

UCLA

UCLA Electronic Theses and Dissertations

Title

Enhancing System Resiliency for 5G and 5G IoT: A Plug-and-Play, SIM/eSIM-based Approach

Permalink

<https://escholarship.org/uc/item/9sg6x9g2>

Author

Zhao, Jinghao

Publication Date

2023

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Enhancing System Resiliency for 5G and 5G IoT: A Plug-and-Play, SIM/eSIM-based
Approach

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

by

Jinghao Zhao

2023

© Copyright by
Jinghao Zhao
2023

ABSTRACT OF THE DISSERTATION

Enhancing System Resiliency for 5G and 5G IoT: A Plug-and-Play, SIM/eSIM-based
Approach

by

Jinghao Zhao

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 2023

Professor Songwu Lu, Chair

5G and 5G IoT technologies have been widely deployed in both consumer and industrial usage scenarios for their ubiquitous coverage, mobility support, and carrier-grade services. The resiliency of such systems is critical to sustaining billions of smartphones and IoT devices. We have identified three roadblocks hindering 5G resiliency: wireless outages, 5G software stack failures, and malicious attacks. To ensure high resiliency and build a highly available and secure 5G service, we have challenged three commonly held perceptions: 1) current failure handling at the infrastructure, modem, OS, and applications is sufficient for high resiliency; 2) providing 100% wireless coverage ensures the best availability; 3) existing 5G security schemes, protected by SIM keys, can shield devices from attacks. Our study thus invalidates all three. The fundamental root cause stems from the complex, software-hardware interactions between devices, infrastructure, and protocol stacks. This complexity affects data transmission, exacerbates failure diagnosis and handling, and exposes new vulnerabilities to attackers.

This dissertation introduces a novel SIM/eSIM-based solution to enhance resiliency in 5G

and 5G IoT systems. By utilizing the SIM card and eSIM chip as an independent miniature system, we provide plug-and-play, highly resilient 5G services without modifications to device firmware, operating systems, or base stations. We prototype and evaluate our proposals using commodity 5G devices with three major US carriers. Our solution addresses all three roadblocks of high resiliency: 1) we develop a novel SIM-based 5G failure diagnosis and handling system, resulting in 0.6x-792x disruption reduction under 5G software failures. 2) we enable rapid inter-carrier switching, reducing unavailability under outages by 28x, while only increasing power consumption by 4.7%. 3) we identify new 5G/4G attacks, such as traffic eavesdropping, man-in-the-middle attacks, and impersonation. To mitigate these threats, we offer both authentication and fine-grained access control for SIM/eSIM, ensuring security with minimal authentication latency (5.5%) and energy overhead (3.2%). By enabling intelligence on the device side with SIM/eSIM, we enhance system resiliency and address the aforementioned roadblocks.

Rethinking the “smart core, dumb terminal” design philosophy in 5G systems allows us to tap into the potential of end-user devices for improved resiliency. This traditional design principle emphasizes the importance of intelligent infrastructures while relegating end-user devices to relatively passive roles, thus neglecting their capacity to respond autonomously during service disruptions. This dissertation demonstrates that, augmenting device-side intelligence with SIMs offers a novel perspective on enhancing resiliency. SIMs enable simple, yet effective device-side handling for improved resilience by incorporating lightweight operations such as multi-tier resets and rapid multi-carrier switching. These operations allow SIMs to accelerate recovery from protocol failures, enhance connectivity outage handling, and strengthen security against vulnerabilities. By harnessing the potential of device-side intelligence, SIMs actively contribute to the resilience of both 5G and 5G IoT systems, paving the way for more robust and reliable communication networks.

The dissertation of Jinghao Zhao is approved.

Ravi Arun Netravali

Jeffrey Aaron Burke

Lixia Zhang

Songwu Lu, Committee Chair

University of California, Los Angeles

2023

TABLE OF CONTENTS

1	Introduction	1
1.1	Challenges for High Resiliency Today	3
1.2	Our Contribution	4
1.2.1	SEED: SIM-Based 5G Failure Diagnosis and Handling	5
1.2.2	SHIELD: Enhancing High 5G Availability with SIM	5
1.2.3	SecureSIM: SIM/eSIM Authentication and Access Control	6
1.3	Organization of the Dissertation	6
2	Resilient 5G Designs: State of the Arts and Limitations	8
2.1	5G System: An Overview	8
2.1.1	5G System Architecture	9
2.1.2	Protocol Stack	10
2.1.3	SIM and eSIM	11
2.1.4	Session Setup Procedures	12
2.1.5	Power Saving	13
2.1.6	5G Security Mechanisms	13
2.2	Practices for High Resiliency: State of the Arts	16
2.2.1	Device: Multi-layer Mechanisms for Resiliency	16
2.2.2	Infrastructure: Enhancing Control and Management for Resiliency	19
2.3	Insufficient Resiliency in Current 5G Architectures	21
3	Overview	22

3.1	Challenges for High Resiliency	22
3.2	Design Space of SIM/eSIM	23
3.3	Overview of SIM-centric Designs	25
3.4	Roadmap	26
4	SEED: SIM-Based 5G Failure Diagnosis and Handling	27
4.1	Cellular Network Failures	27
4.2	Limitations of Current Solutions	28
4.2.1	Failures in the Real World	28
4.2.2	Limitations of Modem Scheme	30
4.2.3	Android: Insufficient Diag & Handling	32
4.2.4	Solution Space	34
4.3	SEED: SIM-Based Failure Diagnosis	35
4.3.1	Case for SIM-based Solution	35
4.3.2	SEED Overview	36
4.3.3	Lightweight SIM Diagnosis	37
4.3.4	SIM-Based Failure Handling	39
4.3.5	Real-Time SIM-Network Collaboration	43
4.4	Enhanced Failure Management	46
4.4.1	Insufficient Standardized Causes	46
4.4.2	Infra-Assisted SIM Diagnosis	46
4.4.3	SIM Handling with Online Learning	48
4.5	Implementations	49
4.6	Evaluation	51

4.6.1	Overall Performance	53
4.6.2	Micro-Benchmarks	56
4.6.3	Security Analysis	59
4.7	Discussion	59
4.8	Related Work	61
4.9	Conclusion	62
5	SHIELD: Enhancing High 5G Availability with SIM	63
5.1	Cloud-based 5G IoT System	63
5.1.1	Current Practice on 5G IoT Availability	65
5.2	5G IoT Availability	66
5.2.1	Wireless-conditioned Availability	67
5.2.2	Prevalent Wireless Outage	68
5.2.3	Interrupted Device Sync Up with Power-saving	69
5.3	SHIELD Design	70
5.3.1	Overview	71
5.3.2	Rapid Network Switching	72
5.3.3	Handling Power-saving Interruptions	78
5.4	Implementations	82
5.5	Evaluation	85
5.5.1	Availability Improvement	86
5.5.2	Power Consumption	89
5.5.3	Choice of Parameter T	90
5.5.4	Availability on Real-world IoT Applications	91

5.6	Discussion: The Road to Five Nines	92
5.7	Related Work	93
5.8	Conclusion	94
6	SecureSIM: SIM/eSIM Authentication and Access Control	96
6.1	Vulnerability & Threat Models	97
6.1.1	Vulnerabilities	97
6.1.2	Threat Model	99
6.2	Attacks	100
6.2.1	One-time Access Attack	100
6.2.2	Traffic Eavesdropping	103
6.2.3	MitM Attack	106
6.2.4	Impersonation	109
6.3	SecureSIM Design	111
6.3.1	Control SIM Access with Graph	112
6.3.2	Rethink SIM with Certificates	116
6.3.3	Security Analysis	121
6.4	Implementation	123
6.5	Evaluation	124
6.6	Discussion	126
6.7	Related Work	127
6.8	Conclusion	128
7	Highly Resilient Open-Source 5G Platform	130

7.1	eSIM Platform	130
7.2	Mobile AR/VR Platform	133
7.2.1	System Components	133
7.2.2	Enabling AR with 5G IoT	135
8	Conclusion and Future Work	136
8.1	Summary of Results	137
8.2	Lessons Learned	138
8.2.1	Emphasizing Device-side Intelligence for 5G Resiliency	138
8.2.2	High Resiliency Is Beyond Enhancing Wireless	139
8.2.3	Optimization Can Be Plug-and-Play	140
8.3	Looking Forward	140
8.3.1	State Snapshot for Resilient 5G/B5G Core Design	141
8.3.2	Trusted Access for Resilient Wireless Fusion	142
8.3.3	Resilient AI/ML Services for Cellular Devices	143
	Appendix A Supplementary Materials for Chapter 4	144
A.1	Standardized Failure Causes Related to Configuration Issues	144
A.2	AT Commands List for Fast Failure Handling	145
	References	147

LIST OF FIGURES

2.1	5G system architecture.	9
2.2	5G session setup procedures.	12
2.3	5G key architecture.	14
2.4	Authentication procedures.	15
4.1	Disruption time with existing modem handling.	31
4.2	Android failure detection latency.	33
4.3	SEED diagram.	36
4.4	Multi-tier reset w/ and w/o root privilege.	40
4.5	Reset data plane without reattachment.	41
4.6	Collaboration with standard-complied signaling messages (a) Network to SIM (b) SIM to Network.	44
4.7	Low-overhead infrastructure assistance.	47
4.8	SEED implementation components.	51
4.9	Experimental testbed setup.	52
4.10	Diagnosis overhead on network/device side.	57
4.11	SIM-Infra collaboration latency.	58
4.12	Recovery time for multi-tier reset.	58
5.1	5G IoT system architecture.	64
5.2	Average disruption time (in seconds).	68
5.3	SHIELD overview.	71
5.4	Backup network discovery procedure.	73

5.5	Pipeline of SIM-based outage detection.	77
5.6	Data service setup time of three US operators.	81
5.7	Mobility-induced out-of-service downtime.	81
5.8	Components in SHIELD implementation.	82
5.9	Experimental setup.	85
5.10	Average disruption time of five solutions.	87
5.11	Wireless-conditioned availability vs. outage frequencies.	87
5.12	Daily energy consumption (mAh/day).	88
5.13	Power consumption vs. outage frequencies.	88
5.14	Energy overhead vs. threshold T.	90
5.15	Availability vs. threshold T.	90
5.16	Deployments in different scenarios.	91
6.1	SIM/eSIM usage survey.	98
6.2	SIM sticker.	101
6.3	Malware attack APDU traces.	103
6.4	Attacker testbed setup.	105
6.5	Eavesdropping implementation.	106
6.6	Eavesdropped traffic decryption.	106
6.7	Context-based MitM.	108
6.8	MitM attack validation.	109
6.9	Impersonation validation.	111
6.10	Managing SIM file access for off-card units and in-card applets with access graph.	112
6.11	SIM certificate chain.	117

6.12	SIM-Modem APDU encryption.	119
6.13	Automated verification procedure	120
6.14	Performance comparison.	124
6.15	Certificate verification	125
6.16	Encryption.	125
7.1	eSIM platform.	131
7.2	Mobile AR/VR platform.	134

LIST OF TABLES

4.1	Top 5 failure causes in control/data plane.	30
4.2	Solution comparison for 5G failure diagnosis & handling.	32
4.3	Failure dandling decisions with diagnosis results.	42
4.4	Disruption (s) percentile with legacy handling and SEED.	53
4.5	Average app disruption (s) with legacy (Leg.) failure handling, SEED-U (S.U) and SEED-R (S.R).	55
5.1	Compared solutions in evaluation.	86
5.2	Wireless-conditioned availability (%) of five solutions under different mobility scenarios.	88
6.1	Requirements of hardware and software attacks.	100
6.2	SIM profile node types.	113
6.3	Access schema.	114
6.4	Access policy examples.	115

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor, Prof. Songwu Lu, for his guidance, support, and encouragement throughout my Ph.D. journey. His intellectual curiosity, passion for research, and commitment to academic excellence have been truly inspiring, and his mentorship has been instrumental in shaping my growth as a researcher. I am honored to have had the opportunity to learn from him and work together. Simultaneously, he encourages me to push my boundaries and fosters stimulating conversations when I encounter obstacles. The assistance and encouragement have been indispensable, and I will remain forever grateful for the guidance.

I would also like to extend my sincere appreciation to my thesis committee members, Prof. Lixia Zhang, Prof. Jeff Burke, and Prof. Ravi Netravali, for their valuable insights and constructive feedback on my academic progress. Their expertise in their respective fields has not only helped refine my research but also broadened my understanding of the interdisciplinary nature of the work. I am truly grateful for their time, effort, and patience in guiding me through this challenging yet rewarding journey.

I am immensely grateful to my labmates, Dr. Zhehui Zhang, Dr. Qianru Li, Dr. Zhaowei Tan, Yunqi Guo, Boyan Ding, Yifei Xu, Zengwen Yuan, Muhammad Taqi Mehdi, and Kaiyuan Chen, for their collaboration and friendship. Their collective knowledge, diverse perspectives, and unwavering support have enriched my research experience and made my time in the lab both enjoyable and fruitful. I am honored to have worked alongside such a talented and dedicated group of individuals, and I will cherish the memories we have shared throughout our time together.

I am deeply grateful for the opportunity to collaborate with numerous brilliant researchers and colleagues from various institutions. I have gleaned much from Prof. Yuanjie Li and Prof. Chunyi Peng. Their dedication, professionalism, and willingness to share their knowledge have enriched my research and life. The summer internships were thoroughly enjoyable. I

am truly fortunate to have had the chance to work together, and I extend my thanks to Alex Rodriguez, Dr. Ulas Kozat, Nick Yurchenko, and Pravin Shelar for their help.

I cannot express enough gratitude for those who supported me in beginning my academic journey during my undergraduate. I am deeply thankful to Prof. Xinbing Wang and Prof. Luoyi Fu at SJTU, who sparked my passion for system research, guided my undergraduate thesis, assisted me in writing my first paper, and supported me as I took my initial steps into research. I also extend my profound gratitude to all the teachers who educated me, believed in me, and prepared me for the subsequent stages of my journey.

Furthermore, I would like to thank my friends, both inside and outside the academic realm, who have been sources of support, encouragement, and motivation throughout my Ph.D. journey. Their companionship helped me navigate the inevitable highs and lows of this endeavor. I am incredibly grateful for their presence in my life, as they have made my journey all the more meaningful and memorable.

Finally, I would like to express my heartfelt gratitude to my parents, Xiaohong and Ruiming, for their unwavering love, support, and belief in my abilities. Their sacrifices, encouragement, and understanding have been a constant source of strength and motivation throughout my life. They have instilled in me the value of perseverance, hard work, and passion for learning, and I am eternally grateful for their invaluable guidance and inspiration. I also want to extend a special thank you to my loving girlfriend, who has been my rock and confidante throughout this process. Her understanding and endless encouragement have been a beacon of light during challenging times, and her companionship has made the journey all the more rewarding. I am truly blessed to have them by my side and dedicate this accomplishment to them.

VITA

2014–2018 B.E., Electrical and Electronic Engineering, SJTU, Shanghai, China

2018–2023 Graduate Student Researcher, Computer Science, UCLA.

2019–2022 Teaching Assistant, Computer Science, UCLA.

2022 Summer Software Engineer Intern, Meta Platforms, Inc., Palo Alto, CA

PUBLICATIONS

Jinghao Zhao, Boyan Ding, Yifei Xu, Songwu Lu. SHIELD: Enabling Energy-efficient Highly Available 5G IoT with SIM Card, Under review, ACM SOSP 2023.

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

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

Jinghao Zhao, Qianru Li, Zengwen Yuan, Zhehui Zhang, Songwu Lu. 5G Messaging: System Insecurity and Defenses, IEEE CNS 2022.

Zhaowei Tan, **Jinghao Zhao**, Boyan Ding, Songwu Lu. CellDAM: User-Space, Rootless Detection and Mitigation for 5G Data Plane, USENIX NSDI 2023.

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.

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

Kaiyuan Chen, **Jinghao Zhao**. Skip The Question You Don't Know: An Embedding Space Approach, IJCNN 2019.

Zhaowei Tan, **Jinghao Zhao**, Yuanjie Li, Yifei Xu, Songwu Lu. LDRP: Device-Centric Latency Diagnostic and Reduction for Cellular Networks without Root, IEEE Transactions on Mobile Computing (TMC) 2023.

Zhehui Zhang, Yuanjie Li, Qianru Li, **Jinghao Zhao**, Ghufraan Baig, Lili Qiu, Songwu Lu. Movement-Based Reliable Mobility Management for Beyond 5G Cellular Networks, IEEE/ACM Transactions on Networking (TON) 2022.

