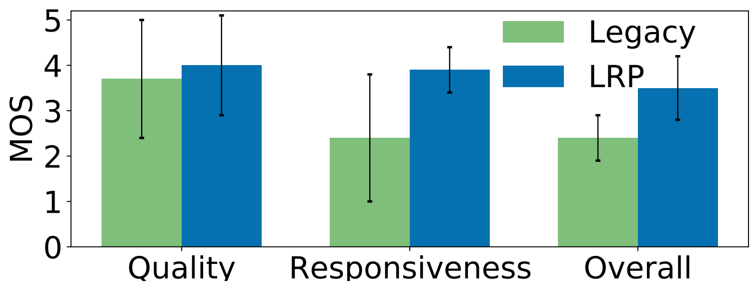


Figure 8.18: MOS of mobile VR.

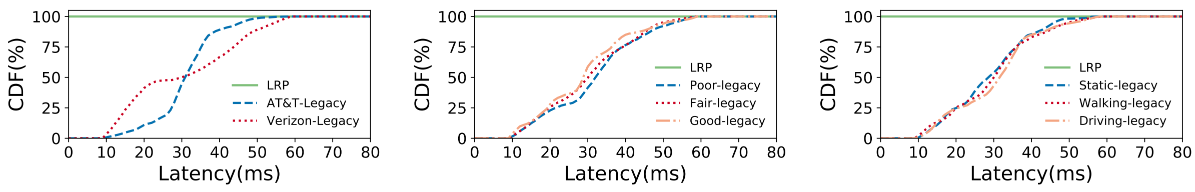


Figure 8.19: LRP reduces DRX doze under different operators, signals, and mobility.

8.6.3 Micro-Benchmarks

We next assess LRP's solution components under various signal strengths and user mobility patterns.

DRX-induced latency reduction (§7.3.1) As shown in §7.3.1, `LRP` helps reduce the DRX doze latency if the inter-packet interval larger than $T_{inactivity\_timer}$ (otherwise the DRX doze latency is always 0 with/without `LRP`). Figure 8.19 shows `LRP`'s DRX latency reduction under various signal strengths and mobility patterns. We run this test under the most popular setting of $T_{inactivity\_timer}$ = 200ms (Table 7.3) when the inter-packet interval is $1.5 \cdot T_{inactivity\_timer}$. We also test other intervals and get similar results. In all scenarios, `LRP` reduces the DRX doze latency to 0 for all LTE carriers. This results in 21–41ms mean latency reduction and 40–57ms 95% latency reduction.

Table 8.10: Scheduling latency (ms) in five mobile carriers. η=(Legacy-`LRP`)/`LRP`.

Scenario		AT&T			Verizon			T-Mobile			Sprint			China Mobile		
		Leg.	LRP	η	Leg.	LRP	η	Leg.	LRP	η	Leg.	LRP	η	Leg.	LRP	η
Static-Poor	Med.	12.0	5.0	1.4×	12.0	9.0	0.3×	11.0	7.0	0.6×	20.0	12.0	0.7×	16.0	5.0	2.2×
	95%	17.0	11.0	0.5×	17.0	17.0	0×	17.0	16.0	0.1×	30.0	26.0	0.2×	26.0	23.0	0.1×
Static-Fair	Med.	13.0	8.0	0.6×	11.0	8.0	0.4×	17.0	13.0	0.3×	18.0	14.0	0.3×	14.0	4.0	2.5×
	95%	17.0	15.0	0.1×	17.0	13.0	0.3×	12.0	6.0	1.0×	32.0	29.0	0.1×	24.0	10.0	1.4×
Static-Good	Med.	13.0	6.0	1.2×	10.0	5.0	1.0×	8.0	6.0	0.3×	13.0	7.0	0.9×	13.0	4.0	2.3×
	95%	17.0	11.0	0.5×	17.0	16.0	0.1×	16.0	11.0	0.5×	27.0	18.0	0.5×	24.0	9.0	1.7×
Walking	Med.	11.0	8.0	0.4×	13.0	6.0	1.2×	12.0	7.0	0.7×	19.0	13.0	0.5×	16.0	9.0	0.8×
	95%	17.0	11.0	0.5×	16.0	16.0	0×	17.0	16.0	0.1×	30.0	26.0	0.2×	30.0	26.0	0.2×
Driving	Med.	14.0	8.0	0.8×	14.0	8.0	0.8×	12.0	8.0	0.5×	17.0	13.0	0.3×	17.0	10.0	0.7×
	95%	18.0	17.0	0.1×	17.0	11.0	0.5×	17.0	16.0	0.1×	29.0	27.0	0.1×	37.0	28.0	0.3×

Scheduling latency reduction (§7.3.2) We next quantify the reduction in uplink scheduling latency. The latency reduction ratio, η, is defined as that of the reduced latency and the `LRP` latency. Table 8.10 shows the results in different carriers, signal strengths, and mobility patterns. In all these scenarios, `LRP` reduces the median scheduling latency by 0.3-2.5×, and reduces the 95th latency by up to 1.7×.

Conflict handling for latency reduction (§7.3.3) We confirm the effectiveness of `LRP`'s conflict resolution/avoidance. We adapt `LRP`'s APIs to enable/disable the conflict handling in §7.3.3. Table 8.11 compares the overall latency with/without `LRP`'s conflict handling. We first illustrate `LRP` can resolve *rouser* and *prefetcher* conflict. We use Localization as

Table 8.11: Latency (ms) reduction with conflict handle.

Conflicts		AT&T	Ver.	T-M.	Spr.	C. M.
rouser & prefetcher	w/o Res.	28.0	30.0	34.0	10.0	13.0
	`LRP`	33.0	36.0	37.0	16.0	16.0
	Extra Red.	17.9%	20.0%	8.82%	60%	23.0%
Data & dummy	w/o Res.	3.5	2.0	2.0	5.0	2.0
	`LRP`	4.0	2.0	2.0	6.0	2.0
	Extra Red.	14.3%	0.0%	0.0%	20%	0.0%

its traffic pattern satisfies the condition (long interval) for potential conflict. Compared with no conflict resolution, `LRP` reduces extra 8.82-60.0% latency in all operators. We next evaluate how `LRP` handles data and dummy packets conflicts. The heavy traffic in the Gaming application potentially causes such conflict. We run the Gaming application with `LRP` and the APIs without conflict avoidance. Compared with no conflict avoidance, `LRP` reduces up to 20% extra latency.

Accuracy of critical parameter inference (§7.4) We finally check how accurate our LTE parameter inference is. For each cell, we first collect ground truth by analyzing the physical/link/RRC-layer signaling messages from MobileInsight. We then use `LRP` component to infer the parameters and compare them with the ground truth. We calculate the average error rate in terms of inference. The results are shown in Table 8.12. As we can see, the inference error rate is at most 3.2% for both parameters in all 5 operators. `LRP` inference is accurate as argued in §7.4 and §8.6.1.

8.6.4 Overhead

Overhead of dummy messages The dummy messages may incur additional data usage and thus billing. Table 8.13 shows that `LRP` incurs no more than 0.6KB data per second under all carriers. The data overhead depends on the frequency of calling LRP APIs. For

Table 8.12: Error rate of LRP parameter inference.

	AT&T	Verizon	T-Mobile	Sprint	C. Mobile
Infer T_{rtt}	3.2%	1.5%	2.0%	3.0%	2.0%
Infer $T_{drx\_doze\_max}$	3.0%	1.3%	3.0%	1.3%	2.5%

Table 8.13: Overhead of LRP.

	AT&T	Verizon	T-Mobile	Sprint	C. Mobile
Extra Data (KB/s)	0.20	0.15	0.41	0.23	0.60
Extra Sig. Msg	3.8%	3.7%	4.3%	3.3%	1.1%
Energy Overhead	1.7%	4.2%	5.8%	2.1%	4.7%

heavy traffic applications (VR, Gaming), the extra overhead is 0.33KB/s while the number for the other two apps is 0.05KB/s. The overhead is acceptable in typical data plans and the extra data is only incurred when LRP APIs are called. As explained in §8.6.1, LRP has minimized the use of dummy for efficient latency reduction.