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

CHAPTER 1

Introduction

5G and 5G IoT technologies have been widely deployed in both consumer and industrial domains because of their expansive coverage, mobility support, and carrier-grade services. This has enabled new opportunities in customer markets, such as faster download and upload speeds, reduced latency [LPZ21], real-time high-quality video streaming [DLH20], and seamless online gaming experiences [TZL21]. Beyond smartphones, 5G Internet-of-Things (IoT) technology has seen widespread adoption, facilitating the interconnection of everyday objects through the Internet. In recent years, IoT systems have been implemented in various scenarios, including smart cities [KRM17], environmental monitoring [AY19], and asset tracking [RVS20]. As the demand for connectivity and the number of connected devices continues to grow, 5G and 5G IoT hold the potential to enable more intimate interactions between users and the physical world in cyberspace.

As 5G and 5G IoT systems expand into mission-critical and enterprise settings, such as AR/VR/MR [TLL18], industrial applications [SHH20], and digital twins [MLC20], ensuring high resiliency becomes both crucial and pressing. Recent studies indicate that back-end cloud services offer three to four nines of availability [Mic23a, AWS23a], while front-end devices demonstrate durability and reliability [EFE23]. Various schemes are implemented on both hardware [TZD23] and software [TDZ21b] levels to ensure security within the infrastructure and devices. Current 5G designs include several schemes to enhance resiliency, leading to three prevailing perceptions: (1) existing failure handling at the infrastructure, modem, OS, and application levels is sufficient for high 5G resiliency; (2) providing 100%

wireless coverage ensures optimal availability; (3) existing 5G security schemes, safeguarded by SIM keys, can effectively protect devices from attacks.

This thesis challenges these common perceptions regarding 5G resiliency through empirical results. First, failures in 5G networks have become the norm rather than exceptions, with an average of one failure event occurring every ten procedures of 5G/4G signaling. Second, even under ideal wireless conditions, power-saving schemes on 5G devices introduce interruptions, reducing availability to 89.9%—failing to achieve even one nine (90%). Third, we identify new vulnerabilities and attacks in 5G that compromise security through traffic eavesdropping, man-in-the-middle attacks, and impersonation. Ensuring high resiliency becomes urgent for 5G services.

The low resiliency in the current 5G network stems from its architectural design. A critical impediment to achieving highly resilient 5G systems lies in the widespread adherence to a “smart core, dumb terminal” philosophy, which emphasizes intelligent infrastructure while relegating end-user devices to passive roles. This approach has led to a predominant focus on network-side mechanisms for ensuring resiliency, thereby limiting the ability of devices to respond autonomously during service interruptions. By enhancing the intelligence of end-user devices, we can unlock their potential to actively contribute to network resilience and improve overall system performance.

To address these challenges, this dissertation proposes a SIM-centric design that enhances device intelligence and harnesses the untapped potential of end-user devices for system resilience. The SIM occupies a unique position within the 5G ecosystem, being both produced and managed by the network while residing on cellular devices. This dual role enables the SIM to act as an intermediary, facilitating device-side intelligence with network knowledge by simple, yet effective operations, and thereby ensuring high resiliency in 5G and 5G IoT systems. We designate the SIM as a mini-system that can be integrated into 5G and 5G IoT devices in a plug-and-play manner without altering the device operating system or firmware. The SIM mini-system effectively offloads the tasks in achieving high resiliency, such as failure

diagnosis and recovery, power-saving management, and security protection. Specifically, we present three major sources hindering the 5G resiliency in §1.1 and subsequently introduce innovative SIM/eSIM-based solutions in §1.2.

1.1 Challenges for High Resiliency Today

Prevalent 5G software protocol failures As system scales increase and small cells are deployed, users and their smartphone applications regularly experience failures. We investigate the diagnosis and treatment of network failures in both control and data planes within the 5G protocol stack. Our analysis of public 5G traces reveals 2,832 failure cases identified from 24k control/data-plane management procedures. Other prior studies [LLL21] also confirmed the prevalence of failures in 5G. If unaddressed, these failures can cause lengthy disruptions for 5G-based Internet access and hinder the proper functioning of 5G applications. Existing solutions to 5G failures adopt either modem-based schemes [3GP21b] or OS-centric approaches [And22e] at the device level. They rely on timeout-based detection with multiple timers and employ a sequential retry approach for failure recovery. However, these detection and reaction schemes are ill-suited for complex 5G failure cases, resulting in prolonged service disruptions ranging from tens of seconds to tens of minutes. The fundamental issue is that 5G failures are highly diversified, stemming from a wide spectrum of control and data-plane management and data packet delivery. Without cooperation from the infrastructure and the device for fine-grained diagnosis, neither approach can determine error causes, resorting to a blind, sequential retry scheme for failure management.

Connectivity outages from both wireless and power-saving FCC datasets reveal that a single carrier’s coverage can be as low as 40% in rural areas [FCC21], with frequent wireless outages impacting 5G resiliency. Our study further demonstrates that even under ideal wireless conditions, availability can only reach a maximum of 89.9%, failing to attain even one nine (§5.2.3). The root causes are diverse, ranging from wireless outages to mobility

and power-saving induced out-of-service or control session terminations. The fundamental issue lies in the inadequacy of a single-carrier approach and the state inconsistency that arises between IoT devices and 5G infrastructure during device mobility or power-saving modes.

Vulnerability in current security design The interoperability between various entities, such as the SIM/eSIM, modem, and protocol stacks, exposes vulnerabilities that jeopardize 5G network security. We identify three vulnerabilities in current SIM/eSIM practices. Firstly, while the current SIM offers simple PIN-based access control, it fails to provide sufficient protection, allowing adversaries to easily access sensitive in-SIM information through hardware or software. Secondly, the interaction between the SIM/eSIM and the 5G/4G protocol stack maintains a security context that can be leaked and reused. Lastly, the communication channel between the modem and the SIM/eSIM remains unencrypted, enabling eavesdropping through malicious hardware. The fundamental issue is that PIN-based, coarse-grained access control is inadequate for protecting SIM/eSIM information from hardware or malware. Additionally, the current SIM/eSIM fails to provide proper authentication for various in-card applets and off-card units legitimately seeking access to in-SIM information, as well as for adversaries.

1.2 Our Contribution

Our study reveals the diverse factors that impede high resiliency in 5G networks. The fundamental cause originates from the intricate software-hardware interactions between devices, infrastructure, and protocol stacks. This complexity not only affects data transmission but also complicates failure diagnosis and handling, while exposing new vulnerabilities to attackers. To address these challenges, we propose novel SIM/eSIM-centric designs that ensure high resiliency for 5G services.

1.2.1 SEED: SIM-Based 5G Failure Diagnosis and Handling

To handle prevalent 5G protocol stack failures, we present SEED, a novel SIM/eSIM-based, pure software solution for failure diagnosing and handling. By leveraging information from the device and the 5G network, SEED offers fine-grained runtime failure detection and recovery using a domain-specific decision tree and online learning algorithms. It utilizes simple multi-tier resets for control-plane, data-plane, and data delivery failure recovery. Designed for fast deployment and compatibility with the ongoing 5G global rollout, SEED can be readily installed during subscriber activation and offers two modes with and without root privilege. SEED poses no new security threats as it works within the 5G framework without requiring changes to the current device firmware or infrastructure hardware. Our evaluations show that, the disruption time under protocol failures is reduced by a factor of $0.6\times\sim 792\times$, from $12.4\sim 476.0$ s by the current modem/OS-based schemes to a mere $0.4\sim 8.0$ s.

1.2.2 SHIELD: Enhancing High 5G Availability with SIM

From the availability perspective, 5G connectivity becomes the system resiliency bottleneck. We introduce SHIELD, a SIM/eSIM-based solution for achieving high availability in 5G IoT systems. SHIELD leverages the SIM card or eSIM chip and exploits a plugged-in, miniature, software-defined receiver hardware at the IoT device. SHIELD employs two innovative strategies: rapid inter-carrier-network switching at the device, which enables fast handling upon outages, and a data-first approach that initiates secure IoT data transfer without establishing the data plane. By piggybacking data into control signals, it enables concurrent data/control transfer using standardized 5G mechanisms. SHIELD is compatible with commodity IoT devices and existing 5G infrastructure, requiring no changes in device firmware, operating systems, applications, or 5G infrastructure. Our implementation and evaluation of SHIELD on commercial IoT devices over operational 5G networks demonstrate a reduction in data access downtime by up to $28\times$ with 4.7% extra energy overhead at the

device, achieving 99.3% availability in two showcase IoT applications and outperforming existing solutions while preserving energy efficiency.

1.2.3 SecureSIM: SIM/eSIM Authentication and Access Control

We investigate SIM/eSIM security in 5G/4G cellular networks, identifying three vulnerabilities in current practices: inadequate PIN-based control, leaked security context, and unencrypted communication between the modem and SIM. Exploiting these vulnerabilities, we devise several practical attacks, including traffic eavesdropping, man-in-the-middle attacks, and impersonation. To counter these threats, we propose a novel SIM/eSIM solution, SecureSIM, for authentication and fine-grained access control, replacing PIN and key-based authentication with a certificate-based scheme that scales to hundreds of entities. Our approach organizes in-SIM files into a multi-rooted tree hierarchy for access policy specification and automated checking. Empirical evaluation demonstrates that the certificate verification and encryption components ensure security with marginal latency and energy overhead, offering a more secure and resilient solution for SIM/eSIM in 5G networks.

1.3 Organization of the Dissertation

This dissertation is organized into the following chapters. Chapter 2 provides an introduction to the architecture of 5G networks. It further introduces the current practices of 5G resiliency designs and their limitations.

Chapter 3 provides an overview of our proposed SIM-centric solutions. We detail the challenges and introduce our designs and insights.

Chapter 4 to 7 introduce our SIM/eSIM-based highly resilient 5G system design. Chapter 4 details a SIM-based approach for addressing 5G protocol stack failures; Chapter 5 presents a SIM-centric software solution with minimal hardware support for enhancing 5G availability; Chapter 6 investigates the vulnerabilities in current 5G/4G designs, proposing

authentication and fine-grained access control measures for SIM/eSIM. Chapter 7 introduces our open-source platform for demonstrating the highly resilient 5G system designs.

Finally, in Chapter 8, we summarize the results and lessons learned, and propose directions for future research.

CHAPTER 2

Resilient 5G Designs: State of the Arts and Limitations

Over the past decades, the mobile network has undergone significant evolution, resulting in substantial performance improvements with advances in wireless access technologies. Nowadays, enterprise and mission-critical applications impose new resiliency requirements for 5G services. However, the current “dumb terminal, smart core” nature of the 5G ecosystem makes it difficult to pinpoint resiliency issues. Our study shows that the current designs could not achieve a high resiliency, leading to widespread protocol failures, frequent connectivity outages, and security vulnerabilities.

In this chapter, we discuss the current 5G system architecture and schemes in §2.1. We examine existing solutions in §2.2 aimed at achieving high service resiliency in 5G systems. Finally, we summarize the limitations of these solutions and conclude the chapter in §2.3.

2.1 5G System: An Overview

5G provides enhanced mobile network performance by increasing throughput, reducing latency, and accommodating a wider variety of device types. In this section, we provide an overview of the current 5G system.

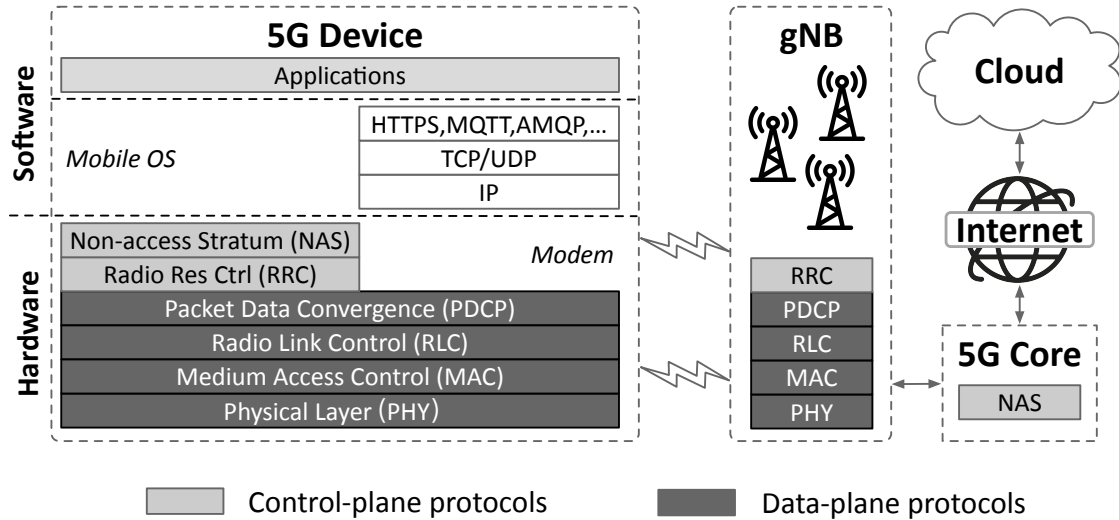


Figure 2.1: 5G system architecture.

2.1.1 5G System Architecture

As shown in Figure 2.1, the 5G device communicates with the 5G base station (gNB), which uses licensed frequency channels to transfer data and control signaling messages with the device. 5G Core (5GC) is in charge of providing user centralized management, including session/mobility management, user authentication, roaming, etc. The control plane and the data plane together facilitate seamless connectivity, high-speed data transmission, and enhanced network performance. The control plane primarily focuses on managing and orchestrating network functions such as session management, mobility, security, and quality of service, ensuring the efficient allocation of resources and optimal routing of data. On the other hand, the data plane is responsible for the actual transmission of user data and application traffic across the network, leveraging advanced technologies like network slicing and edge computing to provide ultra-low latency and high throughput. These two planes, working in tandem, enable 5G networks to support a diverse range of applications.

Inside 5G devices, there exists a crucial component known as a network adapter or modem, which is specifically designed to handle the complexities of 5G protocols and facilitate efficient cellular data transfer. This modem operates in conjunction with the device's mobile

operating system (OS), ensuring seamless communication between the hardware and software components. The mobile OS plays a critical role in managing the modem's functions, while also providing APIs to various applications running on the device. These APIs allow developers to leverage the power of 5G without needing to delve into the intricate details of the underlying network stacks. As a result, the complexities of the network stacks remain concealed from mobile application developers.

The 5G Internet-of-Things (IoT) shares the same 5G architecture with broadband devices (e.g., smartphones). Current IoT services are usually cloud-based. Its main function is to collect data from sensors through 5G network connectivity and perform data analytics in the cloud. The system has three components: front-end devices, back-end cloud, and network connectivity. The front-end devices use sensors to interact with the environment, collect data, and send them to the cloud. The back-end cloud stores and processes data. Network connectivity communicates front-end devices and the back-end cloud.

2.1.2 Protocol Stack

Figure 2.1 illustrates the 5G protocol stack. The control-plane signaling is managed by Non-access Stratum (NAS) and Radio Resource Control (RRC). NAS protocol allows message exchange between the 5G device (UE) and 5GC, while RRC establishes a channel between the UE and the gNB, responsible for data-plane parameters, such as set-up, power management, and handover behavior.

The data plane facilitates IP packet delivery, involving four protocols: Packet Data Convergence (PDCP), Radio Link Control (RLC), Medium Access Control (MAC), and Physical Layer (PHY). PDCP handles control-plane and data-plane packet encryption and integrity protection. RLC ensures reliable, in-order data transfer by performing data concatenation and reorganization. MAC manages radio access control, and PHY executes wireless signal processing.

2.1.3 SIM and eSIM

The Universal Integrated Circuit Card (UICC), commonly known as the SIM card, enables mobile users to access 5G/4G cellular networks by storing a SIM file package containing crucial parameters, such as the identity and configurations. The Embedded SIM (eSIM) is a recent innovation that enhances usability and programmability by integrating a programmable chip directly onto the device's circuit board. eSIM supports remote SIM provisioning, allowing users to download SIM file packages and switch between operators without physically changing SIM cards.

Operators or IoT service providers can offer additional services, such as geofencing, Quality of Service reporting, and SIM status checking [M2M], by installing applets on the SIM. An applet is a small application within the SIM that can communicate with the modem using Application Protocol Data Units (APDUs) to obtain device information, establish calls, and send Short Message Service (SMS) messages.

Current 5G security protection is built based on hierarchical keys. The root key K is stored in the SIM/eSIM. Both SIM and eSIM use various types of files as the basic units for storing the SIM package. A typical SIM includes around 200 files. Each file has a unique ID [3GP20a]. The operators control SIM files and are responsible for managing file access. The current SIM offers four access modes for files: (1) Always (ALW), with no restrictions. (2) PIN, where the cardholder needs to verify the PIN unless PIN verification is disabled. (3) ADM, where the administrative key (owned by the card issuer, e.g., operators) needs to be verified. (4) Never (NEV), with no access. For example, the international mobile subscriber identity (IMSI) is stored with ID 6F07 with ADM access mode so that the modem can only read it but cannot update it. Some other files that need updates from the modem during connection use PIN mode, such as the security context.

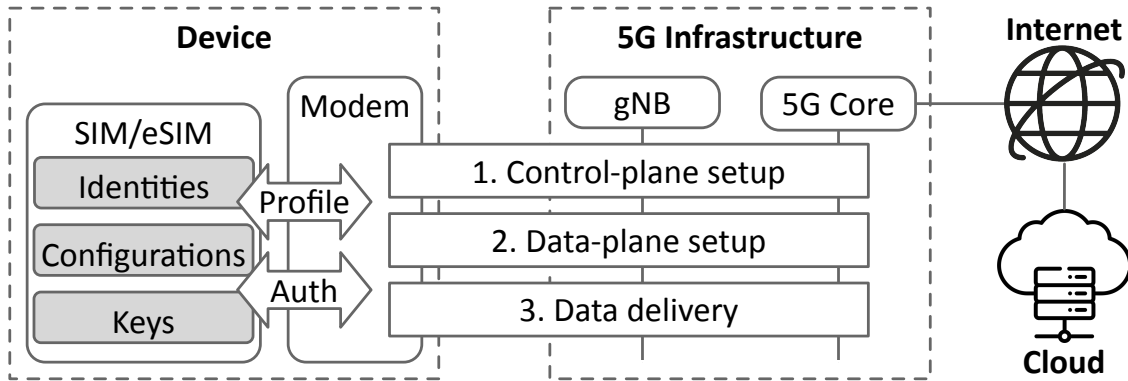


Figure 2.2: 5G session setup procedures.

2.1.4 Session Setup Procedures

In 5G networks, mobile devices utilize two essential modules for network access: the SIM/eSIM and the modem. The SIM/eSIM stores vital subscriber information such as user identities, configurations, and security keys. The modem retrieves this information from the SIM, registering the device on the network by running various applets on the SIM to manage profile transmissions and authentication. The modem and SIM communicate through the application protocol data unit (APDU), which can be considered a byte-stream interface. The modem utilizes it to load files and access the network with loaded files. The mobile operating system or applications can also send APDUs to the SIM through the modem. For example, operator applications could send APDUs to update network configurations in the SIM [UIC].

Data transfer in mobile devices occurs in three stages, as illustrated in Figure 2.2. Firstly, the modem establishes the control plane through 5G signaling messages, which involve identity exchange, mutual authentication between the SIM and the network, and location updates. Secondly, the modem and infrastructure exchange signaling messages to set up the data-plane¹, configuring elements such as the device IP address and the DNS server. The infrastructure comprises both the base stations (gNB) and the 5G core network. Finally, the

¹Referred to as bearer setup in 5G standards [3GP20d].

device initiates data packet delivery for its applications.

2.1.5 Power Saving

5G IoT devices necessitate the implementation of aggressive power-saving strategies to optimize energy conservation and prolong battery life. Two schemes, Discontinuous Reception (DRX) and Power Saving Mode (PSM), have been introduced in 5G standards [3GP21b] to meet these demands.

DRX is a technique that allows a device to enter a low-power sleep state during periods of inactivity. Instead of maintaining continuous connectivity, the device periodically “wakes up” for data transfer, significantly reducing power consumption. This intermittent reception of data enables the device to extend its battery life while still remaining responsive to network communications.

PSM is another power-saving approach that allows a device to enter a deep sleep mode, effectively disconnecting from the network. Unlike DRX, the device in PSM reconnects to the network only when necessary for data transfer, further conserving energy. This mode is particularly useful for IoT devices that require infrequent communication with the network. By employing these power-saving mechanisms, 5G IoT devices can achieve high energy efficiency and extend battery life, making them ideal for various IoT applications where long-lasting connectivity is crucial.

2.1.6 5G Security Mechanisms

Key Derivation In the 5G network, the mobile device, also known as user equipment (UE), gains data service through direct access to base stations (gNB). Figure 2.3 shows the key structures of underlying protocol stacks. The control plane provides control functions such as authentication in the access and mobility management function (AMF). The user plane provides data services with routing packets to the user plane function (UPF). Both

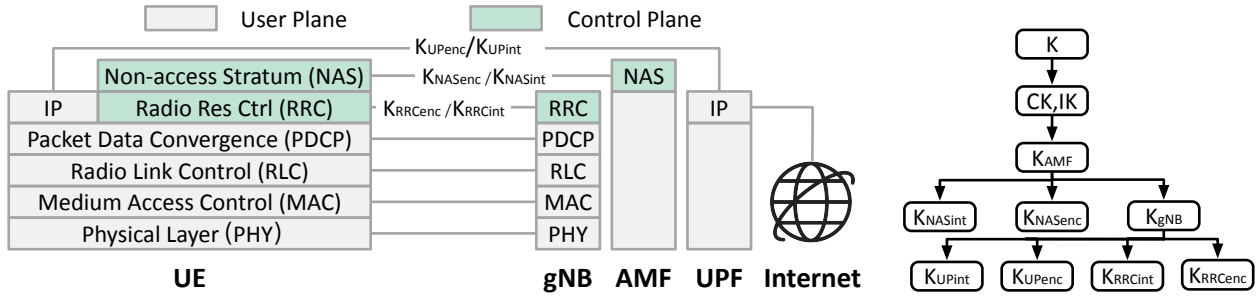


Figure 2.3: 5G key architecture.

control and user plane traffic are protected with symmetric keys. The SIM and the network share the root key K . Operators install the K when initializing the SIM, and users cannot acquire the K . All encryption and integrity keys, listed in Figure 2.3, are derived from K . Keys between the UE and AMF protect the Non-access Stratum (NAS) signaling messages. Other keys are used between UE and gNB at Packet Data Convergence Protocol (PDCP) to provide security for Radio Resource Control (RRC) signaling messages and user data.

As a crucial step for key derivation, the Authentication and Key Agreement (AKA) procedure provides mutual authentication as shown in Figure 2.4(a). First, the UE's modem initializes a NAS Attach Request message, including IMSI as the identity. Upon receiving the request, AMF sends a NAS Auth Request to the UE, including a random byte array (RAND) and authentication token (AUTN) generated based on the K and RAND. The UE's modem forwards the RAND and AUTH to the SIM. The SIM verifies the AMF based on the K and derives Cipher Key (CK), Integrity Key (IK), and the response value (RES). The modem gets keys and RES through APDU. The modem derives the K_{AMF} with CK/IK and sends the RES to the AMF. If it is as expected, the AMF authenticates the UE and agrees on the shared K_{AMF} . Both sides derive NAS encryption and integrity key K_{NASenc}/K_{NASint} from the K_{AMF} with the algorithm identifier in the NAS Security Mode Command (SMC). Finally, both sides derive keys for protecting RRC signaling and user data.

Fast Reattach Procedure To speed up the reattach procedure, the standard allows

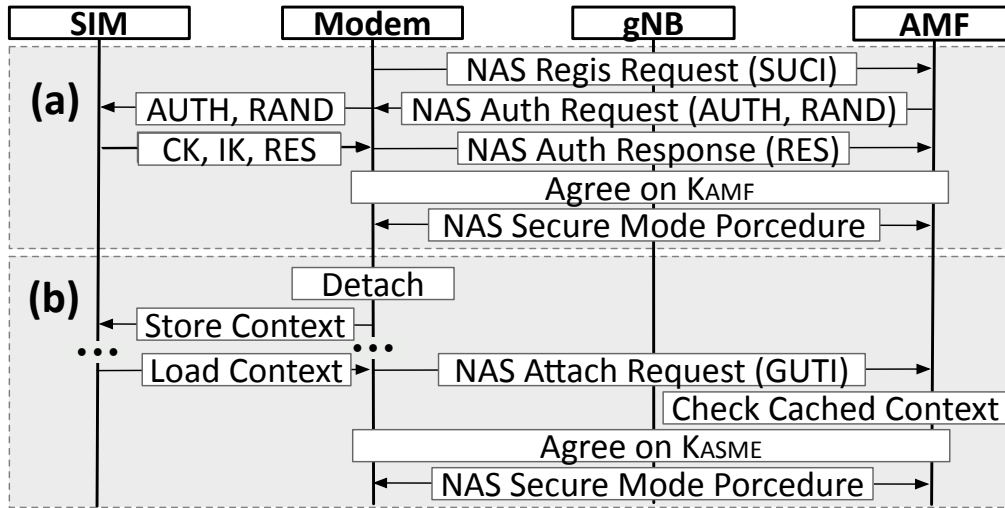


Figure 2.4: Authentication procedures.

the UE to reattach with the cached context² [3GP20b]. When the UE detaches (e.g., loses signal, enables airplane mode, or turns off), the UE shall store the context into the SIM if the SIM supports the fast reattach service. The context includes the identifier called Globally Unique Temporary Identifier (GUTI) in file EPSLOCI, and NAS security context in file EPSNSC. When the UE detects a cached context, it will use the cached GUTI as the identity in the NAS Attach Request, as shown in Figure 2.4(b). The NAS Attach Request will be integrity-protected with the UE’s cached context. If the AMF finds a record corresponding to the context, the operators can bypass the AKA procedure [3GP20b]. In this case, the AMF directly sends the NAS SMC to the UE. They authenticate mutually by checking the message’s integrity with the cached K_{NASint} . Then they derive all the following keys and set up the network connection.

²In 3GPP, it is called EPS Mobility Management Information.

2.2 Practices for High Resiliency: State of the Arts

System resiliency in the context of 5G and 5G IoT systems refers to their ability to withstand, recover, and adapt to various types of failures, disruptions, and security threats while maintaining their essential functionality, performance, and security. This encompasses the prevention, detection, mitigation, and recovery from protocol failures, connectivity outages, and security vulnerabilities.

To enhance system resiliency, solutions can be divided into two domains: device-side and infrastructure-side. Device-side solutions focus on failure handling and security schemes in modems, mobile operating systems, and applications, such as incorporating cross-layer failure detection and reaction schemes, and using secure communication protocols. Infrastructure-side solutions, on the other hand, emphasize deploying heterogeneous mechanisms within the 5G network and edge/cloud servers, including dynamic routing protocols, self-organizing networks, firewalls, intrusion detection/prevention systems, and redundant resources with load balancing techniques for high availability and fault tolerance.

In this section, we discuss current strategies for enhancing 5G network resilience, examining solutions across both device-side and infrastructure-side domains. We analyze how these approaches tackle challenges in achieving highly resilient systems, addressing protocol failures, connectivity outages, and security vulnerabilities.

2.2.1 Device: Multi-layer Mechanisms for Resiliency

The device-side resiliency schemes span across multiple layers, encompassing modems, operating systems, and applications. We present an overview of the current device-side resiliency mechanisms and their limitations.

Modem The modem’s firmware implements the 5G protocol stack for control and data-plane management [3GP21b], using a timeout-based approach to identify and address failed procedures. Standardized protocol messages and finite state machines determine whether to

abort connections or trigger retransmissions. For instance, a modem that fails to receive a Registration Request response waits for T3511 (10s by default) before retrying, ultimately waiting for the longer timer T3502 (12mins by default) after five attempts. Timeout durations vary across different procedures.

Dual-modem solutions are employed to manage connectivity outages, with each modem connected to a distinct carrier network. The backup modem takes over data transfer when the primary modem experiences an outage. However, hardware replicas result in increased power consumption, limiting their use to powerful gateways rather than smartphones or 5G IoT devices [Pep23]. Furthermore, modems rely on timeout-based outage detection, identifying outages after a 15-second timeout [Tel23, DIG23], which may cause the backup modem’s data session to be released due to inactivity [TKS16, Cis23]. This necessitates another data session setup before resuming data transfer.

Regarding security, 5G modems employ cryptographic algorithms and procedures specified by 3GPP standards [3GP20c]. Leveraging an in-SIM root key pre-shared between the SIM and the 5G core network, these mechanisms ensure confidentiality, integrity, and authentication of device-network communications. For example, the modem establishes a secure channel with the network through mutual authentication using subscriber credentials and a network-generated challenge. Additionally, modems utilize encryption algorithms for data confidentiality and integrity protection mechanisms to prevent tampering or manipulation of signaling messages.

In conclusion, the current modem-based approach has limitations in achieving high resiliency for 5G networks. The simplistic timeout-based strategy results in slow failure and outage handling, leading to recurring protocol failures and extended connectivity outages. Moreover, modem interactions with the SIM and protocol stacks expose vulnerabilities, such as the unencrypted channel between the modem and the SIM, which allows attackers to eavesdrop on and spoof SIM-modem communication.

Operating Systems Current operating systems are designed to handle failures and provide security protections to ensure resiliency. For failure and outage handling, they employ detection and reaction mechanisms to enhance resiliency. For example, Android adopts a “probe and retry” approach for data delivery failure detection and recovery [And22e], monitoring network statistics and periodically sending DNS and HTTPS requests. To address these failures, Android utilizes a “sequential retry” strategy, sequentially cleaning up and restarting TCP connections, re-registering the network, and restarting the modem. If unsuccessful, Android retries after a three-minute interval.

To maintain cellular security in 5G networks, mobile OSs deploy multiple defense mechanisms. These include a secure boot mechanism to authenticate firmware and software components, sandboxing techniques to isolate applications and restrict access to sensitive resources, end-to-end encryption to safeguard data integrity and confidentiality during transmission, and anomaly detection algorithms to identify and mitigate malicious network activity.

However, these mechanisms exhibit limitations in addressing challenges specific to 5G systems. Although operators recommend retry strategies [T M22, AT22], Android’s sequential retry actions can prolong recovery attempts. The absence of fine-grained failure diagnosis exacerbates disruptions and may impede recovery. For example, refreshing TCP connections may be futile when 5G failures underpin TCP issues. Furthermore, when devices transition between carriers’ coverage areas, current operating systems struggle to manage connectivity outages, resulting in prolonged service disruptions. Finally, modem vulnerabilities can compromise the 5G security setup as the OS fully trusts the modem.

Resiliency Design in Applications In addition to modem-based and OS-based solutions, there are application-based proposals for addressing network connectivity issues. MobileInsight [LPY16] offers in-device failure detection through continuous monitoring of diag port messages, while commercial tools such as NetMotion [Net22a] use mobile applications to report high-level metrics for failure analysis. However, these application-based solutions have limited recovery capabilities, particularly for cellular stack failures or those resulting

from outdated configurations. In cases requiring user intervention, such as expired data plans or authentication failures, devices lack sufficient information to prompt appropriate actions.

Multi-carrier technologies have been utilized by applications to mitigate connectivity outages, which store profiles from multiple carriers on a single SIM card. This allows devices to switch carriers when connectivity is lost. Solutions like Google Fi coordinate carrier switches through applications, and they rely on infrastructure-side support, such as crowd-sourcing servers, for carrier selection. However, during failures, the data service is disrupted, and devices are unable to collaborate with the network side. Furthermore, the current multi-carrier solutions cannot scan alternative carriers without interrupting ongoing data transfers [LPD18]. This results in prolonged disruptions due to exhaustive searches without network-side knowledge.

To this end, modem, OS, and application-based solutions face significant challenges due to their reliance on limited device-side information. The absence of network-side information hampers the ability to perform fine-grained failure diagnosis and effectively manage connectivity outages, ultimately leading to prolonged data service disruptions. As we transition to 5G networks, the existing security measures prove insufficient in addressing the novel attack vectors introduced by technological advancements. Consequently, this leaves the 5G service vulnerable to a wide array of cyber threats, jeopardizing security and impeding resiliency.

2.2.2 Infrastructure: Enhancing Control and Management for Resiliency

To support advanced functionalities and stringent performance requirements of 5G and 5G IoT applications, a resilient cloud and network infrastructure is indispensable. This section examines the resiliency designs in cloud infrastructure and 5G networks, emphasizing the hardware and software techniques employed to achieve high availability and reliability. However, the limitations in infrastructure resiliency still pose challenges to the overall resiliency

of 5G services.

Resiliency Designs in Cloud Cloud infrastructure serves as the backbone for numerous 5G application services, necessitating both hardware and software aspects of system resiliency to be addressed. Hardware redundancy can be achieved through solutions such as backup power supplies [AAB21], RAID storage [KMA22], and hot-swappable components [SHC18]. Conversely, software techniques include failover mechanisms [KJR21], data replication [KMA22], load balancing [BTY20], and auto-scaling [KLH18], among others. By leveraging these hardware and software techniques, back-end cloud systems attain high resiliency, ensuring the seamless operation of 5G and 5G IoT applications while meeting the increasing demands of users and IoT devices.

Resiliency Designs in 5G Networks To achieve high resiliency in 5G services, the current 5G infrastructure employs various schemes. These schemes focus on enhancing network component robustness and redundancy, implementing self-healing mechanisms, and utilizing software-defined networking (SDN) and network function virtualization (NFV) technologies [BAM20, GZL19] for rapid adaptation and recovery from failures. Moreover, advanced algorithms for real-time monitoring, anomaly detection, and machine learning techniques are incorporated to predict and preemptively address potential network disruptions [AHB21, MAQ21, ZZY19].