Extra signaling message The dummy messages incur extra signaling between the device and the BS. We measure this overhead as shown in Table 8.13. LRP incurs up to 4.3% messages, which are marginal compared with the total volume of signaling messages. Reducing latency for apps with DRX doze generates more messages. LRP incurs on average 1.6 extra signaling messages per second for Location and Object Detection. While the other two apps with LRP generate 0.8 extra message every second on average.

Energy consumption While LRP exploits the DRX for lower latency, it still respects the LTE's energy saving with accurate timing control and incurs marginal energy cost. We first compare the percentage of the extra ON period with and without LRP. We track the CDRX events with MobileInsight. As shown in Table 8.13, for all carriers, at most 5.8% of extra ON period is invoked. Furthermore, we fully charge the device and run Object Detection (with DRX doze) and VR (no DRX doze) applications for one hour, and compare the energy consumption with or without `LRP`. With `LRP`, two applications incur 2.5% (16.12% to 16.52% of total battery) and 1.0% (37.04% to 37.40% of total battery) extra battery consumption, respectively. This overhead is marginal, as we adjust the timing of *rouser*s to reduce unnecessary energy waste.

Network impact We measure the network impact of `LRP` in our SDR testbed. Even in the absence of data transfer, the server spends 0.055ms on average to process the collected signal in every subframe (1ms). In contrast, processing `LRP`'s extra signaling costs 0.002ms, about 3.6% extra overhead.

Impact on other users We next examine whether `LRP` affects those non-`LRP` users. We test a two-device scenario, with both running the Gaming app. Device A never uses `LRP`, whereas device B turns on/off `LRP` in the test. When B does not run `LRP`, A's average uplink network latency is 15.79ms, and the 95th percentile is 24.0ms. When B activates `LRP`, A's average latency becomes 15.84ms and the 95th percentile is 24.0ms. Both numbers are not visibly affected. Therefore, the latency of non-`LRP` device is not affected, regardless of whether the other runs `LRP` or not.

CHAPTER 9

Conclusions and Future Work for 5G/4G/xG Systems

9.1 Summary of the Results

The thesis takes a step forward towards a more secure 5G/4G system, and provides insights on future xG system design. On the attack part, we discover novel attacks with forged data-plan signaling messages. We also present a new methodology `FANE` to launch the attacks without any impractical assumptions (no FBS, key compromise, insider attacker, etc.). To defend against any forgery attacks including our proposed ones, we explore two complementary philosophies to take them: proactive defense and reactive detection/mitigation. Together, they can both protect legacy devices and secure future generations of mobile network users. We further explore the approaches to improve performance without sacrificing security by proposing a rootless design.

9.1.1 New Attacks are Still Feasible

Novel attacks are still feasible after mutual authentication.

`CDS` We study a relatively unexplored security topic on cellular IoT: data-plane signaling-induced attacks. We show that, despite all existing security mechanisms on both control-plane signaling messages and data packet forwarding inherited from the legacy 5G/4G networks, new attacks exploiting the unprotected data-plane signaling are still feasible in C-IoT networks. Our attack, `CDS`, can forge data-plane signaling messages that are successfully decoded and accepted by the receiver at both UL and DL. This is achieved by leveraging the cleartext data-plane signaling and other vulnerabilities in RLC/MAC/PHY protocols. Moreover, an attacker can leverage the forged data to inflict damages way beyond the commonsense denial-of-service related threats. Adversaries may breach location privacy, limit the C-IoT device throughput, create local forwarding loops, reset the C-IoT connection at will, prolong the packet delivery, and drain the C-IoT radio resources. Such attacks will expose unattended cellular IoT devices to bigger damages and risks.

We implement the data-plane signaling forgery and the attacks that leverage these forged data-plane signaling messages. Our testbed evaluation and validation over operational networks have confirmed the feasibility of such new attacks. To protect the data-plane signaling messages with low overhead, we propose a time-based defense solution at MAC. It leverages the synchronized timers that are readily available at MAC to derive keystream for integrity protection and encryption. We implement the solution and our evaluation confirms its effectiveness and cost-efficiency.

`FANE` we have described novel 5G/4G data-plane attacks that allow us to break the over-the-air data delivery, despite security countermeasures deployed in cellular networks. Such attacks exploit multiple vulnerabilities at different layers of the 5G/4G protocol stack. Indeed, each individual vulnerability does not appear threatening and can be often overlooked. However, their concerted actions may yield an avalanche effect in the system. The deeper root cause lies in isolated protection in 5G/4G system. From the security perspective, segmentation arises between high layers and underlying layers, and between control and data planes. Consequently, securing one vulnerability (say, one single data plane layer or control plane only) cannot protect others. An adversary can thus bypass security checks on control plane and higher layers, yet deploy a number of puppet nodes for attacks at lower layers on data plane. Such attacks do not even require link-layer connectivity with the victim device; they thus bypass the countermeasures against current threats via fake base stations or unprotected broadcast signaling messages. The forged data packets can break the users' Web access or breach their privacy.

9.1.2 Novel Countermeasures that are Feasible

DP^2 In this dissertation, we propose a new protection scheme for 5G from a new angle. Instead of data (data-plane packets or signaling), we propose a solution that targets meta-info, which is necessary for successful data delivery. If the meta-info is secured, an adversary will fail to forge data that can be accepted correctly. This voids any attempt to forge or manipulate data in the access network, including new attacks like FANE that cannot be protected by the existing 5G security schemes.

We design DP^2, our meta-info protection scheme to protect DCI information. It is secure, as it leverages the proven algorithms and synchronized time/frequency information to generate keystream without key reuse. Meanwhile, it incurs marginal overhead with the low DCI overhead and inference techniques. The solution does not incur extra data overhead by not requiring integrity protection. We validate the correctness of DP^2 and evaluate its low overhead in a real SDR-based testbed. Our proposed remedies target immediate protection for user traffic and signaling messages, even against non-standard-compliant attackers. The solution is also capable of mitigating attacks in future releases of 5G standards, as it fundamentally blocks any attacks that require data forgery.

CellDAM Our proposal CellDAM takes the "detect and mitigate" approach, in order to be immediately deployable without any infrastructure upgrade. It makes three contributions. First, our analysis of 5G data-plane attacks reveals a key finding that, any known signaling forgery or packet fabrication will trigger abnormal data-plane signaling changes. We thus can detect attacks via inspecting the data signaling messages, which are much more lightweight compared with the 5G data traffic. Second, we propose a novel state-dependent model checking without root privilege or firmware access. This is done through new inference technique on all signaling messages and involved 5G configuration parameters at the companion SecHub. Third, we propose a mitigation scheme based on rootless frequency band switching. This is to exploit the current 5G handover procedure and its dense cell deployment practice, and focus on the protected user device only. Our security analysis, prototype, and empirical assessment have confirmed that CellDAM can detect known and unreported attacks and mitigate their damages in a timely manner with small overhead.

9.1.3 Security-Aware App Optimization

Diagnosing latency bottleneck and reducing it is critical to many delay-sensitive applications, such as mobile AR/VR, mobile gaming, sensing, machine learning, and robot/drone-based image/speech recognition. In mobile networks, reducing uplink latency is more challenging than its downlink counterpart, since it involves multiple latency elements stemming from power-saving, scheduling, on-demand resource allocation, mobility, etc. However, doing so should not require extra privilege, which exposes new security loopholes and is sometimes infeasible.

We have designed and implemented `LRP`, a device-based solution to LTE latency without any infrastructure changes. `LRP` does not require root privilege at the device and works with mobile apps directly. It learns the critical LTE parameters to infer the uplink latency components for latency diagnosis. `LRP` then leverages them to reduce the latency. It ensures the network latency is no worse than the legacy 4G LTE, and is applicable to 5G with experimental validation. By design, `LRP` uses small dummy messages with proper timing control and conflict handling, in order to eliminate unnecessary latency components from scheduling and power-saving operations. Our experiments have confirmed its effectiveness with a variety of mobile apps.

In the broader context, exploring pure device-based solution, which does not require root privilege, offers a secure, nice complement to the infrastructure-centric design, which needs standard update and typically takes years to be deployed. Besides, reducing latency poses a more challenging problem than improving throughput for the networked system community. In the mobile network domain, various tricks have been invented for boosting throughput (e.g., massive MIMO, new modulation, mmWave, etc.). This is not the case for latency. Both its fundamental theory and effective practice are lacking.