Despite existing measures, attaining high resiliency in 5G services is constrained by infrastructure limitations. These include insufficient access to higher-layer information (e.g., transport and app layers), complicating the accurate diagnosis of high-layer failures. The lack of direct device control further restricts responsiveness to connectivity outages. Moreover, the infrastructure's reliance on the in-SIM key for mutual authentication leaves device-side vulnerabilities unaddressed through infrastructure upgrades. Consequently, the limited capability to control device behavior results in marginal improvements to 5G service resiliency. Thus, infrastructure-side solutions face challenges in effectively managing failures, outages, or attacks to ensure high resiliency for 5G and 5G IoT devices.

2.3 Insufficient Resiliency in Current 5G Architectures

In summary, the current schemes deployed across multiple entities in the 5G ecosystem are insufficient to ensure highly resilient 5G services. The device-side solutions, including modem, operating system, and application-based resiliency designs, face considerable challenges due to their reliance on limited device-side information. These approaches exhibit shortcomings in performing fine-grained failure diagnosis, effectively managing connectivity outages, and addressing novel attack vectors. For instance, timeout-based strategies result in slow failure handling, existing multi-carrier and dual-modem are insufficient for connectivity outages, and the vulnerabilities at the SIM jeopardize the 5G security. As a result, these limitations lead to prolonged data service disruptions and leave 5G services vulnerable to a wide array of cyber threats, resulting in low resiliency.

Furthermore, limited capabilities to control device behavior and the absence of device-side knowledge impede infrastructure-based improvements to resiliency. In addition to the limitations of individual entities, our study reveals that interactions between devices, infrastructure, and protocol stacks introduce new security vulnerabilities, further undermining the 5G system’s resiliency. These issues stem from the “dumb terminal, smart core” nature of the current 5G architectural design. A lack of effective device-side disruption handling schemes compromises resiliency.

CHAPTER 3

Overview

This dissertation investigates the potential of SIM-centric designs to enhance the resiliency of 5G and 5G IoT systems. We first summarize the challenges in achieving high resiliency, which includes protocol failures, connectivity outages, and security vulnerabilities (§3.1). We then explore the potential of SIM/eSIM capabilities (§3.2). We present innovative SIM-based designs that address the resiliency issues (§3.3), and conclude this chapter with a roadmap of how to build resilient 5G systems (§3.4).

3.1 Challenges for High Resiliency

Current designs face difficulties in achieving high resiliency due to various challenges. Here, we briefly present the issues and root causes.

Challenges in fine-grained failure diagnosis The first challenge comes from the failure diagnosis [Com21]. Despite 5G advancements, users still experience data service-related failures with strong radio signals on their smartphones. Our evaluation of existing device failure handling mechanisms reveals their inability to provide fine-grained diagnoses, leading to insufficient management of diverse failures. Limited device-side information is the root cause, preventing proper identification and handling of failures originating from the cellular network. The lacking fine-grained failure diagnosis results in prolonged application disruptions or completely fail to recover.

Limitations in connectivity outage handling The second challenge is connectivity

outage handling. Single-carrier solutions are constrained by coverage limitations [FCC21]. However, existing multi-carrier switching schemes, such as Google Fi [Goo23] and OneSIMCard [One23], also suffer from the current disconnect-scan-switch approach in managing connectivity outages. This method disrupts ongoing data transfers, leading to prolonged disruptions. Moreover, power-saving schemes on 5G and 5G IoT devices further complicate matters, as devices in sleep mode cannot detect mobility or session updates, causing inconsistent states between the device and the 5G infrastructure upon waking.

Issues in current security mechanisms Our study challenges the perception that 5G technology ensures security with existing in-SIM keys. We uncover new vulnerabilities, leading to traffic eavesdropping, man-in-the-middle attacks, and impersonation. The root cause lies in the limitations of the PIN-based security protection implemented on SIM and eSIM, which fails to provide fine-grained access control. Managing access control policies for approximately 200 files on modern SIM cards presents significant challenges. Moreover, fine-grained access control alone is insufficient; robust authentication mechanisms are necessary to protect against threats. However, the current PIN-based schemes cannot distinguish diverse in-SIM applets and off-SIM units. New security designs are urgently needed to ensure the 5G service resiliency.

3.2 Design Space of SIM/eSIM

The SIM card or eSIM chip naturally exists on 5G IoT devices. We investigate the current capabilities of SIM/eSIM. The results show that we could harness the readily available features of SIM/eSIM to enable highly resilient 5G system designs.

Independent Miniature System The SIM/eSIM functions as an independent miniature system, featuring a processor, storage, and security fences. This architecture allows the SIM/eSIM to securely store user profiles, process data, and authenticate device access. Commercial SIMs typically have 320KB of storage and a 10MHz processor [Emn23], which

are sufficient to run lightweight applications on the SIM/eSIM. Industrial IoT SIMs exhibit exceptional reliability with lifespans exceeding ten years and the ability to withstand extreme conditions, such as vibrations, chemical exposure, and corrosion [Emn23].

Effective Device Control The SIM/eSIM provides mature interfaces that facilitate device control, enabling efficient network management. Utilizing proactive commands [ETS19b], the SIM/eSIM is capable of controlling modem behaviors. This functionality allows the SIM to actively monitor signal conditions, track modem status, execute multi-tier reset/redo actions, and switch between different carriers when necessary. As a result, the SIM can perform simple yet effective actions for handling failures and outages, ensuring its adaptability to changing network conditions and maintaining high resiliency.

Bridging Device and Infrastructure The SIM over-the-air (OTA) channel allows the SIM to send and receive data packets, providing a secure and dependable connection for the device. Our design enhances this capability. The SIM can actively track available data channels to proactively detect disruptions. We further leverage it for the rapid deployment of security features. Moreover, we venture beyond traditional OTA channels by empowering the SIM to interact with 5G infrastructure through signaling messages. This allows for fine-grained diagnostics and secure updates, even in the absence of data services.

Current practices primarily utilize the SIM for device identification purposes [SBA21]. Our proposal exploits the full potential of the SIM/eSIM to provide failure diagnosis, connectivity outage handling, and security protection. We demonstrate that the SIM-centric software design enables plug-and-play functionality. By harnessing the SIM’s capabilities, we can implement simple yet effective actions at the device, and enable new device intelligence for highly resilient 5G and 5G IoT systems.

3.3 Overview of SIM-centric Designs

To address the resiliency issues identified in 5G and 5G IoT systems, we propose innovative, plug-and-play SIM-based solutions.

SIM-based 5G failure diagnosis & handling We aim to facilitate fine-grained failure diagnosis and handling using SIM. The key idea is to utilize error codes present in standardized 5G signaling messages to infer the root cause of failures. By incorporating a domain-specific machine-learning algorithm, we enhance the diagnostic process. Once the failure cause is identified, the SIM is enabled to execute multi-tier reset/redo actions, such as resetting protocol operations, refreshing outdated configurations, and reloading profiles. Consequently, this approach allows for swift and effective handling of protocol failures with the SIM/eSIM. Furthermore, we innovatively leverage the standardized 5G signaling to transfer the failure causes between the SIM and 5G infrastructure. This design enables fine-grained diagnosis under failures, even if the data plane is broken or has not been established.

SIM-based high availability design for 5G connectivity outages We address connectivity outages caused by wireless outages, mobility, and power-saving schemes, which negatively impact 5G service availability. Our novel SIM/eSIM-based solution mitigates these availability gaps by employing a device-based, SIM-centric software approach and plug-and-play software-defined receiver hardware. This enables rapid switching between multiple carrier networks. Our design adopts a data-first approach, utilizing concurrent data/control transfer and fast reconnect to minimize data access downtime induced by power saving. Evaluations using commodity IoT devices and commercial 5G networks confirm the solution’s viability. It readily achieves two to three nines of availability (with the potential to reach four to five nines), while preserving energy efficiency in the evaluated application scenarios.

Secure Authentication and Access Control with SIM/eSIM The current 5G network is vulnerable to traffic eavesdropping, man-in-the-middle, and impersonation attacks,

primarily due to insufficient authentication and inadequate access control for the numerous in-SIM files associated with in-card applets and off-card units. To tackle this issue, we develop a novel access graph to systematically organize SIM files and analyze common scenarios requiring SIM access, encompassing all off-card units and in-card applets. Our multi-rooted tree structure within the access graph provides fine-grained access control to various entities, detects conflicts, and generates the access graph effectively. To ensure authentication, flexibility, and scalability, we devise a certificate-based solution, constructing a certificate chain to differentiate between multiple entities, while certificate ID binding and selective encryption offer protection against a range of attacks. Our solution effectively safeguards 5G devices while incurring only marginal overhead.

3.4 Roadmap

The following chapters detail the components of the SIM-based solution and demonstrate how they enhance resiliency in 5G and 5G IoT systems. Chapter 4 presents the fine-grained 5G failure diagnosis and handling enabled by the SIM. Chapter 5 introduces a SIM-centric, highly available 5G design that rapidly addresses connectivity outages. Chapter 6 showcases a certificate-based SIM design that provides fine-grained access control and effectively mitigates heterogeneous attacks. Lastly, Chapter 7 introduces our open-sourced platform for demonstrating the SIM-based highly resilient 5G system designs.

CHAPTER 4

SEED: SIM-Based 5G Failure Diagnosis and Handling

This chapter introduces our SIM-based solution, SEED, aiming to provide fine-grained diagnosis and handling to 5G protocol failures. We overview the failures in cellular networks in §4.1 and introduce the limitations of current solutions in §4.2. §4.3-§4.4 elaborates on the details of SEED designs. We introduce the implementation in §4.5 and evaluate it in §4.6. §4.7 further discuss how to gradually deploy SEED in current 5G networks. We present the related work in §4.8 and conclude this chapter in §4.9.

4.1 Cellular Network Failures

Failures have become the norm rather than an exception in real-world 5G usage. Recent studies show that, 30% of 5G devices experience failures during an 8-month measurement [LLL21]. Various failures are also reported by users each day [Dow22b, Dow22a]. From the technology perspective, 5G is using small cells in the mmWave bands with smaller single-cell coverage, thus triggering more frequent handovers [XZZ20]. Therefore, 5G may incur more failure events with the increased frequency to sync up the control-plane state, security update, and data sessions.

Failures in 5G are consequently becoming diversified. In general, three types of failures arise at different stages of data transfer. Control-plane management failures arise during the control signaling operations, including setup, tracking area update, or teardown procedures. Data-plane management failures affect the data session setup, modification, or release proce-

dures. Data delivery failures incur packet delivery stall with established data-plane sessions due to various causes (e.g., DNS errors, port blocking, etc).

These failures happen at both device and network sides. The device may use outdated configurations, resulting in connectivity failure [the20]. The network could suffer from congestion, thus unable to respond to requests in time [ZZL14]. Every component (hardware, cellular stack, OS, etc.) could be the source of failure [LLL21], given the diversity and complexity of devices and infrastructure in both hardware and software.

4.2 Limitations of Current Solutions

Despite the current solutions at the device, 5G users still regularly perceive data service related failures, even with a strong radio signal bar at the smartphone [Com21]. We thus seek to first understand how failures exhibit in practice given the deployed failure mechanisms through a trace analysis. We then assess the existing modem and Android failure handling. Our results show that both solutions only perform coarse-grained diagnoses and cannot handle diversified failures well. Existing modem schemes incur repeated failures and long disruptions, and Android suffers from prolonged failure detection and false positives.

4.2.1 Failures in the Real World

In this work, we focus on failures related to the 5G protocol stack. The network failures induced by other components, such as internet outages, erroneous application implementations, or OS firewall settings (e.g., user-determined network restrictions for apps), are out of scope. We first analyze control/data-plane management failures in 5G. The 5G standard provides failure causes to indicate failure reasons for control/data-plane management [3GP21b], which inherit from LTE with 5G context. We perform our trace analysis on the publicly available, 6.7TB 5G/4G datasets collected from 2015-Q3 to 2021-Q4. These public datasets include 4.7 million signaling messages collected by 30+ device models using open-source tools

of MobileInsight [LPY16] and MI-LAB [MIL22]. The traces contain 8 mobile carriers from the US and China. We found 2832 failure cases from 24k control/data-plane management procedures; this gives a nontrivial, over 10% failure ratio per control/data-plane management lifespan. Table 4.1 shows the top 5 failure causes in the control and data plane management. We elaborate on them next.

For control plane management, 3 out of 5 most frequent failures are due to infrastructure-device status synchronization. The infrastructure fails to derive the updated device identity (15.2%), releases previous data bearer context (7.5%), or sends mismatched signaling (2.8%). One common reason for the unsynchronized status is that, when the device moves to a new tracking area after handover, the infrastructure fails to sync up states with the previous tracking area. With state mismatch, the device suffers from long disruptions during reattaching with outdated identities and contexts.

For data plane management, the top 2 failures are due to configurations (requested service option not subscribed, and invalid mandatory information). Although the configurations could be proactively checked from the network side, the operational failures cannot be completely eliminated in practice. The configurations could be outdated on the device and result in data plane failure. When such failures happen, the infrastructure only provides failure causes without the correct, up-to-date configurations. The device thus fails to recover, and repeated failures arise. In addition to those failures from outdated configurations, diverse failures are exhibited, including security check (user authentication failed), insufficient resources, etc. Failures caused by expired subscriptions can only be recovered with user actions such as reactivating the data plan. Reject messages may also include unspecified causes that are seen at the infrastructure or devices.

During data delivery, the three most common failures incur data stall in 5G cellular networks: TCP, UDP, and DNS [XZZ20, PA20, LLL21]. The TCP anomaly is observed in operational 5G networks [PA20]. The UDP port blocking is also widely reported by users under 5G deployment [T M21b]. For DNS failures, public DNS services such as Google DNS

Class	Failure Causes
Control Plane (56.2%)	UE identity cannot be derived by the network (15.2%)
	No Suitable Cells In tracking area (12.6%)
	PLMN not allowed (10.3%)
	No EPS bearer context activated (7.5%)
	Message type not compatible with the protocol state (2.8%)
Data Plane (43.8%)	Requested service option not subscribed (7.9%)
	Invalid mandatory information (5.9%)
	User authentication failed (4.7%)
	Request rejected, unspecified (2.6%)
	Insufficient resources (1.9%)

Table 4.1: Top 5 failure causes in control/data plane.

typically do not apply to cellular networks. Carriers usually configure users’ DNS with their local DNS resolvers (LDNS), which is less stable due to user mobility and congestion [RB14]. Although operators’ DNS servers may work correctly during the device registration, they may experience outages thereafter. Neither Android nor iOS provides default DNS configuration for backup, which makes devices difficult to recover from carrier DNS failures [AFP17]. Users have reported DNS failure instances among operators [AT20, T M21a].

4.2.2 Limitations of Modem Scheme

The current modem-based solution handles both control- and data-plane management failures. However, it does not perform fine-grained diagnosis or take precise actions for different failures. Note that it could have obtained the standardized failure causes from the signaling messages, but did not leverage them for fine-grained diagnosis. The modem either aborts the connection and retracts to idle, or triggers retransmission upon timeout. Our analysis shows that, the modem might keep on resending the signaling message with outdated status, which causes repeated control-plane management failures until the modem reboots. When outdated configurations further trigger data plane management failures, the modem cannot update them. Repeated failures are observed thereafter.

Empirical Validation We measure the disruption time with the existing modem handling

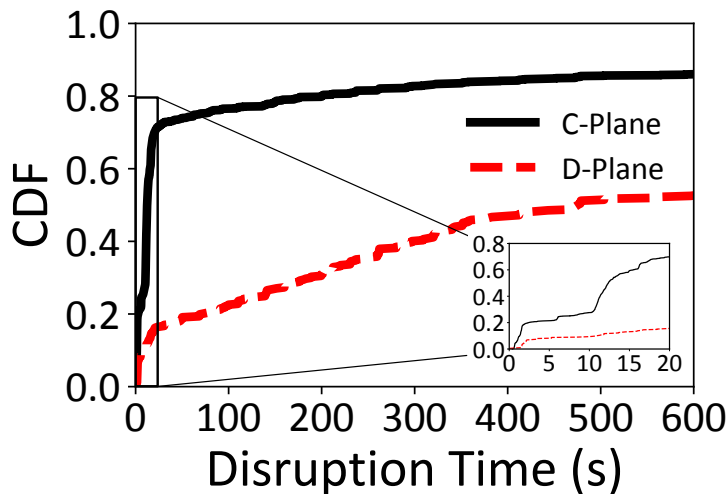


Figure 4.1: Disruption time with existing modem handling.

scheme using traces in §4.2.1. As shown in Figure 4.1, 50% of control-plane setup failures cause more than 12.4s of disruptions. Only 19% of failures could be recovered within 2s. The modem tries to reattach when various timers are triggered after 10s. However, the timeout prolongs the disruption, and only 27% of failures are recovered within 10s. Repeated failures happen when the modem retries with previous data-plane configurations. For example, when the access point name (APN) is outdated, 5G data plane setup fails. The modem activates reattachment, but still uses the previous APN during the data-plane setup, making the device fail for the same reason repeatedly. The frequent, repeated failures prolong the disruption. Only 9% of data plane management failures could be recovered within 10s. Half of the failures need about 8 minutes to be recovered.

With limited information from the network side, current modems cannot ensure fine-grained failure diagnosis and recovery. Furthermore, modem-based solutions may suffer from three problems when collaborating with the network: First, multiple parties (operators, modem vendors, etc.) need to follow the same protocol for collaborations, requiring a long time to be standardized and deployed; Second, operators may not want to leak their network-side information to third parties (e.g., modem vendors); Third, the security for modem-

network collaboration requires extra infrastructure support (e.g., public key infrastructure) and increases deployment cost.

Solutions	Failure Detection & Diagnosis?	Failure Recovery (No-user-action Required)		Failure Recovery (User-action Required)
		Config-related	Non-config-related	
Modem-based	Only device-side	Not support	Timer-based retry	Not support
OS-based	Only device-side	Not support	Layer-by-layer retry	Not support
App-based	Only device-side	Not support	Transport reconnection	Not support
Infra-based	Only infra-side	Infra-side config updates	Waiting for device retry	User Notification
SEED	Both infra & device-side	Both-side config updates	Multi-tier reset	User Notification

Table 4.2: Solution comparison for 5G failure diagnosis & handling.

4.2.3 Android: Insufficient Diag & Handling

Android further monitors failures of the data delivery stage. However, it suffers from limited detection schemes and long detection latency. First, Android does not check for those failures related to UDP, which is widely used in WebRTC, QUIC, etc., for 5G IoT and real-time applications. Second, Android provides a timer-based failure detection without distinguishing application requirements, thus resulting in prolonged disruption for all applications. For example, while video streaming apps could tolerate seconds of disruptions with a large buffer, 5G AR/VR apps fail to function properly with 100ms disruption [LVY20].

Android takes the sequential retry approach upon detecting failures via timeout. Retry is an effective solution that is also suggested by operators [T M22, AT22]. However, This level-by-level retry actions by Android yield long time intervals between two actions. While the OS-based solution could effectively check the device-side firewall, user-determined app data restrictions, etc., the limited device-side information cannot make it elude those failures from the cellular network. Without fine-grained failure diagnosis, sequential retry incurs prolonged application disruption or even no recovery. For example, if the TCP failure is caused by underlying control/data plane failures, refreshing TCP connections cannot help.

Empirical Validation We assess Android detection for TCP, UDP, and DNS failures on the latest Android 12. We connect the device with Magma cellular testbed [Mag22a].

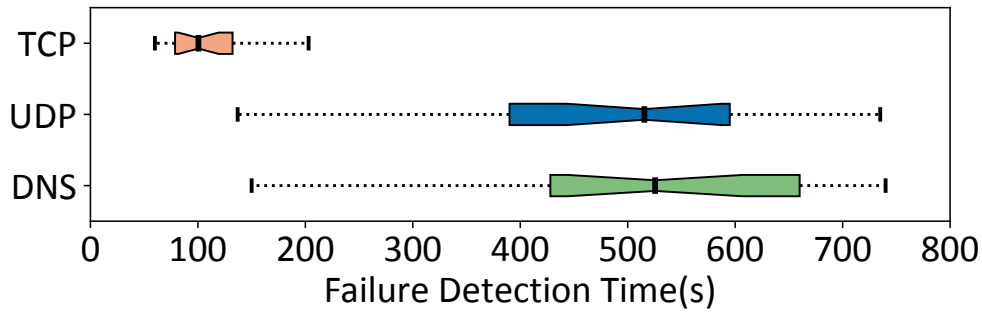


Figure 4.2: Android failure detection latency.

After the device successfully acquires the data service, we block TCP, UDP, and DNS queries, respectively, at the core network. During experiments, we play the same background Youtube video and visit websites from the browser every 5 seconds to simulate daily usage scenarios. We measure the failure detection latency from the time when failure happens to the instant Android reports data stall.

Figure 4.2 plots the latency distribution for different failure detections. For TCP failures, Android takes 1.8 minutes on average to report data stall. We note DNS and UDP failures are not well dealt with in Android. Results show that 50% of DNS failures cannot be detected within 8.7 minutes. For UDP failures, Android could only detect it if the failure leads to consecutive DNS timeouts; it takes 8 minutes on average. Otherwise, UDP failures could not be detected. Furthermore, we test Android when connections to the preset URL are blocked; this simulates failures due to server issues. Android still reports data stall alerts, which causes false positives. It further triggers recovery actions and disrupts existing connections. The interval between level-by-level reset actions is 3.5 minutes on average, which results in long disruptions.

In summary, both schemes take the device-based approach and suffer from several downsides. First, the restricted device-side information limits the fine-grained failure diagnosis and recovery schemes. Both modem and OS based solutions incur prolonged disruptions upon failures with their timeout-based detection. Second, neither exploits available error

causes carried by 5G messages. Specifically, the modem could leverage the embedded failure causes carried by the reject signaling messages for diagnosis. However, it did not. Android maps part of standardized failure causes with DataFailCause [And22f], but did not use them in failure diagnosis and recovery. Third, failure handling is simplistic. The sequential retry (i.e., level-by-level recovery) leads to long disruptions. The naïve retry by modem does not infer the causes, thus unable to fix failures due to outdated configurations. Simple retry further aggravates congestion upon failures of cell/core overload.

4.2.4 Solution Space

In addition to the modem-based and OS-based solutions, there exist some application-based proposals. MobileInsight [LPY16] provides an in-device failure detection through continuous monitoring of the diag port messages. Commercial tools such as NetMotion [Net22a] leverage a mobile application to report high-level metrics (e.g., device types, network performance, etc.) to pre-deployed servers for failure analysis. Although the application could detect failures, the recovery is limited. Applications without root can only take the transport-layer reconnection action, which cannot recover cellular stack failures. Even if the app owns root access, similar to modem/OS-based solutions, simple retries can only recover failures from temporary infrastructure-device status unsynchronization, but not failures caused by outdated configurations. Moreover, for failures that require user actions to recover (e.g., expired data plan subscription, identity authentication failures, etc.), the device cannot obtain enough hints about failures to take proper actions. Furthermore, existing SIM add-on services (e.g., [SIM19]) could only monitor the SIM hardware health (say, read/write cycles), cell signal strengths, etc., but cannot diagnose complex protocol failures. In conclusion, the device-side solutions cannot acquire the network-side information for fine-grained failure diagnosis and handling.

While the device-side approach has limitations, the second solution option is an infrastructure-based scheme. This choice also has several limitations in both failure detection and reaction.

First, the infrastructure may not have access to higher-layer information (e.g., transport and app layers), thus unable to infer high-layer failures accurately. Second, monitoring data traffic over high-speed 5G may incur significant processing overhead. Third, the infrastructure cannot differentiate which case happens in the absence of data traffic: whether misconfiguration blocks the device’s traffic, or the device is idle without data to transfer. While the infra-based solution could send failure notifications through other channels (e.g., email), it lacks device control for failure reactions. Although the infrastructure could acquire standardized causes for control/data-plane management failures, or even pinpoint the root cause (outdated configurations, customized policies, etc.), it can only update infra-side configurations for failure recovery but cannot notify devices at runtime with corresponding action commands (e.g., update SIM configurations). We summarize the existing solutions to 5G failure diagnosis and handling in Table 4.2.

The third option is to let the device and the infrastructure collaborate in failure diagnosis and reactions. The device can detect high-layer failures reliably and take direct low-level reset actions via modem. The infrastructure can directly correct misconfigurations and use crowdsourcing among devices to infer failure causes. However, a naive approach in this option also suffers from three limitations. First, the device and infrastructure might not communicate failure-related messages upon failures. Second, exposing failures may compromise system security. Third, the solution may not work within the 5G framework.

4.3 SEED: SIM-Based Failure Diagnosis

4.3.1 Case for SIM-based Solution

We make a case for a SIM-based solution that addresses all the above limitations. First, the SIM and the network can communicate via signaling messages over signaling channels, rather than data packets. The channel subsists even when the data session is not established or broken. Second, SIM is produced by operators and trusted by the in-network devices. In-

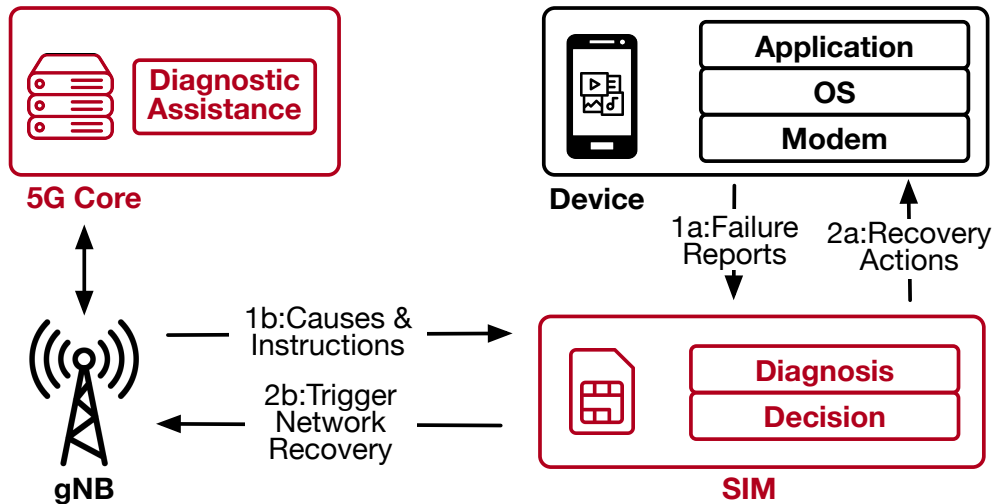


Figure 4.3: SEED diagram.

SIM keys could further protect the SIM-network communication without extra certificates. Therefore, SIM bridges the network and devices as a trusted agent. Adversaries cannot access the SIM info without the in-SIM keys. Third, SIM is applicable to nearly all cellular-connected devices. New functions could be deployed in the form of a SIM applet. They can be upgraded with the over-the-air (OTA) mechanisms on existing SIM/eSIM. Last, the SIM-based scheme offers a purely software-based solution without changing 5G standards, device firmware, or infrastructure hardware.

4.3.2 SEED Overview

We thus design SEED, a SIM-based solution to 5G failure diagnosis and handling. SEED offers a software-based scheme deployable by 5G mobile carriers. It offers lightweight, fine-grained failure detection and recovery at runtime with SIM’s constrained hardware capability. SEED leverages 5G standardized messages to infer failure causes inside the SIM. It further uses multi-tiered resets/redos for differentiated failure handling. In contrast to complex failure recovery (e.g., rollover, logging and checkpointing, recovery from crashes), SEED uses simple, yet effective resets to handle all three types of control-plane, data-plane, and

data delivery failures.

Figure 4.3 shows the overall system diagram of SEED. First, SIM receives failure reports from applications (1a) and the network (1b). Failure reports include failure clues such as network-side diagnosis, instructions with updated configurations, and device-side failure details (no connection, DNS/UDP failure, etc.). With such clues, SIM performs local diagnosis, makes handling decisions, and triggers recovery actions at the device (2a) or the network (2b). SEED addresses three issues in its SIM-based design:

- **How does SIM pinpoint failures at low overhead?** We ensure the solution is viable on resource-constrained SIM hardware. To this end, SEED combines standardized failure causes with the up-to-date configurations from the infrastructure, as well as OS/App failure reports from the device. SEED further performs fine-grained failure diagnosis with limited SIM processing and storage.
- **How does SIM handle diverse failures that arise at different stages?** We develop simple and fast failure recovery via multi-tier reset. SIM could perform profile reloads, configuration updates and failure notifications on commercial off-the-shelf devices without root access. It further supports faster control/data plane resets with root privilege.
- **How does SIM collaborate with the infrastructure when the data plane is broken?** The SIM obtains information from the infrastructure for fine-grained diagnosis and handling. We leverage existing signaling messages to transmit diagnosis information, thus ensuring runtime SIM-network information exchange upon failures of control/data-plane management or data delivery.

4.3.3 Lightweight SIM Diagnosis

We now introduce how the SIM performs fine-grained failure diagnosis with both-side information. For control-plane and data-plane management failures, the SIM receives the standardized 5G failure causes from the infrastructure, and leverages them for diagnosis. The

SIM further enables apps to report data delivery failures for fast detection and diagnosis.

4.3.3.1 Failures in Control/Data-plane Management

The 5G standardized failure causes provide a good source for SIM diagnosis. 5G defines 80+ failure codes, which are embedded in signaling messages. Most of the messages containing failure codes are the “reject” messages from the network or the device, such as Authentication Reject, PDU Session Modification Reject, etc. The messages that embed standardized causes have been widely deployed in practice [Cis22].

SEED achieves lightweight and fine-grained SIM diagnosis with such standardized causes. When the infrastructure composes the reject or receives device reject messages, it extracts the embedded standardized cause and sends the cause code to the SIM (more details in §4.3.5). The SIM applet stores all standardized cause codes and looks up the received cause to quickly detect and pinpoint the failure. Although the SIM’s storage is limited (32~128KB), it is sufficient to hold all cause codes for in-SIM analysis. The causes could be categorized into control-plane management and data-plane management. Control-plane management causes include failures related to UE identification, subscription options, network congestion, authentication, invalid messages, etc. Data-plane management causes include configuration failures and protocol errors. Standardized causes are already supported at the infrastructure and do not need extra modules or algorithms, thus resulting in marginal overhead.

SEED further exploits SIM capability to prevent repeated failures. If the failure cause is related to outdated configurations, simple retries cannot succeed but result in repeated failures. Therefore, when the infrastructure initializes the reject due to outdated device configurations, it sends the up-to-date configuration together with the cause code to the SIM. We list the failure causes related to outdated configurations in Appendix A.1. Upon receiving these cause codes, the SIM parses the configurations based on the cause code and stores them for next-step handling (§4.3.4).

4.3.3.2 Failures in Data Delivery

For packet delivery failures, current user applications cannot diagnose them with limited low-layer information from the mobile OS. Emerging 5G applications are disruption-sensitive and require quick recovery. SEED thus enables these applications to report failures for fast diagnosis. Applications could call the failure report API if they need fast failure handling. The API carries three parameters (failure type, traffic direction, address). The failure types support the three most common failures discussed in §4.2.1: DNS, TCP, and UDP. The application indicates the failed traffic direction, including uplink, downlink, or both. The address contains the IP and port for TCP/UDP failures. These fields are used by the 5G Traffic Flow Template (TFT) to regulate the traffic and activate IP/port blocking with incorrect configurations. The domain names are embedded in the address field for DNS failures. The report enables disruption-sensitive apps to bypass long Android detection and speed up the diagnosis. SIM further leverages existing APIs to acquire the Android data stall notification. Whenever receiving the App/OS failure reports, SIM leverages the reported information for fast failure handling in §4.3.4. Note that SEED does not explicitly diagnose and handle instantaneous, underlying radio link failures. However, such physical-layer issues may affect control/data-plane management and data delivery, which will be observed at higher layers and handled by SEED.

4.3.4 SIM-Based Failure Handling

With the diagnosis result, we address the next issue of reacting to diverse failures that arise at different stages. Different from the blind retry by the modem and Android, SEED handles diverse failures via the multi-tier reset mechanism directly, facilitated by both-side information to pinpoint the failure, thus leading to fast failure recovery. We first list the multi-tier reset actions taken by the SIM. We then elaborate on how the SIM decides which action to take accordingly.

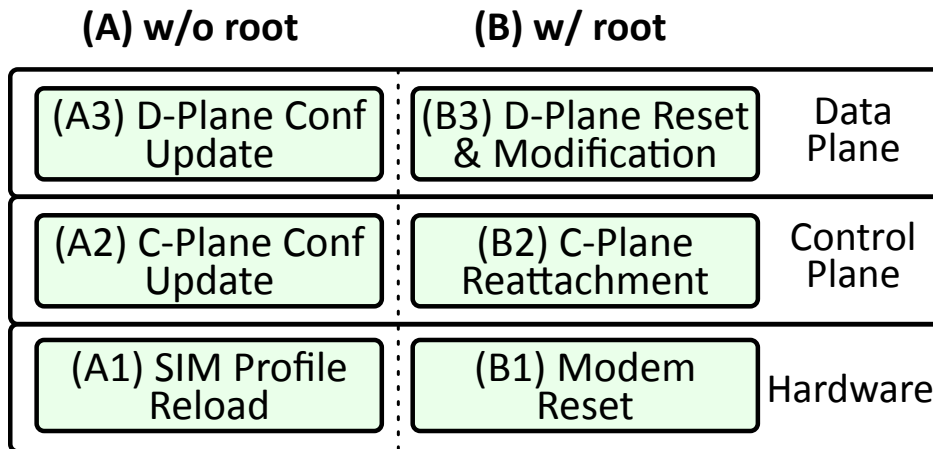


Figure 4.4: Multi-tier reset w/ and w/o root privilege.