9.2 Lessons Learned

9.2.1 Don't trust any message, even after authentication

The traditional Telecom systems take a "closed-community" trust model, in which a network node trusts every incoming information from the authenticated peer. However, as we have shown in `CDS` and `FANE`, a message can come from an adversary even after mutual authentication. This is due to the open channel design of the 5G/4G networks and the overall trending to more data-centric nature. As a result, carefully inspecting the incoming data is necessary for sufficient security.

9.2.2 Protect necessary instead of all data

It takes a couple of years for 5G to enable full-rate support for data packet integrity protection. Yet, its usage is still optional. Considering the high data throughput and massive connected devices for a cell, adding per-message protection incurs prohibitively high overhead. However, from this dissertation, we demonstrate that a more lightweight solution can be achieved if we focus on *necessary* condition of the forgery.

9.2.3 Verify what's right, don't target certain attacks

Many security solutions target specific attacks, which can be a quick fix to the threats. However, this philosophy suffers from its applicability as other attacks cannot be protected or potentially find attacks that bypass the solution. Instead, a more promising solution is to check the correct operations stipulated by the standards and treat undesired behaviors as suspicious. This method can find attacks that are even not reported, as they potentially trigger behaviors that are incompatible with the normal operations.

9.2.4 Device side can do more for security

The traditional approaches attempt to deploy solutions on the network side, as the end devices are restricted to having sufficient information for any protection or detection. However, with the development of new device-side tools [LPY16, Qua], an application can get much detailed information in 5G/4G systems. Therefore, device side can take more responsibility of the security in mobile systems with the additional capability.

9.2.5 Optimization does not necessarily need extra privilege

Improving performance for emerging applications does not necessarily need to sacrifice security by yielding certain permissions. Instead, with domain knowledge, the critical network-specific information can be inferred in the userspace without root. With this solution idea, performance boosting is achievable while not affecting security.

9.3 Looking Forward

Given the high priority of cellular system security, continuous efforts must be devoted into this area. Besides the technical contribution of this dissertation, we further propose multiple directions where urgent actions are needed to secure the security and privacy of mobile network users.

Firstly, we will discuss how to extend the similar philosophy to study the security of other 5G/4G/xG operational modes (§9.3.1). Second, a complementary topic of passive attacks needs to be studied that cannot be addressed by the current techniques in this dissertation (§9.3.2). Third, our rootless solution approach can potentially be applied to emerging application requirements other than low-latency (§9.3.3). Finally, we discuss the opportunity to redesign future 6G systems which can fundamentally eliminate multiple security threats (§9.3.4).

9.3.1 Emerging 5G/4G Operational Modes

With the development of new technologies, our attacks and countermeasures in current 5G/4G might not be directly applicable. This contains multiple aspects. For broadband device, 5G adopts URLLC (Ultra-Reliable Low Latency Communications) as one of its three pillar use cases. This is an important operational mode to support applications with requirements on reliability or latency. 3GPP is proposing a set of different technical designs for the operational mode. For instance, the access control in URLLC has been redesigned: instead of base station-controlled schemes, each device can be assigned some fixed resource to access the radio resource. This way, the DCI might be no longer necessary. `FANE` attacks and the protection schemes proposed by this dissertation will need certain adaptation. How to attack and protect devices that run URLLC need further study.

For narrowband technologies, many new technologies are proposed to secure the IoT device communication. In addition to Control Plane CIoT Optimization mentioned in the text, 3GPP has designed and deployed User Plane CIoT Optimization. It takes a different design approach. Control plan is redesigned, e.g., certain signaling messages are cancelled as compared to legacy 5G/4G. For attacks targeting such IoT devices, new approaches and countermeasures need to be further studied.

We further note that, studying these new operational modes need system building efforts, too. For one, studying the new features requires a tool to collect logs from COTS devices. We need to continuously apply new message decoding support with MobileInsight IoT. Besides, evaluating new attacks and defenses demands flexible, SDR-based testbeds. Therefore, it is important to take efforts and implement the new features in Flora and Sonica.

9.3.2 Passive Attacks and Countermeasures

This dissertation mostly focuses on active attacks and how to defend against them. However, these proposed approaches cannot effectively tackle passive attacks. Under this threat model, the attacker does not actively send signals. Instead, it passively eavesdrops on the channels and analyzes the user traffic. This allows an attacker to identify the application type [BBM17], Web [KRH19], or video [bae22] that the victim is viewing. Note that, this category of attacks cannot be effectively protected by `DP`^{2} or `CellDAM`, as the attacker only needs to learn the metadata of the packets. These attacks are substantial threats to users' privacy.

Proper redesign is necessary to disallow an attacker from eavesdropping on the channel. One observation is, an attacker needs to use C-RNTI to identify the traffic that belongs to a specific user. This C-RNTI ID is not correctly bound with the higher layer key credentials. One potential idea is, the attacker and base station can use a dynamic C-RNTI scheme over time. This prevents the attacker from tracking the victim effectively.

The approach comes with multiple challenges, though. First, the device and base station need to update the C-RNTI in sync. Otherwise, the correctness of data delivery is not guaranteed. Second, each device needs to have a distinct C-RNTI. This is to avoid collision among different users in a cell. Third, the scheme needs to ensure the security even against a smart attacker who is aware of the existence of the device. A smart attacker can either track the change of IDs or use the header information to decide which C-RNTIs belong to the victim. Thus, this requires a carefully-designed secure C-RNTI scheme. Finally, the update cannot incur too much overhead, as the base station can serve hundreds of users in a cell. Excessive overhead will decrease the capacity of a cell.

9.3.3 Optimizing Emerging Applications: Beyond Low Latency

Emerging applications demand diversified features from 5G/4G networks. This includes high throughput, extreme mobility, low latency, long battery life, etc. In this dissertation, we explore the aspect of low-latency without sacrificing security. We present a solution that can handle latency reduction at the application layer without root or extra privilege. To support other demanding features, it will require us to design similar rootless, user-space solutions that can serve the emerging applications while preserving high security.

However, it is not easy to achieve the goals and here are the intuitions. The throughput, power consumption, or mobility events are mostly dictated by base station scheduling or configurations. They cannot be accessed by the device side or rootless applications. Even if an attempt can be made to force changes on the base station side, it could potentially incur fairness concerns. In addition, unlike the latency components in `LRP` that can be inferred from application layer, other features such as power consumption are not possible to be inferred without root.

One potential idea is to leverage the solution component from `CellDAM`. With a companion node, the device side is capable of diagnosing the power consumption and other latency components by monitoring the channel signaling messages. In addition, the companion node can actively transmit uplink and downlink signals to facilitate the applications requirements.

9.3.4 Rethinking the Design of xG Mobile Systems

Given the loopholes and shortcoming of the state-of-the-art 5G/4G systems, it is worth exploring new security philosophies and redesigning system components in future xG mobile systems. The current system is inherited and patched from the traditional circuit switching networks for phone call and text messages. However, some principles are no longer valid for the current packet-centric 5G/4G networks. For instance, the trust model in the traditional network will no longer hold. Instead of a closed community, mobile networks are connecting more and more legitimate and illegal system components with the development of handover, roaming, fake base stations, and shadow access network attackers. Simple, unprotected access control no longer suffices. Another example is the protocol stack design that uses PDCP encryption. This fundamentally results in the insecurity in the lower layers. A novel design can overcome the issue by rearranging the capabilities in different layers.

Looking forward, xG systems have much potential to serve emerging applications for a large number of users. With continuous efforts, we are confident that future xG systems will be more secure, more scalable, and more efficient for next-generation user scenarios.

APPENDIX A

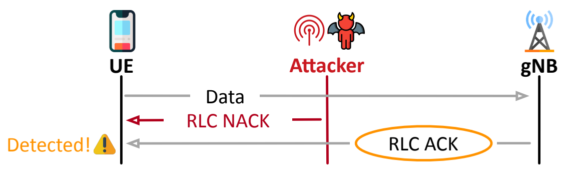
Details of Undesired Behavior for Each Attack

In this appendix, we present the details of each attack shown in Table 6.2, including the attack procedure that leverages the forgery messages and the attack consequences. We also elaborate on how each attack incurs undesired behavior that can be detected by `CellDAM` validations.