4.3.4.1 Handling with Multi-tier Reset

SEED leverages the SIM to initiate multi-tier reset actions. This is nontrivial, since the SIM does not directly control those cellular connections. SEED explores the limited interfaces supported by current 5G devices, and designates two modes with different device privileges. Without root privilege, SEED-U uses the multi-tier reset to reload the failed module directly. When root privilege is available, SEED-R further improves the recovery speed.

Multi-tier Reset without Root Access SEED-U takes multi-tier reset actions for failure handling at three levels as shown in Figure 4.4(A). At the hardware level, reset enforces the modem to clear its cached contexts, preventing it from being stuck in prolonged attempts with invalid caches. The SIM triggers profile reloading at the modem to sync its SIM profiles (A1). Different from the naive retry scheme in the current modem/OS, the SIM also retrieves the latest configurations from the infrastructure to handle outdated configurations. The SIM updates the control-plane configurations (A2) (e.g., PLMN list) to reduce excessive search time. The mismatched control-plane states/identities (shown in Table 4.1) are also refreshed in the reset. SEED-U leverages the proactive commands between the SIM and the modem to realize these two actions [ETS19b]. The proactive command is usually used to provide carrier services such as OTA updates, which has been supported

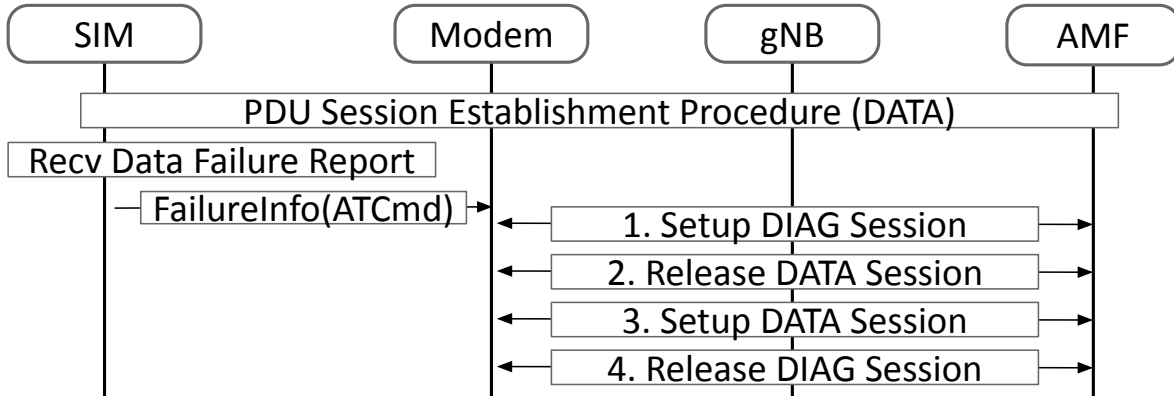


Figure 4.5: Reset data plane without reattachment.

by current smartphones without root privilege. To our knowledge, SEED-U is the first to leverage it for failure handling. SEED-U could further update the acquired data-plane configurations from the SIM (A3), such as DNNs or APNs, leveraging the Android carrier app [And22g]. All such actions do not require root privilege at smartphones.

Boost Multi-tier Reset with Root Privilege With root privilege, SEED-R further improves granularity and speed for diagnosis. 5G/4G devices provide AT commands [3GP21c] as another interface for fine-grained modem control but require root privilege. When the carrier app detects that root access is permitted, it will notify the SIM through APDU to enable the SEED-R mode. Figure 4.4(B) shows the multi-tier reset with root privilege. Upon hardware failures, SIM restarts the modem (B1). It recovers the modem from being stuck in internal errors. For control-plane failures, SEED-R directly controls the modem for control-plane reattach (B2), which improves the recovery speed without prolonged search procedures.

SIM further collaborates with the infrastructure for data-plane resets (B3). Resetting only the data session speeds up the handling. However, 5G gNB releases the last radio bearer once the last data session is released, thus causing the control-plane reattach. SEED designs the fast data plane reset in Figure 4.5 without resetting the control plane. Upon receiving failure reports about the initial data session, the SIM triggers the modem to set up

another data session with DNN “DIAG”. The reattach will not happen when “DATA” session is released as “DIAG” session and corresponding 5G gNB radio bearer still exist. Finally, the device reconnects “DATA” session and releases the “DIAG” session. The network could also modify the existing “DATA” bearer rather than reset it, if only configurations (e.g., TFT) need to be updated. All the session setup/release/modification signalings are standardized in 5G [3GP21b]. SEED leverages them to handle data delivery failures without disrupting the existing, established data plane.

4.3.4.2 Deciding on Reset Actions

Diagnosis Class	Failure Handling w/o Root	Failure Handling w/ Root
Control-plane Causes	SIM Profile Reload (A1)	Reset Modem (B1)
Control-plane Causes w/ Config	Parameter Update & Reload (A2 & A1)	Control-Plane Reattach w/ Update (B2)
Data-plane Causes	SIM Profile Reload (A1)	Data-plane Reset (B3)
Data-plane Causes w/ Config	Configuration Update (A3)	Data-plane Modification (B3)
Data Delivery Failures from App/OS	Configuration Update (A3)	Data-plane Reset / Modification (B3)

Table 4.3: Failure dandling decisions with diagnosis results.

Resetting different modules require different latencies. To speed up failure handling, the SIM uses the diagnosis results and the current mode to perform targeted reset action without layer-by-layer retry.

Table 4.3 shows the SIM handling decisions without root (SEED-U) and with root privilege (SEED-R). For control-plane management failures not caused by outdated configurations, SEED-U reloads the SIM profile to reattach (A1). With root privilege, SEED-R performs modem reset using AT commands (B2). When outdated configurations incur failures, SEED-U updates the control-plane parameters and reloads SIM profiles for registration (A2 & A1). With root privilege, SEED-R updates the configurations and triggers reattach on modem for fast handling (B2). As 20% of control-plane management failures could be recovered within 2s (§4.2.2), SEED sets a 2s timer before triggering hardware and control plane reset. The short timeout enables speedy recovery upon such failures.

When data-plane management failures arise, with SEED-U, the SIM triggers the SIM

profile reloading. The data plane will be reset after the control-plane reattach. With root access, SEED-R triggers data-plane reset for faster failure handling (B3). When the SIM acquires data-plane configurations (DNN, PDN type, etc.) from the infrastructure, the SIM further triggers configuration updates (A3) with SEED-U or data-plane modification (B3) with SEED-R.

SIM may receive applications/OS reports for data delivery failures. If it received causes on control/data-plane management failures within the last 5s, there could be an ongoing handling. The SIM does not trigger handling to avoid conflict. Otherwise, the SIM triggers the configuration updates in the carrier app (A3) to reset data connection without root. SEED's rate-limit design does not perform the same reset action consecutively and frequently; the signaling messages are thus not overwhelming. With root access, the SIM sends the failure report collected from App/OS (§4.3.3.2) to the infrastructure with real-time SIM-Infra collaboration (details in §4.3.5). The infrastructure checks if the failure type, direction, and address conflict with user policies, or if DNS failure happens. It then modifies the data session with updated user policies when conflict arises for TCP/UDP, or configures a new DNS server in the followup reset (B3).

4.3.5 Real-Time SIM-Network Collaboration

We next address the issue of enabling SIM-infra interaction when the data plane is broken, without changing modem/gNB firmware or modifying standardized messages. Note that SIM needs to acquire the information of failure causes and updated configurations from the infrastructure. SIM also notifies the infrastructure for data-plane resets. Although SIM OTA provides a channel for SIM-Infrastructure communication [SIM19, SIM22], it relies on data service (e.g., TCP/UDP) and cannot work during connection setup. Moreover, upon data delivery failures, packets may not be delivered and SIM OTA is unavailable.

SEED leverages standard-compliant control-plane signaling messages. The infrastructure embeds the failure-related info in Authentication Request signaling messages. SIM embeds

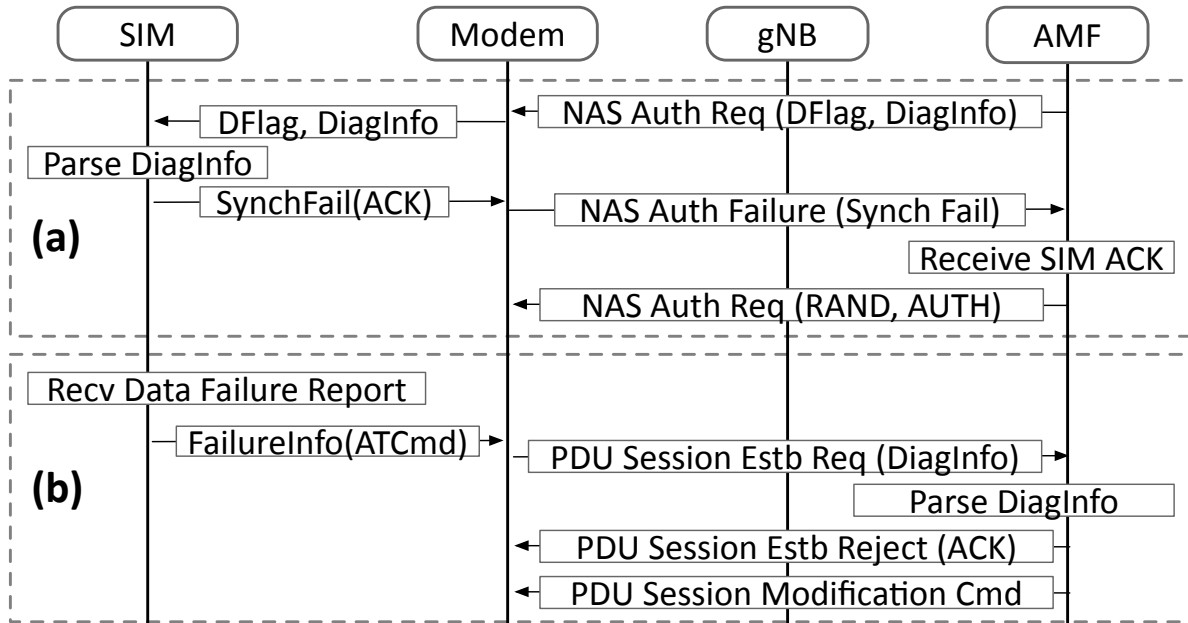


Figure 4.6: Collaboration with standard-complied signaling messages (a) Network to SIM (b) SIM to Network.

failure diagnosis results in PDU Session Establishment Request to trigger data-plane resets. SEED design is compatible with 5G commodity devices without modem or gNB firmware modifications. These messages are available in the presence of failures. Hence, the infrastructure and the SIM could perform real-time interactions for failure diagnosis and handling, even when the data bearer is not up or broken.

How Does Infrastructure Deliver Info to SIM? In SEED, the network leverages the 5G Authentication Request message used for mutual authentication to send diagnosis info. In this message, the network generates a 16-byte RAND and a 16-byte AUTH, which the modem will forward to the SIM for authentication [ISO13]. SEED reserves a RAND value (FF...FF) as the Diagnosis Flag (DFlag). As shown in Figure 4.6(a), the 5G network embeds the diagnosis info in the AUTH field and sets the RAND as DFlag. When the SIM sees the reserved DFlag, it does not verify the key but parses the AUTH, which is encrypted and integrity protected with a counter using the pre-shared in-SIM key. SIM returns synchronization failure as the ACK upon successfully receiving the diagnosis. If the

sync failure is not responded, the modem may label the network as untrusted. The network then resends a normal Authentication Request. The 16B AUTH suffices to hold the cause code and most updated configurations. The network could embed more information with multiple transmission rounds. Note that, the network can send Auth Request at any time with a NAS signaling connection [3GP21b]. Although the control plane is not fully established (with successful completion of both authentication and configurations), the network could still transmit Auth Request to the SIM, thus enabling collaboration in the presence of control/data-plane failures.

How Does SIM Transfer Info to Infrastructure? Data delivery failures may block packet transfer with data plane set up. SEED embeds the failure report collected by SIM in the PDU Session Establishment Request to report data delivery failures, as shown in Figure 4.6(b). After control-plane setup, the device requests the data bearer with a corresponding Data Network Name (DNN) in 5G [3GP21b]. Standards support sending DNNs for multiple data sessions, which enables SEED to report failures anytime after control plane setup with a new request. SEED leverages the undefined field to embed the diagnosis information in the DNN field [3GP21a], which is also encrypted and integrity protected using the in-SIM key. The 100B DNN size is sufficient for the current report; our experiments further validate that, a longer report triggering multiple consecutive requests can be fragmented into several DNNs. SEED triggers the modem to send the request with a special diagnosis DNN starting with “DIAG”. After the network gets the DNN and parses the report, it validates the failure with current user settings, responds with a reject as ACK, and modifies the current data session configs or follows Figure 4.5 for the reset.

The real-time SIM-Infrastructure collaboration is compatible with 5G devices. The gNB/modem firmware remains unchanged. Carriers could update the SIM via OTA for new applet logic. The network-side functions could be extended with a new core module handling diagnosis messages, which current cloud-based core network implementations could quickly deploy [Azu22, Mag22a]. SEED only requires a small amount of extra signaling mes-

sages for collaboration with marginal overhead at the device and the network. Note that, the real-time collaboration cannot work if the radio access is broken. The radio link issues and recoveries are well studied [PKA10, ARA18, PLK20]. The real-time collaboration in SEED is designed to supply communication channels, in the presence of failures on control/data-plane management and data delivery.

4.4 Enhanced Failure Management

4.4.1 Insufficient Standardized Causes

Standardized causes provide a good source for failure diagnosis. However, they are insufficient for devices in three aspects. First, they cannot cover all control/data plane failures. Failures from operators' customized policies, such as the supported device list [Ver22b], do not fit into any standardized causes. Second, the data delivery failure could happen due to gNB/core congestion. Without knowing it, the reset may further increase the loads. Third, the failure's root cause could be unspecified without a recovery action. The coarse-grained information is insufficient for failure recovery.

Therefore, the SIM needs more information for failure diagnosis. Thus, we leverage the infrastructure assistance for SIM diagnosis to cover diverse failures (§4.4.2). We further show how SEED automatically handles failures with an unknown root cause (§4.4.3).

4.4.2 Infra-Assisted SIM Diagnosis

We first introduce infrastructure assistance for the diagnosis. This component leverages the deployed metrics in the infrastructure, which avoids redundant processing and is scalable for massive devices. We then elaborate on how SIM diagnoses failures with information from the infrastructure.

Infrastructure Assistance The infrastructure classifies failures with a decision tree as

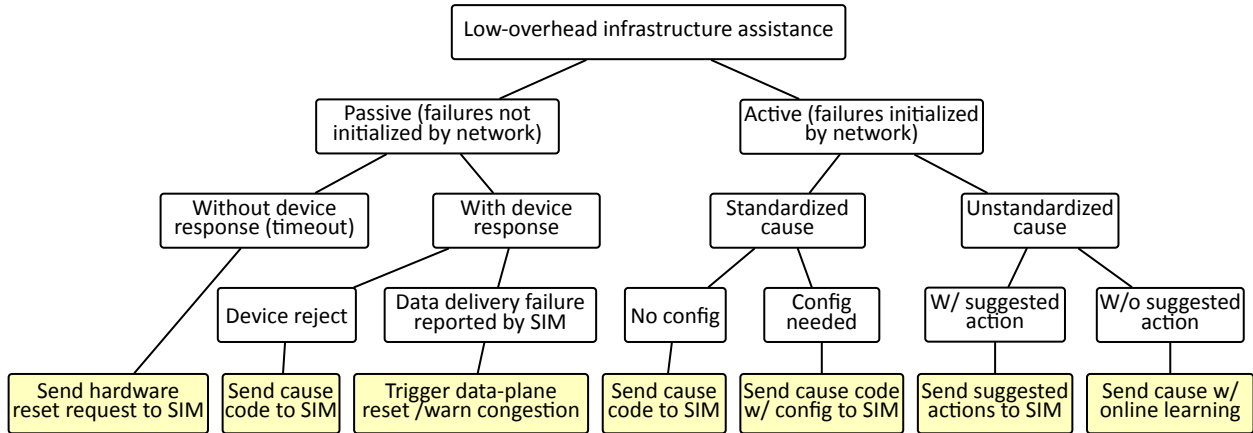


Figure 4.7: Low-overhead infrastructure assistance.

shown in Figure 4.7. It then sends the corresponding assistance information to the SIM with real-time collaboration (§4.3.5). The assistance information includes four types: failure causes, suggested configurations, suggested reset actions, and congestion warnings. SEED acquires them from the existing monitoring and management functions in current 5G infrastructure [Mag22b, KSK21] to assist failure diagnosis without complex processing.