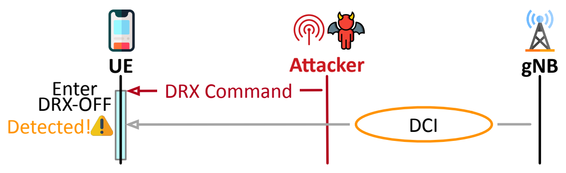
A.1 Packet Delivery Loop

Attack Procedure We now present this attack in more details as shown in Figure A.1(a). The attacker first eavesdrops on the data channel and learns the packet sequence number that is delivered over the air. It then forges an RLC control that NACKs the packet and thus forces the retransmission. Since the packet is not acknowledged yet, UE will recognize it as a legitimate message without `CellDAM`. Consequently, the unnecessary retransmission is forced, which blocks future legitimate transmission as MAC will prioritize the RLC retransmission.

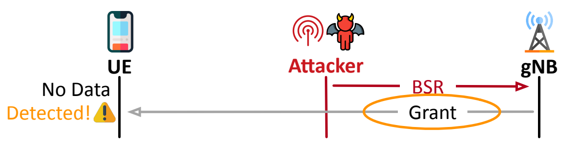
Undesired Behavior The attack will cause undesired behavior on RLC and MAC protocols. From the perspective of the legitimate gNB, the data that is forced to retransmit has already been received. Therefore, the gNB will still send an RLC ACK to the device later, indicating the packet has been delivered and received by RLC. However, this packet might arrive before the retransmission on MAC (forced by attacker) is finished. Therefore, the UE can detect a potential attack with c_1, as the RLC ACK is received in the state when the MAC delivery is not acknowledged.



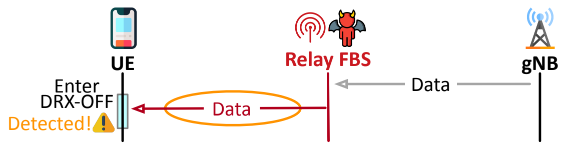
(a) Packet delivery loop.



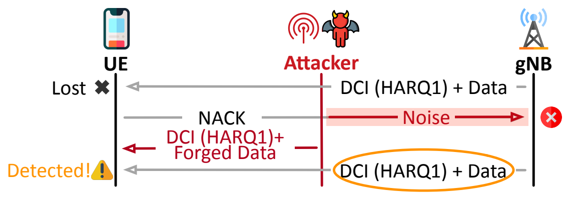
(b) Prolonged packet delivery.



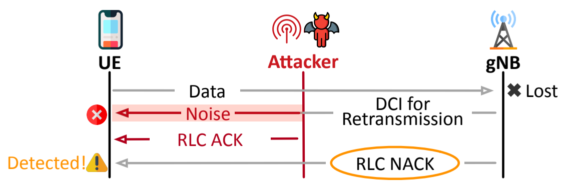
(c) Radio resource draining.



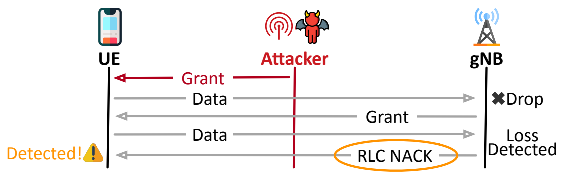
(d) Downlink data forgery with FBS.



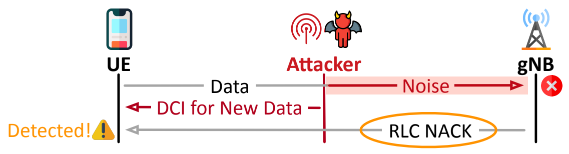
(e) Downlink data forgery with retx.



(f) Break reliable transfer.



(g) Data collision.



(h) Delayed transfer.

Figure A.1: Illustration of undesired behavior caused by attacks.

A.2 Prolonged Packet Delivery

We consider connected mode Discontinuous Reception (CDRX) in this attack. In the RRC connected state, the gNB will only deliver data during DRX ON state. A device will be in DRX ON for a small time period in a fixed periodicity. If any data is received during this period, the DRX ON state is extended for a constant amount of time. gNB will configure the ON period, DRX cycle, and the extended timer amount in encrypted RRC messages.

Attack Procedure The detailed procedure is shown in Figure A.1(b). In this attack, the adversary forges a DRX command to the device. DRX command terminates the DRX

ON state prematurely and the device enters the sleep mode. Therefore, the device will be unable to receive all subsequent transmission in the current DRX cycle with DRX turned to OFF state. The data delivery can be delayed for hundreds of milliseconds given long DRX cycle. The attack is adapted from [TDZ21]. Note that, this attack will not stop uplink data delivery, as a UL data transmission initiated by UE can again switch the DRX state to DRX ON.

Undesired Behavior It is not possible to receive downlink DCI or data during DRX OFF. If such an event happens, the previous DRX command could be forged. As `CellDAM` does not have root access, it is difficult for `SecHub` to monitor internal DRX state in real-time or infer the state with encrypted DRX configurations in RRC messages. However, there is another indication for a message during DRX OFF: If the device is in DRX OFF, it will not respond to any downlink data with ACK or NACK in PUCCH. Therefore, `SecHub` can detect such attack by checking the downlink data without any acknowledgement, after an incoming DRX command. This violates the validation check c_4. In addition, gNB will not send DRX command when the previous DL transmission has not finished, which means there will be more transmission in the DRX cycle. This violated c_1.

A.3 Radio Resource Draining

Attack Procedure 5G/4G adopts scheduling-based data delivery. To transmit uplink data, the device needs to send the buffer status report (BSR) to the gNB for asking grants. An attacker can forge a BSR to the gNB. In the forged BSR, the attacker falsely indicates the victim device has a large amount of data to send. The gNB subsequently assigns excessive resource blocks to the device, wasting wireless resource and blocking other users' access. The detailed procedure is shown in Figure A.1(c). The attack is adapted from [TDZ21].

Undesired Behavior Although the attacker forges an uplink message, it will incur observable undesired behavior on the device side as well. This is because the DCI (for UL grant) will be triggered by the forged BSR message. If the device receives grant when there is no prior request, the grant can be caused by a forged request by the attacker. We notice that, a gNB can send a device "free" small grants in case the device has something to send. However, these grants are small for the device to sufficiently deliver BSR. Therefore, to reduce false positive, we set a threshold (120 bytes) in the verification check c_5 to find unwanted resources.

A.4 Downlink Data Forgery with FBS

Attack Procedure The detailed attack procedure is shown in Figure A.1(d). The attacker sets up a fake base station (FBS). It usually runs on a different band from the real base station, and sends strong broadcast signal to lure the device into connection. The attack also sets up a fake UE that connects to the real base station. It then relays the connection setup messages without any manipulation from/to the real base station. Afterwards, although all data packets are encrypted, the attacker can flip the bits to change the content of the forgery, if data-plane integrity protection is not enforced. This is doable as the encryption is done with simple XOR. as long as the original content is known (e.g., DNS server set up by the operator), the attacker can flip bits and change the contents to target values. For instance, the attacker can manipulate the IP header of a DNS request and compensate the IP header checksum to pass the checks [RKH19].

Undesired Behavior In this attack, we assume the adversary cannot infer the data-plane configurations encrypted in RRC messages. Therefore, when the attacker forwards the data packet, some configurations might be incorrectly set and detected on the device side. One example is DRX configuration. Without the configuration, the forwarded packet might fall in the DRX OFF period, as shown in the figure. As the DRX ON period is usually very short (e.g., 10ms), most messages might be delivered outside of the period, violating check c_4. Although the FBS can repeat transmitting until the victim device acknowledges to ensure delivery, this behavior will be detected by `CellDAM`. Note that, 5G DRX includes the mechanism to stay in ON period for an extended period when a new data is received. Therefore, when the traffic is heavy, the device might keep staying in ON state The attack will be more detectable for light traffic. For this attacker, the PHY layer detection methods to detect FBS that transmits abnormal signal.

A.5 Downlink Data Forgery with Retransmission

Attack Procedure The detailed attack procedure is shown in Figure A.1(e). This approach can manipulate data plane packet without FBS. We consider the victim device is directly connected to the authentic base station. Note that, the attacker might not be able to forge data-plane packets directly, as they are encrypted (unlike integrity protection, which is optional). Therefore, to forge data packets, the attacker still needs to take the bit flipping approach as in A4. One viable way is to forge the data as retransmission. The attacker can eavesdrop on the channel and look for data transmission that fails on victim device (i.e., trigger NACK), while it successfully decodes the encrypted data (due to less noisy environment, etc.). The attacker can then forge the DCI and manipulated data as the retransmission.