The infrastructure classifies the failures into two types: passive and active. The passive type includes failures not initialized by the infrastructure, such as device response timeout, device reject, or SIM-reported data delivery failure. Standardized causes are sent to the SIM as §4.3. For customized failures, it sends suggested reset actions for SIM handling. It further notifies the SIM with cell/core congestion. The active type includes network-initialized rejects. In addition to standardized failures, the infrastructure provides customized causes with suggested actions to cover failures from customized policies. For causes without suggested actions, we propose an online learning algorithm to handle them (§4.4.3).

SIM Diagnosis SIM receives the four types of assistance info and performs the following actions accordingly. The SIM applet stores all supported failure causes and assistance info parsing functions. They follow a similar decision tree scheme at the network side and could be deployed with limited SIM processing and storage. SIM handles standardized failures and refreshes configurations as in §4.3.4. The SIM performs the suggested reset action for

customized failures, enabling operators to deploy handling for new failures. When the SIM receives the congestion warning, it does not trigger the reset but waits for a timer embedded in the message. SIM parses the assistance information from the infrastructure, and handle diverse standardized and customized failures accordingly with the multi-tier reset. It further notifies users of failures requiring user actions to recover (e.g., reactivating the data plan).

However, there are still failures causes without corresponding handling. The infrastructure may map unstandardized causes to policies or modules but do not have any clues for the device handling. We further design an online learning algorithm to handle failures without a known handling action. We elaborate on it next.

4.4.3 SIM Handling with Online Learning

While the root cause is unclear, previous devices' successful handling probably works for new devices facing the same failure. Although failures may appear from different functions in one module, the multi-tier action directly resets the whole module and has similar effectiveness for various devices. SEED proposes an online learning algorithm to crowdsource the handling history from SIMs, and picks out suggested actions when the same failure happens in the future. The infrastructure and SIMs keep evolving and automatically train the model for failures with unknown handling.

The online learning algorithm (Algorithm 1) includes the SIM side (line 1-7) and infrastructure side (line 8-17). When the infrastructure does not know the handling action for a failure, it generates a customized code to identify the failure, such as the conflicting policies or modules. The cause code is sent to the SIM through SIM-Network collaboration (§4.3.5). SIM tries all the supported retries and resets sequentially from the data plane to the hardware (line 2). It records the successful handling that resolves the issue and notifies the infrastructure with OTA (line 3-7). The infrastructure crowdsources SIM records and updates the network-side record (line 8-10). It then sends out suggested action for part of later devices controlled by a learning rate lr . Otherwise, the infrastructure does not suggest

Algorithm 1: Collaborative Online Learning

```
1 def SIM-RecvUnknownFailure(cause):
2   for action  $\leftarrow$  [B3, A3, B2, A2, B1, A1] do
3     if DoRecovery(action) == success then
4       SIMRecord[cause][action]  $\leftarrow$  +1
5       break
6   if SendToInfra(SIMRecord) == success then
7     SIMRecord = dict[][]
8 def Infra-Crowdsorce(SIMRecord):
9   for cause, action  $\leftarrow$  SIMRecord do
10    NetRecord[cause][action]  $\leftarrow$  +SIMRecord[cause][action]
11 def Infra-SendUnknownFailure(cause):
12   if cause  $\in$  NetRecord then
13     sgstAction = argmax(NetRecord[cause])
14     if rand() <  $\frac{1}{(1+e^{-lr*size(NetRecord[cause])})}$  then
15       SendtoSIM(cause, sgstAction)
16       return
17   SendtoSIM(cause, null)
```

any actions, ensuring that the model is trained and evolving (line 14). If the suggested handling failed, the SIM takes the same action as receiving unknown failure and tries all the supported actions sequentially. In our online learning procedure, the SIM only stores customized error codes and corresponding actions. The data volume is small enough to be held within the limited SIM storage. With the collaboration between SIMs and the infrastructure, online learning provides automatic failure handling for unknown causes. The decision model also evolves gradually without heavy training, which is lightweight and scalable for massive devices.

4.5 Implementations

Figure 4.8 shows the implementation of SEED. The operator owns controls for all SEED components in practice, including the infrastructure module, SIM applet, and the carrier app.

Solution prototype We develop a SIM applet on Javacard-based eSIM [ZDG21], which is compatible with most mobile OS (e.g., Android, iOS, etc.). The applet contains 1244 lines of Java with two modules. The diagnostic module receives the infrastructure assistance information through the modem with APDU interface [ETS19b], and app/OS failure report through the carrier app with TelephonyManager API [And22c]. The decision module uses SEED-U mode by default for the multi-tier reset. For SEED-U mode, the decision module sends proactive commands through APDU to the modem for profile reloading and control-plane configuration updates, and updates data-plane configurations with the carrier app. SIM is notified by the carrier app when root privilege is available. It then enables SEED-R mode and sends AT commands listed in Appendix A.2 to the carrier app for faster failure handling.

We extend the Magma 5G NSA core [Mag22a] with a plugin to assist SIM diagnosis with 1035 lines of C++. The diagnosis assistance module hooks the reject generation functions to acquire the standardized failures. It acquires the latest configurations from the orchestrator API [Mag22c] and extra information such as RAN/core load from Magma NMS [Mag22b]. We extend the orchestrator API to receive SIM recovery records and forward them to the assistance module for online learning (§4.4.3). The real-time collaboration module reuses the Auth Request functions and hooks the PDU establishment handling function. The information is encrypted with 128-EEA2 and integrity protected with 128-EIA2 using the pre-shared in-SIM key to prevent information leakage and malicious requests.

We develop a carrier app with Android UICC Privilege API [And22g] to update carrier configurations, and receive data stall notifications, etc. It contains two modules with 842 lines of Java. The failure report service receives app reports with Android Service [And22b] and OS reports with Connectivity Diagnostics API [And22d]. For unrooted devices, the recovery action module updates configurations with UICC Privilege API. If it detects root privilege with Runtime API [And22a], it notifies the SIM to enable SEED-R mode for it to trigger AT commands.

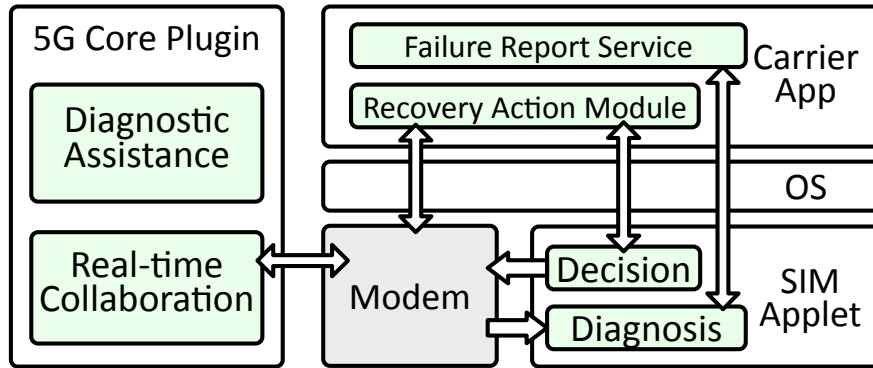


Figure 4.8: SEED implementation components.

Deploying SEED in practice The operators have access to all components that SEED involves, including the core, SIM, and the carrier app. Therefore, SEED is a viable solution that can be deployed by operators alone, without any help from modem or phone vendors. Besides, SEED extends the current standard without changing any existing protocols. Deploying SEED thus does not affect any operating 5G functions.

Incremental deployment Operators can gradually deploy SEED, as a partial implementation already diagnoses some failures. They can first deploy infra and SIM Applet modules to support diagnosis and handling of control/data-plane failures. These modules can cover 63% of failure cases in the traces. The first stage is easy to deploy: all necessary info for the network module can be extracted from core signaling, while the SIM applet could be updated through a readily-available OTA channel. The operators can then update the carrier app to include a failure report service and action module. The carrier app has been widely deployed by operators [Ver22a, Ver22c]. With the enhanced failure handling added, *all* considered failures in this dissertation can be diagnosed and handled.

4.6 Evaluation

We evaluate how SEED diagnoses and handles failures. We first evaluate the overall performance on the testbed with failures in our datasets. We also compare the application

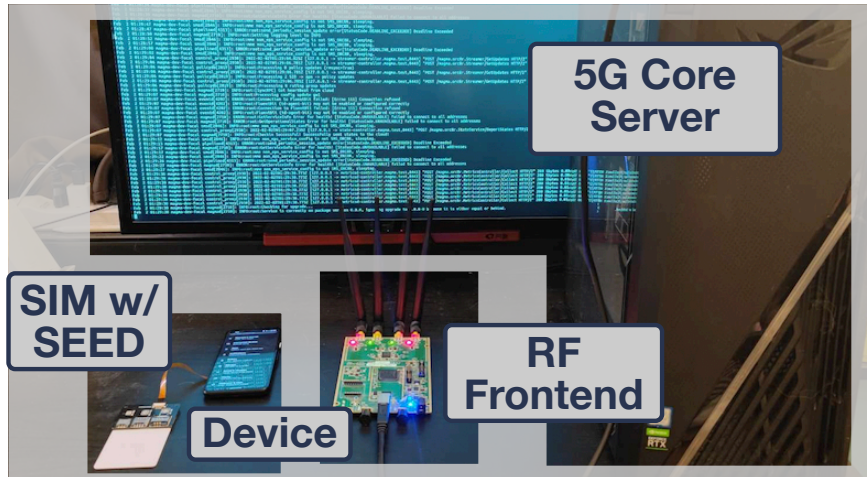


Figure 4.9: Experimental testbed setup.

performance between SEED and existing failure handling schemes. We then assess SEED overhead, diagnosis time, and recovery speed for multi-tier reset with and without root privilege.

Experimental Setup We implement the diagnosis assistant module with 5G-compliant Magma Core [Mag22a] on an Ubuntu 18.04 server with i7-9700K 8-core CPU and use USRP B210 as the RF frontend (Figure 4.9). Our testbed utilizes the 5G-NSA. The long timers are shared between SA and NSA and dominate the disruption time during failures. Moreover, most of SA failure cause codes have inherited NSA error codes. Although SA splits its core functions into different components, the signaling processing and transmission times between SA and NSA incur negligible difference for SEED. The results thus also reflect SEED performance for SA deployment. We deploy the SEED applet on a Javacard-based eSIM [ZDG21] with 180KB EEPROM and 8KB RAM. We assess the performance on Google Pixel 5 with Qualcomm Snapdragon 765G running Android 12.

4.6.1 Overall Performance

4.6.1.1 Comparison of failure diagnosis and handling

We first examine SEED overall performance. To compare it with the current modem-based solution, we utilize the dataset in §4.2.1 to evaluate how many control/data plane management failures SEED could handle. We extract failure traces from the dataset and reproduce failures on the testbed to assess SEED handling. For data delivery failures, we compare the current Android scheme and SEED handling.

Failures	Handling	Median	90th
Control Plane	Legacy	12.4	1024.0
	SEED-U	8.0	76.7
	SEED-R	4.4	48.6
Data Plane	Legacy	476.0	2659.4
	SEED-U	0.9	1.0
	SEED-R	0.6	0.7
Data Delivery	Legacy	31.2	45.7
	SEED-U	1.1	1.3
	SEED-R	0.4	0.7

Table 4.4: Disruption (s) percentile with legacy handling and SEED.

For control-plane management failures, 89.4% of failures in the dataset could be handled by SEED. The remaining cases are due to identity authentication failures from unauthorized subscribers. Table 4.4 compares the disruption time with existing device failure handling and SEED. Without root privilege, SEED-U could reduce the median disruption time by a factor of $0.6\times$ ($12.4\rightarrow 8.0$ s). SEED-R further speeds up the recovery and reduces the 90th percentile disruption by $20\times$ ($1024.0\rightarrow 48.6$ s). The waiting timer (2s in testing) in SEED before triggering control-plane failure handling ensures that the transient failure will not be delayed by reset. With the timer, SEED control-plane handling only causes longer disruption for 5% (SEED-U) and 2% (SEED-R) failures.

For data-plane management failures, SEED handles 95.5% of the cases in the dataset with its configuration update and fast data-plane reset. Other cases are from expired subscribers

and require reactivation of their data plans. The failure handling without root privilege could recover 90% of cases with $<1s$ disruption. With the root privilege, half of the failure cases could be recovered within 0.6s, which reduces the disruption time by $792\times$. SEED prevents the long disruptions incurred by repeated, blind retries at devices.

We further evaluate how well SEED handles data delivery failures. Our experiments show that, if data delivery failures are induced by widely reported incorrect network-side configurations (e.g., TCP/UDP blocking, etc.), Android or modems' naive retry schemes cannot recover from them. Current application-level tools (e.g., NetMotion [Net22a]) also rely on the ongoing data connection and cannot report the failure under traffic blocking. In contrast, we validate that SEED successfully transmits failure reports, which can trigger network-side policy checking and updating for failure recovery. For failures that could be recovered from reconnections (outdated gateway status in mobility, etc.), we compare the choices of Android sequential retries and SEED multi-tier reset. The Android timers between recovery actions are set to the recommended configuration values (21s/6s/16s) in [LLL21]. Despite with shorter timers, Android still incurs more than 31.2s disruptions for 50% of cases. In contrast, SEED fast data-plane reset and modification handle all cases in the experiments, and recover the data connection within 0.4s for 50% of cases and 0.7s for 90% of cases.

4.6.1.2 Reducing Application Disruption

We further examine the failure impact on various applications with current device handling schemes and SEED. In the experiment, we assess both the SEED-U and SEED-R modes. We measure the average app disruption time on five types of latency-sensitive applications, including video (with YouTube [You22]), live streaming (with Twitch [Twi22]), Web browsing (with Chrome [Chr22]), navigation app (with Google Maps [Goo22]), and an edge AR application developed by us. The video app has its long buffering time ($\sim 30s$) while the live streaming possesses a shorter buffer ($\sim 3s$). The Web browser visits the social network site, and the navigation app periodically uploads its location for the latest traffic information.

The AR app keeps sending the camera view to the edge and retrieves real-time recognition results without a video buffer. We collect traffic traces of five applications and develop a background daemon to emulate the corresponding app’s traffic pattern and send failure reports for the application.

Apps	C-plane (s)			D-plane (s)			D-Delivery (s)		
	Leg.	S.U	S.R	Leg.	S.U	S.R	Leg.	S.U	S.R
Video	68.3	1.1	1.0	184.5	0.0	0.0	75.0	0.0	0.0
Live Stream	79.2	4.3	3.5	199.2	1.5	1.1	105.4	0.5	0.0
Web	80.3	6.8	5.4	200.8	1.8	1.6	110.5	0.8	0.3
Navigation	78.3	5.0	4.1	199.9	1.3	1.2	106.7	0.2	0.0
Edge AR	81.9	6.7	5.7	201.9	2.6	2.1	108.2	1.3	0.4

Table 4.5: Average app disruption (s) with legacy (Leg.) failure handling, SEED-U (S.U) and SEED-R (S.R).

SEED reduces failure recovery time for all five applications, as shown in Table 4.5. The fast failure report scheme and multi-tier reset allow the video app to tolerate all data-plane management and data-delivery failures in experiments. SEED reduces disruptions by up to $67\times$ ($68.3\rightarrow 1.0s$) for control-plane failures. For live streaming with a short buffer, SEED reduces the average control-plane failure time to 4.3s (SEED-U) and 3.5s (SEED-R). For data delivery failure, SEED-R could still handle such cases and mask user-perceived disruptions. For Web browsing, SEED reduces disruptions from $80.3\sim 200.8s$ to $0.3\sim 6.8s$ for various failures. SEED also reduces the navigation app disruptions from $78.3\sim 199.9s$ to $0.2\sim 5.0s$ (SEED-U) or $0\sim 4.1s$ (SEED-R). The AR app is the most disruption-sensitive app. Although Android is reconfigured with a shorter action timer, its limited detection scheme takes more than 1 minute to detect the data stall failure and recovers after 108.2s disruptions on average. In contrast, SEED takes the fast data-plane reset approach, and recovers the AR service within 1.3s (SEED-U) and 0.4s (SEED-R) on average.

4.6.2 Micro-Benchmarks

4.6.2.1 Lightweight Failure Diagnosis

We next examine the SEED scalability at the network and the overhead at the device. Our experiments confirm that, SEED is scalable to large device population, and lightweight with low overhead at devices. For the network, we use the magma RAN/UE emulator to emulate loads in the core. We emulate 200 devices performing attach/detach procedures randomly and trigger failure events with different frequencies. Figure 4.10a shows the average CPU utilization with the default magma core with and without SEED. SEED incurs only 4.7% extra CPU processing, in the stress test of artificially injecting 100 failures per second. SEED scales with decision-tree-based failure diagnosis without heavy processing. The number of extra signaling messages (Auth Request/Failure or PDU Session Estb Request/Reject) from the real-time collaboration is marginal compared with the normal control/data plane procedures.