Undesired Behavior This attack requires sending forged DCI and data to the device. This DCI can be received in an improper context. Each DCI includes a HARQ ID, which indicates the process of the transmission. Each process takes a stop-and-wait procedure. Before the last packet is acknowledged, the process will not move on to transmit the next one. Therefore, the DCI from the authentic gNB can arrive after the DCI forged by attacker and before the forged retransmission has been acknowledged. This causes an undesired behavior that can be observed on the device with c_1.

We further note two things. First, this attack method can forge *uplink* data packet without obvious undesired behavior on the device side. We do not consider such attack with pure uplink forgery in our detection. However, as we discussed in §6.2.1, a reasonable forgery attack needs to manipulate both uplink and downlink data. Second, an interested reader might ask whether the attacker can use the legitimate DCI to send the forged data. However, the DL DCI and the corresponding forged data are usually sent in the same time slot in 5G. It is thus hardly possible to infer DCI for data forgery in advance.

A.6 Break Reliable Transfer

Attack Procedure The detailed procedure of the attack is shown in Figure A.1(f). The attacker can corrupt the data retransmission on MAC layer. It then forges an RLC ACK to UE. Receiving it, the victim device RLC protocol wrongly thinks the packet has been delivered, discarding it in the buffer. Therefore, this packet cannot be reliably delivered in 5G. This might further trigger TCP retransmission, which can cause more serious damage.

Undesired Behavior From the perspective of the base station, it will detect a packet gap in RLC protocol when it receives a later packet from UE. Therefore, it will still attempt to recover it by sending a RLC NACK. The NACK timing is unknown for the attacker, thus cannot be targeted for corruption. The device side will thus receive an RLC NACK first and then an ACK. This behavior is undesired in 5G and can be detected by validation check c_3.

A.7 Data Collision

Attack Procedure The detailed procedure of the attack is shown in Figure A.1(g). The attacker forges grant to the victim device. The device will send data using the forged grant. However, any data using these non-authorized grants will not be correctly accepted by the gNB. In addition, the legitimate transmission in the same time and frequency by another user will be corrupted by the victim's false transmission.

Undesired Behavior The attack will trigger undesired behavior on the device side. Since the UL data using false grants will not be accepted, gNB will not ACK or NACK the message delivery and consequently trigger an RLC NACK. `CellDAM` detects the undesired behavior with RLC control NACK and lack of MAC layer feedback with validation c_5.

A.8 Delayed Transfer

Attack Procedure The detailed procedure of the attack is shown in Figure A.1(h). In this attack, the adversary sends noises and corrupts the uplink data. The attacker can learn the time and frequency of the delivery by eavesdropping on DCI for UL grants. It consequently forges DCI for new data (i.e., indicator for ACK) to stop the UE from retransmitting the corrupted data. Consequently, the UE will start sending new data on MAC. This will later trigger retransmission on the RLC layer, which can take up to hundreds of milliseconds compared to a MAC fast retransmission.

Undesired Behavior The forged DCI from the attacker can be sent in a wrong context, where two consecutive DCIs are received but the data using the first one has not been acked yet. This can also be detected with c_1.

APPENDIX B

List of Accepted Messages in `CellDAM` DFA

In this appendix, we elaborate on c_1 by enumerating each state and the list of accepted message for each one. The results are shown in Table B.1. If a message is in the list, we show the next state if all other validations are passed. Otherwise, we mark an $\times$ in the table which means an undesired behavior that fails c_1. We do not include s_5 and s_9 in the table, as they are accept states. $\times$ means that the message in this state is not allowed and cannot pass c_1, $-$ means that the state is unchanged with this message. "Different" or "same" means the receiving DCI compared with the first DCI for an RLC data packet, i.e., RLC retransmission will reset HARQ and NDI with the first DCI after. As we mentioned, our method prioritizes soundness. Therefore, for messages that are not explicitly considered, `CellDAM` will ignore them and stay in the current state.

Table B.1: List of the accepted data-plane signaling for each DFA state and their state transition.

Data-Plane Signaling Message	Current State						
	s_1	s_2	s_3	s_4	s_6	s_7	s_8
MAC DCI for DL grant with same HARQ, same NDI	$-$	$-$	$-$	$-$	s_7	$\times$	$\times$
MAC DCI for DL grant with same HARQ, flipped NDI	$-$	$-$	$-$	$-$	$\times$	$\times$	$-$
MAC DCI for DL grant with different HARQ	$-$	$-$	$-$	$-$	$-$	$-$	$-$
MAC DCI for UL grant with same HARQ, same NDI	s_2	$\times$	s_2	$\times$	$-$	$-$	$-$
MAC DCI for UL grant with same HARQ, flipped NDI	$\times$	$\times$	s_4	$\times$	$-$	$-$	$-$
MAC DCI for UL grant with different HARQ	$-$	$-$	$-$	$-$	$-$	$-$	$-$
PUCCH ACK for the previous DCI	$-$	$-$	$-$	$-$	$\times$	s_8	$\times$
PUCCH NACK for the previous DCI	$-$	$-$	$-$	$-$	$\times$	s_6	$\times$
RLC Control with ACK for this packet	$\times$	$\times$	$\times$	s_5	$\times$	$\times$	s_9
RLC Control with NACK for this packet	$\times$	s_1	s_1	$\times$	s_6	s_6	$\times$
DRX Command	$-$	$\times$	$\times$	$\times$	$-$	$\times$	$\times$
BSR	$-$	$-$	$-$	$-$	$-$	$-$	$-$
Any other messages	$-$	$-$	$-$	$-$	$-$	$-$	$-$

APPENDIX C

Discussion on the Corner Case in LRP

The corner case happens when the grant from an SR is sufficient for the data packet, but insufficient for the data packet and its *prefetcher*. In this appendix, we discuss the size of the grant from an SR and the probability of this corner case.

Since the size of a data packet and a *prefetcher* is fixed, the occurrence of the corner case depends on the grant from an SR. The BS assigns a grant for the SR according to 3GPP standard [3GP19b]. However, it has the freedom to determine the size of the grant. It can assign certain RBs to the user. The number of the RBs is denoted as N_{PRB}. The RB amount is not sufficient to determine the grant size, which is also affected by the modulation index, denoted as I_{MCS}. A BS sends a grant with I_{MCS}, which is affected by channel condition, device power, etc. N_{PRB} can be selected from a subset of discrete values from {1, ...,110}, depending on the channel bandwidth. I_{MCS} can be selected from a subset of discrete values {0, ..., 63}, depending on the modulation capability of the device. Multiple I_{MCS} can map to the same modulation scheme. The UL data that can be sent using this grant is a function of both N_{PRB} and I_{MCS}. This discrete function, denoted as $F(N_{PRB}, I_{MCS})$, is shown in a 110x44 table in 3GPP 36.213 [3GP19b].

$F()$ is monotonically increasing with either N_{PRB} or I_{MCS}. Suppose the selection of N_{PRB} and I_{MCS} are independent. Let the probability of the BS selecting $P(N_{PRB} = j) = p_j$, where $j \in \{1, ..., 110\}$ and $\sum p_j = 1$. Similarly, let $P(I_{MCS} = i) = q_i$, where $i \in \{0, ..., 63\}$ and $\sum q_i = 1$. Let the size of the data packet be a and the size of a *prefetcher* be a'. $(j, i) \in X_1$ if $a \leq F(j, i) < a + a'$. Otherwise, $(j, i) \in X_2$ Therefore, $p_{corner} = \sum_{(j,i) \in X_1} p_j \cdot q_i$.

From the operational traces, a BS tends to assign $N_{PRB} = 2$ or 3 in response to an SR. When $N_{PRB} = 2$, any $I_{MCS} \geq 3$ can guarantee $F(N_{PRB}, I_{MCS}) > 100$. When $N_{PRB} = 3$, any $I_{MCS} \geq 2$ can guarantee $F(N_{PRB}, I_{MCS}) > 100$. In our experiments, >99% of initial grants exceed 100B in all operators. This is sufficient for a small uplink sensory data packet and a small *prefetcher* message.

APPENDIX D

Proof for Theorem 7.5.1

Proof. `LRP` operates based on the value of $T_{interval}$. When $T_{interval} \geq T_{inactivity\_timer}$, `LRP` sends a *rouser* to eliminate DRX doze latency. When the next packet arrives later than expected ($T > T_{interval}$), `LRP` is still very likely to reduce DRX doze latency as the *rouser* sent T_{drx_doze} before the next packet will keep the device in ON state for $T_{inactivity\_timer}$. Therefore, as long as $T \leq T_{interval} - T_{drx\_doze} + T_{inactivity\_timer}$, `LRP` still reduces DRX doze latency. If $T > T_{interval} - T_{drx\_doze} + T_{inactivity\_timer}$, the device might have already turned to DRX OFF when the next packet arrives. In this situation, the DRX doze latency still exists but `LRP` does not add extra latency source. Similarly, when the next packet arrives early ($T < T_{interval}$), `LRP` still reduces doze latency when the *rouser* precedes the data packet, namely $T \geq T_{interval} - T_{drx\_doze}$. `LRP` cannot eliminate the entire DRX doze latency as the optimal solution, but can still reduce doze latency to $T - (T_{interval} - T_{drx\_doze})$. Otherwise, `LRP` does not send any *rouser* and the latency is the same compared to no `LRP`. In summary, if the next packet actually arrives in T where $T_{interval} - T_{drx\_doze} \leq T \leq T_{interval} - T_{drx\_doze} + T_{inactivity\_timer}$, `LRP` still eliminates or reduces the DRX doze latency. The margin allowed for error (T_{drx_doze} and $T_{inactivity\_timer} - T_{drx\_doze}$) can be 30-80ms in reality depending on common LTE parameters. Otherwise, `LRP` does not increase the latency.

When $T_{interval} < T_{inactivity\_timer}$, `LRP` sends a *prefetcher* only (T_{sr_grant} before the next packet) to reduce scheduling latency. If data arrives later, the *prefetcher* is still possible to save its latency if its requested grant can be used by the next packet, which is $T < T_{interval} + T_{sr\_wait}$. Similarly, if data arrives earlier, the *prefetcher* reduces scheduling latency if it is sent before the real data packet, i.e., $T > T_{interval} - T_{sr\_grant}$. When the next scheduled packet arrives in $T_{interval}$ where $T_{interval} - T_{sr\_grant} < T < T_{interval} + T_{sr\_wait}$, `LRP` still reduces the LTE uplink scheduling latency. This margin allowed for error (T_{sr_grant} and T_{sr_wait}) is usually 8-20ms in reality depending on LTE parameters. Otherwise, the scheduling latency is not reduced but `LRP` does not incur extra latency. $\square$

[3GP17] 3GPP. “Proposed way forward on IMT-2020 submission.”, Sep. 2017.

[3GP19a] 3GPP. “TS33.501: Security architecture and procedures for 5G System-V15.4.0.”, May. 2019.

[3GP19b] 3GPP. “TS36.213: Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures.”, Sep. 2019.

[3GP19c] 3GPP. “TS36.321: Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC) protocol specification.”, Sep. 2019.

[3GP20a] 3GPP. “NR; Medium Access Control (MAC) protocol specification.”, Dec. 2020.

[3GP20b] 3GPP. “TS33.809: Study on 5G security enhancements against False Base Stations (FBS).”, Dec. 2020.

[3GP20c] 3GPP. “TS36.211: Evolved Universal Terrestrial Radio Access (E-UTRA); Physical channels and modulation.”, Sep. 2020.

[3GP20d] 3GPP. “TS36.331: Radio Resource Control (RRC).”, 2020.

[3GP20e] 3GPP. “TS38.331: NR; Radio Resource Control (RRC); Protocol specification.”, Dec. 2020.

[3GP21a] 3GPP. “NR; Radio Link Control (RLC) protocol specification.”, Jan. 2021.

[3GP21b] 3GPP. “TS33.220: Generic Authentication Architecture (GAA); Generic Bootstrapping Architecture (GBA).”, Dec. 2021.

[3GP21c] 3GPP. “TS36.201: Evolved Universal Terrestrial Radio Access (E-UTRA); LTE physical layer; General description .”, Jan. 2021.

[3GP21d] 3GPP. “TS36.212: Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding.”, Jan. 2021.

[3GP21e] 3GPP. “TS36.321: Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC) protocol specification.”, Jan. 2021.

[3GP21f] 3GPP. “TS36.322: Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Link Control (RLC) protocol specification.”, Jan. 2021.

[3GP21g] 3GPP. “TS38.211: NR; Physical channels and modulation.”, Jan. 2021.

[3GP21h] 3GPP. “TS38.213: NR; Physical layer procedures for control.”, Jan. 2021.

[3GP21i] 3GPP. “TS38.214: NR; Physical layer procedures for data.”, Jan. 2021.

[3GP22a] 3GPP. "TS124.501: Non-Access-Stratum (NAS) protocol for 5G System (5GS); Stage 3.", Mar. 2022.

[3GP22b] 3GPP. "TS33.501: Security architecture and procedures for 5G System-V16.4.0.", Mar. 2022.

[3GP22c] 3GPP. "TS37.340: NR; NR; Multi-connectivity; Overall description;.", Jan. 2022.

[AAP17] Pilar Andres-Maldonado, Pablo Ameigeiras, Jonathan Prados-Garzon, Jorge Navarro-Ortiz, and Juan M Lopez-Soler. "Narrowband IoT data transmission procedures for massive machine-type communications." *IEEE Network*, **31**(6):8–15, 2017.

[Abr14] Michael Abrash. "What VR Could, Should, and almost certainly Will be within two years." `http://media.steampowered.com/apps/abrashblog/Abrash\%20Dev\%20Days\%202014.pdf`, 2014.

[ADL19] Mikhail Afanasov, Alessandro Djordjevic, Feng Lui, and Luca Mottola. "FlyZone: A Testbed for Experimenting with Aerial Drone Applications." In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 67–78, 2019.

[AE15] Elsayed Ahmed and Ahmed M. Eltawil. "All-Digital Self-Interference Cancellation Technique for Full-Duplex Systems." *IEEE Transactions on Wireless Communications*, **14**(7):3519–3532, 2015.

[AF19] Arslan Ali and Georg Fischer. "Enabling fake base station detection through sample-based higher order noise statistics." In *2019 42nd International Conference on Telecommunications and Signal Processing (TSP)*, pp. 695–700. IEEE, 2019.

[Ama20] Amazon. "Amazon Alexa." `https://www.amazon.com/Amazon-Echo-And-Alexa-Devices/b?ie=UTF8&node=9818047011`, Mar. 2020.

[Ama21] Amarisoft. "Technology – Amarisoft.", December 2021. `https://www.amarisoft.com/technology/`.

[APIa] Android Blurtooth API. `https://developer.android.com/guide/topics/connectivity/bluetooth`.

[APIb] Android Telephony API. `https://developer.android.com/reference/android/telephony/package-summary`.

[bae22] "Watching the Watchers: Practical Video Identification Attack in LTE Networks." In *USENIX Security 22*, Boston, MA, August 2022. USENIX Association.

[Bak95] Fred Baker. "RFC1812: Requirements for IP version 4 routers.", 1995.

[BBC17] Tristan Braud, Farshid Hassani Bijarbooneh, Dimitris Chatzopoulos, and Pan Hui. "Future networking challenges: The case of mobile augmented reality." In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pp. 1796–1807. IEEE, 2017.

[BBM17] Arjun Balasingam, Manu Bansal, Rakesh Misra, Rahul Tandra, Aaron Schulman, and Sachin Katti. "Poster: Broadcast LTE data reveals application type." In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, pp. 531–533, 2017.

[BBM19] Arjun Balasingam, Manu Bansal, Rakesh Misra, Kanthi Nagaraj, Rahul Tandra, Sachin Katti, and Aaron Schulman. "Detecting if LTE is the Bottleneck with BurstTracker." In *The 25th Annual International Conference on Mobile Computing and Networking*, pp. 1–15, 2019.

[Bla20] James Blackman. "A 2019 surge, a 2020 wobble, a 2025 boom - NB-IoT and LTE-M on track, says Ericsson." `https://enterpriseiotinsights.com/20200616/channels/news/nb-iot-and-lte-m-on-track-says-ericsson`, Jun 2020.

[Bre15] Thomas Arthur Herbert Bressner. "Development and Evaluation of UTDoA as a Positioning Method in LTE." `https://www.diva-portal.org/smash/get/diva2:867797/FULLTEXT01.pdf`, Jun 2015.

[Cal] Understanding Wireless Range Calculations. `https://www.electronicdesign.com/technologies/communications/article/21796484/understanding-wireless-range-calculations`.

[CCH20] Chieh-Chun Chen, Ray-Guang Cheng, Chung-Yin Ho, Matthieu Kanj, Bruno Mongazon-Cazavet, and Navid Nikaein. "Prototyping of Open Source NB-IoT Network." In *GLOBECOM 2020-2020 IEEE Global Communications Conference*, pp. 1–5. IEEE, 2020.

[Cel21] CellMapper. "CellMapper." `cellmapper.net`, Mar 2021.

[Cha19] Arvind Chakrapani. "NB-IoT uplink receiver design and performance study." *IEEE Internet of Things Journal*, **7**(3):2469–2482, 2019.

[Clo20] Open AR Cloud. "Open AR Cloud." `https://www.openarcloud.org/`, Mar. 2020.

[CR19] Jiasi Chen and Xukan Ran. "Deep learning with edge computing: A review." *Proceedings of the IEEE*, **107**(8):1655–1674, 2019.

[CRD15] Tiffany Yu-Han Chen, Lenin Ravindranath, Shuo Deng, Paramvir Bahl, and Hari Balakrishnan. "Glimpse: Continuous, real-time object recognition on mobile devices." In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, pp. 155–168, 2015.

[DAS20] Jean-François Determe, Sophia Azzagnuni, Utkarsh Singh, François Horlin, and Philippe De Doncker. "Collisions of uniformly distributed identifiers with an application to MAC address anonymization." *arXiv preprint arXiv:2009.09876*, 2020.

[Day] Google Daydream. `https://arvr.google.com/daydream/`.

[DLH20] Haotian Deng, Qianru Li, Jingqi Huang, and Chunyi Peng. "Icellspeed: Increasing cellular data speed with device-assisted cell selection." In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, pp. 1–13, 2020.

[DPW16] Adrian Dabrowski, Georg Petzl, and Edgar R Weippl. "The messenger shoots back: Network operator based IMSI catcher detection." In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pp. 279–302. Springer, 2016.

[EAW21] Mitziu Echeverria, Zeeshan Ahmed, Bincheng Wang, M Fareed Arif, Syed Rafiul Hussain, and Omar Chowdhury. "PHOENIX: Device-centric cellular network protocol monitoring using runtime verification." *arXiv preprint arXiv:2101.00328*, 2021.

[Fis14] Sven Fischer. "Observed Time Difference Of Arrival (OTDOA) Positioning in 3GPP LTE." `https://www.qualcomm.com/media/documents/files/otdoa-positioning-in-3gpp-lte.pdf`, Jun 2014.

[Flo] "Flora: Flexible Mobile Network Platform." `http://metro.cs.ucla.edu/flora.html`.

[FW19] Robert Falkenberg and Christian Wietfeld. "FALCON: An accurate real-time monitor for client-based mobile network data analytics." In *2019 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7. IEEE, 2019.

[GGS16] Ismael Gomez-Miguelez, Andres Garcia-Saavedra, Paul D Sutton, Pablo Serrano, Cristina Cano, and Doug J Leith. "srsLTE: An open-source platform for LTE evolution and experimentation." In *Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization*, pp. 25–32, 2016.

[GJB17] Giulio Grassi, Kyle Jamieson, Paramvir Bahl, and Giovanni Pau. "Parkmaster: An in-vehicle, edge-based video analytics service for detecting open parking spaces in urban environments." In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, pp. 1–14, 2017.

[Glo21] GlobeNewswire. "Global 5G Connections Are Growing Rapidly.", September 2021. `https://www.globenewswire.com/news-release/2021/09/22/2301608/0/en/Global-5G-Connections-Are-Growing-Rapidly.html`.

[Goo20] Google. "Google API for Android.", Mar. 2020.

[Goo21] Google. " Transport Security State Static ." https://chromium.googlesource.com/chromium/src/net/+/master/http/transport_security_state_static.json, Jan 2021.

[GSM21] GSMA. "Mobile IoT LPWA - LTE-M & NB-IoT Commercial Launches: GSMA." https://www.gsma.com/iot/mobile-iot-commercial-launches, Mar 2021.

[HCM18] Syed Hussain, Omar Chowdhury, Shagufta Mehnaz, and Elisa Bertino. "LTEInspector: A systematic approach for adversarial testing of 4G LTE." In *Network and Distributed Systems Security (NDSS) Symposium 2018*, 2018.

[HEC19] Syed Rafiul Hussain, Mitziu Echeverria, Omar Chowdhury, Ninghui Li, and Elisa Bertino. "Privacy Attacks to the 4G and 5G Cellular Paging Protocols Using Side Channel Information." In *NDSS*, volume 19, pp. 24–27, 2019.

[HEK19] Syed Rafiul Hussain, Mitziu Echeverria, Imtiaz Karim, Omar Chowdhury, and Elisa Bertino. "5GReasoner: A Property-Directed Security and Privacy Analysis Framework for 5G Cellular Network Protocol." In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 669–684, 2019.

[HJB12] Jeff Hodges, Collin Jackson, and Adam Barth. "RFC 6797: HTTP Strict Transport Security (HSTS).", 2012.

[HRO16a] Silke Holtmanns, Siddharth Prakash Rao, and Ian Oliver. "User location tracking attacks for LTE networks using the interworking functionality." In *2016 IFIP Networking conference (IFIP Networking) and workshops*, pp. 315–322. IEEE, 2016.

[HRO16b] Silke Holtmanns, Siddharth Prakash Rao, and Ian Oliver. "User location tracking attacks for LTE networks using the interworking functionality." In *2016 IFIP Networking Conference (IFIP Networking) and Workshops*, pp. 315–322, 2016.

[Ipe] Iperf3. https://github.com/esnet/iperf.

[KFS18] Seiran Khaledian, Farhad Farzami, Besma Smida, and Danilo Erricolo. "Inherent Self-Interference Cancellation for In-Band Full-Duplex Single-Antenna Systems." *IEEE Transactions on Microwave Theory and Techniques*, **66**(6):2842–2850, 2018.

[KRH19] Katharina Kohls, David Rupprecht, Thorsten Holz, and Christina Pöpper. "Lost traffic encryption: fingerprinting LTE/4G traffic on layer two." In *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*, pp. 249–260, 2019.

[Lab17] Kaspersky Lab. "Rooting your Android: Advantages, disadvantages, and snags." https://www.kaspersky.com/blog/android-root-faq/17135/, Jun. 2017.

[LCC15] Kyungmin Lee, David Chu, Eduardo Cuervo, Johannes Kopf, Yury Degtyarev, Sergey Grizan, Alec Wolman, and Jason Flinn. “Outatime: Using speculation to enable low-latency continuous interaction for mobile cloud gaming.” In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 151–165, 2015.

[LHC19] Zeqi Lai, Y Charlie Hu, Yong Cui, Linhui Sun, Ningwei Dai, and Hung-Sheng Lee. “Furion: Engineering high-quality immersive virtual reality on today’s mobile devices.” *IEEE Transactions on Mobile Computing*, 2019.

[LJL16] Marc Lichtman, Roger Piqueras Jover, Mina Labib, Raghunandan Rao, Vuk Marojevic, and Jeffrey H Reed. “LTE/LTE-A jamming, spoofing, and sniffing: threat assessment and mitigation.” *IEEE Communications Magazine*, **54**(4):54–61, 2016.

[LPY16] Yuanjie Li, Chunyi Peng, Zengwen Yuan, Jiayao Li, Haotian Deng, and Tao Wang. “Mobileinsight: Extracting and analyzing cellular network information on smartphones.” In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*, pp. 202–215, 2016.

[LPZ21] Yuanjie Li, Chunyi Peng, Zhehui Zhang, Zhaowei Tan, Haotian Deng, Jinghao Zhao, Qianru Li, Yunqi Guo, Kai Ling, Boyan Ding, et al. “Experience: a five-year retrospective of MobileInsight.” In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, pp. 28–41, 2021.

[LYP17] Yuanjie Li, Zengwen Yuan, and Chunyi Peng. “A control-plane perspective on reducing data access latency in LTE networks.” In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, pp. 56–69, 2017.

[LZG14] Muyuan Li, Haojin Zhu, Zhaoyu Gao, Si Chen, Le Yu, Shangqian Hu, and Kui Ren. “All your location are belong to us: Breaking mobile social networks for automated user location tracking.” In *Proceedings of the 15th ACM international symposium on Mobile ad hoc networking and computing*, pp. 43–52, 2014.

[Mar09] Moxie Marlinspike. “New tricks for defeating SSL in practice.” *Black Hat DC*, **2**, 2009.

[MO17] Stig F Mjølsnes and Ruxandra F Olimid. “Easy 4G/LTE IMSI catchers for non-programmers.” In *International Conference on Mathematical Methods, Models, and Architectures for Computer Network Security*, pp. 235–246. Springer, 2017.

[One] HackRF One. `https://greatscottgadgets.com/hackrf/one/`.

[Ope18] OpenAirInterface. “OpenAirInterface Official Website.”, April 2018. http://www.openairinterface.org.

[PAC00] Vern Paxson, Mark Allman, Jerry Chu, and Matt Sargent. “RFC 2988: Computing TCP’s retransmission timer.”, 2000.

[Pos18] Positive Technologies. "Diameter Vulnerabilities Exposure Report.", 2018.

[Pos20] Positive Technologies. "Security assessment of Diameter networks.", 2020.

[PUB20] PUBG. "PUBG." `https://www.pubg.com/`, Mar. 2020.

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

[Rad] USRP Software Defined Radio. `https://www.ettus.com/products/`.

[Rep] Unity Technologies Report. `https://www.tweaktown.com/news/74682/index.html`.

[Res00] Eric Rescorla et al. "RFC 2818: HTTP over TLS.", 2000.

[Res21] Precedence Research. "5G Services Market.", December 2021. `https://www.precedenceresearch.com/5g-services-market`.

[RJP16] David Rupprecht, Kai Jansen, and Christina Pöpper. "Putting LTE Security Functions to the Test: A Framework to Evaluate Implementation Correctness." In *10th USENIX Workshop on Offensive Technologies (WOOT 16)*, 2016.

[RKH16] Siddharth Prakash Rao, Bhanu Teja Kotte, and Silke Holtmanns. "Privacy in LTE Networks." In *Proceedings of the 9th EAI International Conference on Mobile Multimedia Communications*, pp. 176–183, 2016.

[RKH19] David Rupprecht, Katharina Kohls, Thorsten Holz, and Christina Pöpper. "Breaking LTE on layer two." In *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 1121–1136. IEEE, 2019.

[RKH20a] David Rupprecht, Katharina Kohls, Thorsten Holz, and Christina Pöpper. "Call Me Maybe: Eavesdropping Encrypted {LTE} Calls With {ReVoLTE}." In *29th USENIX security symposium (USENIX security 20)*, pp. 73–88, 2020.

[RKH20b] David Rupprecht, Katharina Kohls, Thorsten Holz, and Christina Pöpper. "IMP4GT: Impersonation attacks in 4G networks." In *Symposium on Network and Distributed System Security (NDSS). ISOC*, 2020.

[SBA15] Altaf Shaik, Ravishankar Borgaonkar, N Asokan, Valtteri Niemi, and Jean-Pierre Seifert. "Practical attacks against privacy and availability in 4G/LTE mobile communication systems." *NDSS*, 2015.

[SBP18] Altaf Shaik, Ravishankar Borgaonkar, Shinjo Park, and Jean-Pierre Seifert. "On the impact of rogue base stations in 4G/LTE self organizing networks." In *Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pp. 75–86, 2018.

[Sch13] Rhode & Schwarz. "LTE: System Specifications and Their Impact on RF & Base Band Circuits." `https://cdn.rohde-schwarz.com/pws/dl_downloads/dl_application/application_notes/1ma221/1MA221_0e.pdf`, Apr 2013.

[SGJ19] Shu Shi, Varun Gupta, and Rittwik Jana. "Freedom: Fast recovery enhanced vr delivery over mobile networks." In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 130–141, 2019.

[SHZ18] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. "Mobilenetv2: Inverted residuals and linear bottlenecks." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

[Sir20] Apple Siri. "Siri." `https://www.apple.com/siri/`, Mar. 2020.

[Son] Project Sonica. "Sonica: Software-defined Open NB-IoT Research Platform." `http://metro.cs.ucla.edu/sonica.html`.

[SRS] SRSRAN. `https://www.srslte.com/`.

[STM21] STMicroelectronics. "LTE Cellular to Cloud Pack with STM32L496AG MCU." `https://www.st.com/en/evaluation-tools/p-l496g-cell02.html`, Jan 2021.

[Str] Unity Render Streaming. `https://github.com/Unity-Technologies/UnityRenderStreaming`.

[sys16] sysmocom. "sysmoUSIM-SJS1 SIM + USIM Card (10-pack)." `http://shop.sysmocom.de/products/sysmousim-sjs1`, 2016.

[TDZ21] Zhaowei Tan, Boyan Ding, Jinghao Zhao, Yunqi Guo, and Songwu Lu. "Data-plane signaling in cellular IoT: attacks and defense." In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, pp. 465–477, 2021.

[TLL18] Zhaowei Tan, Yuanjie Li, Qianru Li, Zhehui Zhang, Zhehan Li, and Songwu Lu. "Supporting mobile VR in LTE networks: How close are we?" *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, **2**(1):1–31, 2018.

[TZL21] Zhaowei Tan, Jinghao Zhao, Yuanjie Li, Yifei Xu, and Songwu Lu. " Device-Based LTE Latency Reduction at the Application Layer." In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2021.

[Ube20] Uber. "Uber." `https://www.uber.com/`, Mar. 2020.

[ubl20] ublox. "Even with 5G on the horizon, 4G LTE-M and NB-IoT devices show no signs of going obsolete." `https://www.u-blox.com/en/blogs/insights/even-5g-horizon-4g-lte-m-and-nb-iot-devices-show-no-signs-going-obsolete`, Apr 2020.

[Uni] WebRTC for Unity. `https://github.com/Unity-Technologies/com.unity.webrtc`.

[Vir12] Network Functions Virtualisation. “An introduction, benefits, enablers, challenges & call for action.” In *White Paper, SDN and OpenFlow World Congress*, 2012.

[w3t21a] w3techs. “Usage statistics of default protocol https for websites .” `https://w3techs.com/technologies/details/ce-httpsdefault`, Jan 2021.

[w3t21b] w3techs. “Usage statistics of HTTP Strict Transport Security for websites .” `https://w3techs.com/technologies/details/ce-hsts`, Jan 2021.

[Waz20] Waze. “Waze.” `https://www.waze.com/`, Mar. 2020.

[XZZ20] Dongzhu Xu, Anfu Zhou, Xinyu Zhang, Guixian Wang, Xi Liu, Congkai An, Yiming Shi, Liang Liu, and Huadong Ma. “Understanding operational 5G: A first measurement study on its coverage, performance and energy consumption.” In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pp. 479–494, 2020.

[YBS19] Hojoon Yang, Sangwook Bae, Mincheol Son, Hongil Kim, Song Min Kim, and Yongdae Kim. “Hiding in Plain Signal: Physical Signal Overshadowing Attack on LTE.” In *28th USENIX Security Symposium (USENIX Security 19)*, pp. 55–72, 2019.

[YC17] Ming Yang and Tom Chin. “Scheduling request during connected discontinuous reception off period.”, July 20 2017. US Patent App. 14/996,153.

[Zer22] ZeroMQ. “ZeroMQ: An open-source universal messaging library .” `https://zeromq.org/`, Jan 2022.

[ZJZ18] Zhou Zhuang, Xiaoyu Ji, Taimin Zhang, Juchuan Zhang, Wenyuan Xu, Zhenhua Li, and Yunhao Liu. “FBSleuth: Fake base station forensics via radio frequency fingerprinting.” In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pp. 261–272, 2018.