We further gauge the overhead at the device side. The SEED diagnosis is based on SIM's built-in processor and RAM, which is more energy efficient compared with the phone's CPU. By default, we measure the device battery consumption without background application traffic. We then run a stress test that triggers the SIM diagnosis once per second to quantify its energy overhead. As shown in Figure 4.10b, SEED consumes an extra 1.2% (5.4%→6.6%) of the total battery in 30 minutes. Given that the failure frequency in our test is much higher than in reality, the SIM diagnosis incurs negligible overhead. We further compare SEED with the device-side cellular diagnosis application MobileInsight [LPY16]. MobileInsight relies on the message decoded from the diag port for analytics and consumes an extra 8.5% (5.4%→13.9%) of the total battery in 30 minutes. SEED thus performs lightweight diagnosis at the device.

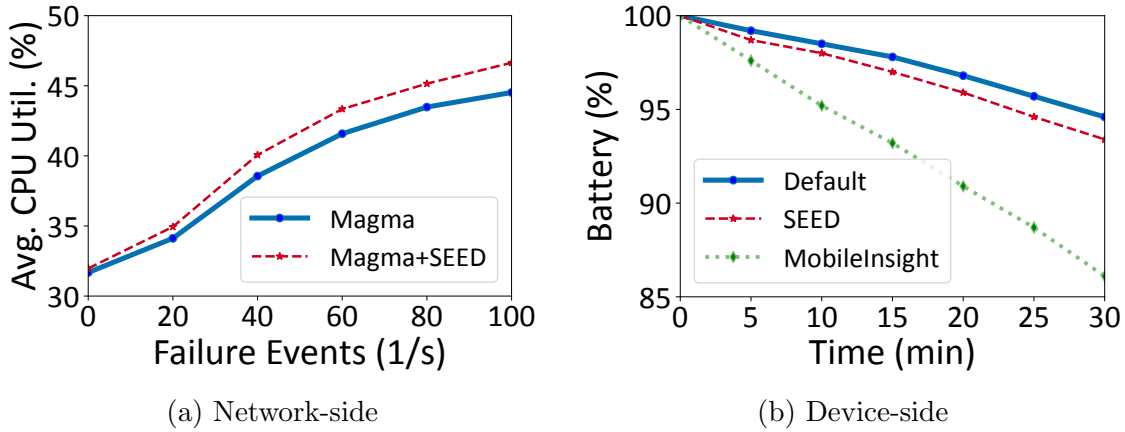


Figure 4.10: Diagnosis overhead on network/device side.

4.6.2.2 Real-time SIM-Network Collaboration

SEED enables the real-time collaboration with standard-compliant signaling messages. Figure 4.11 shows the total latency of network-to-device (downlink) and device-to-network (uplink) directions. For the downlink, when the network detects the failure, it first prepares the message with extra information and encodes it into the Authentication Request, which takes 12.8ms on average. The transmission takes 41.2ms on average from the message sent out to the ACK received. On the device side, SEED provides APIs for App/OS failure report to speed up the failure detection. The preparation includes information reporting, SIM encoding, and message generation, which takes 35.9ms on average. Then the transmission takes 46.3ms on average to notify the network side for further actions. Compared with the Android failure detection, which needs 1.8 minutes to detect the failure, SEED speeds up the failure detection stage with fast SIM-Infra collaboration.

4.6.2.3 Multi-tier Reset

With diagnosis information from both sides, SEED performs the multi-tier reset for fast recovery. Compared with the legacy level-by-level sequential retry, SEED directly resets the corresponding module, eliminating long waiting interval between actions. For baseline,

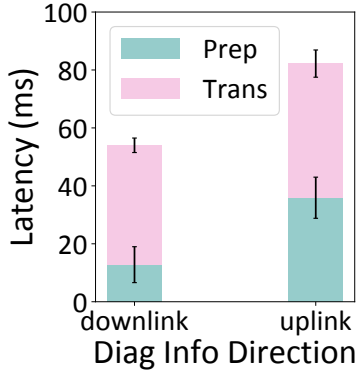


Figure 4.11: SIM-Infra collaboration latency.

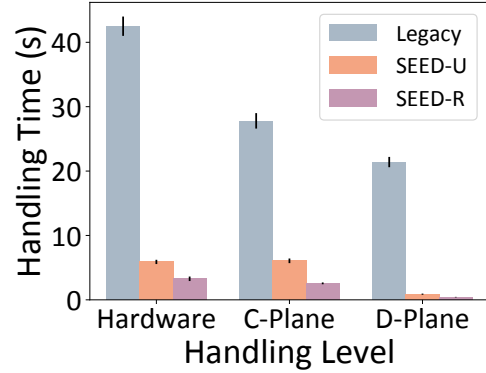


Figure 4.12: Recovery time for multi-tier reset.

we use Android sequential retry with the recommended intervals (21s/6s/16s) between four actions in [LLL21]. Although these intervals are much shorter than the Android default 3-min interval, it still causes a long time to trigger handling actions.

As shown in Figure 4.12, the legacy scheme takes 42.5s on average to reset the hardware. Without root access, SEED takes 5.9s on average for hardware reset. SEED further speeds up the hardware reset with root privilege leveraging AT commands, which takes 3.3s and reduces 92% waiting time. For control-plane reset, the legacy scheme takes 27.8s. The SIM parameter updates (A2) need to be combined with reloading to trigger the control-plane reset and take 6.1s. With root privilege, the control-plane reattachment takes 2.6s for control-plane reset. The legacy scheme does not reset the data plane but all TCP connections, which still needs 21.4s to trigger the failure handling. SEED triggers the carrier app to update configurations for the data-plane reset (A3) and designs fast data plane reset/modification with root privilege (B3), which takes 0.88s and 0.42s, respectively. SEED multi-tier reset shows a fast failure handling without long level-by-level retries.

4.6.2.4 Online Learning

The current public datasets do not provide the infrastructure-side failure traces. We validate the effectiveness of the online learning algorithm by triggering failures at our testbed. In our

experiments, 6 phones of different models (Google, Xiaomi, etc.) are connected to the testbed network. On the network side, we choose 4 control-plane and 4 data-plane functions and manually trigger failures for each function 50 times to generate unstandardized failures. The network customizes failure codes based on the failed function and performs online learning for future recovery suggestions. Our results show that the crowd-sourced SIM records correctly classify all failures into control or data plane failures and recommend corresponding reset actions, which shows the effectiveness of the online learning algorithm.

4.6.3 Security Analysis

Our analysis shows that SEED does not degrade the legacy SIM security. The applet could only be installed with the carrier’s key; adversaries cannot modify or replace it. Only the operators could update the SIM configurations, or perform reset from the SIM. The SIM-infra communication is encrypted and integrity protected with the pre-shared in-SIM key with the message counter, which uses the same crypto algorithms as the 5G signaling. The new applet interface within the assistance information is protected with cellular-grade security. The in-SIM key is hard to be compromised by attackers, thus the information is hard to be faked and skip the SIM checking. The applet leverages existing channels provided by the mobile OS to communicate with the modem/carrier app, which does not induce new loopholes. At the device, only the application matching SIM-embedded signatures could acquire the carrier privilege. The carrier app ensures SIM security by isolating direct communication between user apps and SIM. The carrier app further checks and filters the failure report inputs to ensure security.

4.7 Discussion

SEED focuses on failure diagnosis and treatment for the 5G protocol stack, but does not explicitly diagnose and handle underlying radio link issues or application-level failures. Such

physical-layer or app-level issues may affect control/data-plane management and data delivery. The resulting failures will be implicitly detected and handled by SEED. SEED can be further extended to support radio condition diagnosis with runtime modem measurements and gNB information on dynamic radio signals.

SEED leverages the undefined fields of the 5G signaling messages within the 3GPP standards, thus being compatible with current mobile OSes and radio access networks. SEED still requires changing the SIM applet and the core network. For SIM/eSIM, operators could update the applet through OTA. The eSIM uses a programmable chip to store SIM profiles from different operators. eSIM supports the SIM's applet format, and SEED applet could be directly applied. For the network side, operators with the cloud-based core network implementations could deploy SEED with software updates at the core. A new module can be added to handle diagnosis messages and perform online learning. SEED only introduces a small amount of extra signaling, thus unlikely to trigger anomaly detection deployed by operators. If false alarm is triggered, operators may readjust the related filter rules for diagnosis messages.

The SEED-R mode requires root access to send the AT commands. The current standard has supported the SIM to trigger AT commands directly at the modem with proactive commands [ETS19b]. It has been deployed on some IoT modems [U b22], but not on current 5G smartphone modems yet. If the modem enables the interface, SEED becomes a rootless solution. With the current modems, if the OS provides APIs for the carrier app to initialize AT commands without root, SEED could also become rootless.

SEED can be adapted to diagnose new 5G functionalities. One upcoming feature is network slicing [Fie21, FPE17], where failure could arise to a given slice. Although this increases the complexity of detection and handling, SEED enables fine-grained diagnosis and handling. Therefore, it could reset or modify the failed network slice without affecting other functioning slices.

4.8 Related Work

Mobile failure diagnosis has been an active topic for years. It is an important area, as a correct diagnosis result helps optimize device performance [YLL18, LDP16, LYP17, LDL16, DLH20] or fixes RAN/core [PEC18, AJI20]. Unfortunately, such diagnosis typically relies on the one-side information, either at the device [LPY16, YZH20, XZZ20] or inside the network [FR21, NET22b]. The lack of panoramic view makes diagnosis slow and error-prone. On the other hand, our failure diagnosis combines the information from both sides with a novel SIM-based solution that enables collaboration between the device and the network. To the best of our knowledge, SHIELD is the first work that proposes SIM-based failure diagnosis. Current commercial diagnosis tools such as NetMotion [Net22a] collect both-side information, but only monitor the high-level metrics (e.g., network performance, connection drop rate, etc.) without failure diagnosis in the cellular stack. The collaboration between the network and the device also halts once the data connection is broken upon failures [SIM22]. In contrast, SHIELD diagnoses cellular failures and performs runtime handling even if the data connection is broken or not fully established.

Prior efforts focus on other issues and cannot achieve failure handling with decent speed and accuracy. [LLL21] measures cellular reliability and optimizes the existing Android error handling scheme; it lacks fine-grained failure diagnosis and handling. Meanwhile, network-side diagnosis [PEC18] can barely help the devices recover from failures. Our work bridges the network and the device, and achieves runtime, fine-grained failure handling. Unlike conventional data center network failure diagnosis [FLM20, ZKC15] that utilizes active probing [GYX15] or trace monitoring [KSK15], our design leverages the 5G standardized failure causes with readily accessible metrics for diagnosis and handling, thus keeping the solution light-weight and scalable.

4.9 Conclusion

The global rollout of 5G mobile systems is underway. Similar to every large-scale networked system, failures become the norm, rather than exceptions, in 5G. This has been confirmed by recent empirical studies [XZZ20, LLL21]. As 5G is going to higher radio spectrum and the cell size is getting smaller, frequent handovers further aggravate the chances for failures. If left unattended, such failures will affect the normal operations of 5G applications, particularly those emerging ones (e.g., AR/VR/MR), due to prolonged network disruptions. The current solutions do not diagnose the error causes and use the blind, sequential retry approach to failure handling.

In this chapter, we describe the design, implementation, and evaluation of SEED, a novel SIM-based solution to 5G failure diagnosis and handling. SEED leverages the available error codes carried by standardized 5G signaling messages for root cause inference. It further enhances the diagnosis with a simple, domain-specific machine-learning algorithm. SEED takes adaptive, multi-tier reset/redo actions (reset protocol operations, refresh outdated configurations, reload profiles, etc.) once the failure cause is inferred. Our evaluation has confirmed the viability of SEED.

For fast adoption and deployment, we take the operator’s view in the design of SEED. As the global 5G rollout is ongoing, we believe the operator is in the best position for 5G failure management solutions. The components of SEED can be readily installed to 5G subscribers when they activate their devices with the carrier. The software updates can be easily completed with the current operators’ practice. We are working with a prime US operator for assessment and early trials. In the broader context, we believe 5G failure management needs more activities from the research community; this work describes our initial effort along this direction.

CHAPTER 5

SHIELD: Enhancing High 5G Availability with SIM

In this chapter, we present our SIM-based solution to address 5G connectivity outages, ensuring the high resiliency requirements for industrial and mission-critical 5G IoT systems. We introduce SHIELD, a novel SIM/eSIM-based approach to highly available 5G systems. SHIELD provides a device-based, SIM-centric software solution, leveraging the in-device SIM card and a plug-and-play, miniature, software-defined receiver hardware. Our evaluation demonstrates that our SIM-based method achieves two to three nines of availability (with potential for four to five nines), while maintaining energy efficiency.

The organization of this chapter is as follows: §5.1 outlines the current cloud-based 5G IoT systems and existing practices for handling connectivity outages. We propose wireless-conditioned availability as a metric to evaluate 5G availability and analyze the sources of unavailability in §5.1.1. §5.3 introduces the SIM-centric software-hardware co-design, which aims to achieve high 5G availability. We discuss the implementation in §5.4 and evaluate the approach in §5.5. In §5.6, we explore how our solution could be extended to attain four to five nines of availability. We present the related work in §5.7 and conclude this chapter in §5.8.

5.1 Cloud-based 5G IoT System

The cloud-based 5G Internet-of-Things (IoT) system is shown in Figure 5.1. Its main function is to collect data from sensors through 5G network connectivity and perform data an-

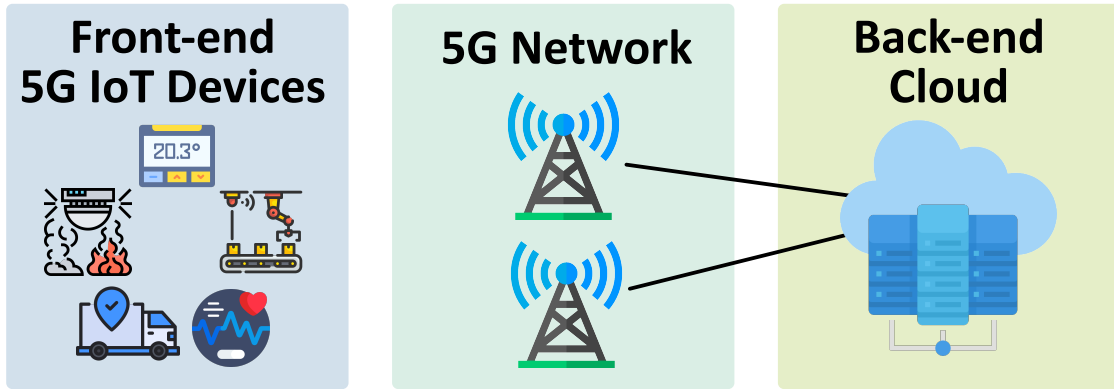


Figure 5.1: 5G IoT system architecture.

analytics in the cloud. The system has three components: front-end devices, back-end cloud, and network connectivity. The front-end devices use sensors to interact with the environment, collect data, and send them to the cloud. The back-end cloud stores and processes data. Network connectivity communicates front-end devices and the back-end cloud.

The 5G IoT¹ has been the most widely deployed technology among various IoT choices [IoT22], e.g., LoRa [LoR23], Wi-Fi [Wi 23], and Zigbee [Zig23], for its universal coverage, mobility support, and carrier-grade service assurance. The front-end IoT device communicates with the 5G base station through its network adapter, called *modem*. The 5G base station uses licensed frequency channels to transfer data and control signaling messages with the device. A SIM/eSIM is attached to the modem, which holds user subscription information, configurations, keys, etc., for wireless connectivity management. Note that, each 5G IoT device must use aggressive power-saving strategies, such as Discontinuous Reception (DRX) and Power Saving Mode (PSM) [3GP21b], to conserve device energy. These modes enable a device to enter a low-power sleep state during inactivity, thereby extending battery life. Specifically, DRX allows a device to periodically “wake up” for data transfer without retaining continuous connectivity. Similarly, PSM enables a device to enter deep sleep mode and reconnect to the network only when needed. These power-saving mechanisms allow 5G IoT

¹5G standards have merged Cat-M/NB-IoT as WB-N1/NB-N1 modes [3GP21b].

devices to achieve high energy efficiency.

The cloud-based 5G IoT systems have been used in various settings such as smart meters [WM 23], fleet trackers [tal23], etc. More recent focus has been on enterprise and mission-critical IoT applications, such as remote pipeline monitoring, autonomous vehicles, logistics, etc. Consequently, ensuring high system availability is essential to 5G IoT services.

The back-end cloud system already ensures high availability. Major service providers have claimed three to five nines (99.9% to 99.999%) of cloud availability over IoT, e.g., Azure IoT Hub [Mic23b] and AWS IoT Core [AWS23b]. The front-end IoT devices also retain high availability to meet the stringent requirements of automated, human-free operations. Measurements show that, only 4% of IoT failure cases result from hardware-related issues [AD22]. IoT applications also implement simpler logic than cloud systems [KK20], thus incurring fewer software-related problems. Moreover, over-the-air updates further help to resolve software issues [EB22].

Compared with the back-end cloud and front-end devices, 5G IoT connectivity becomes the bottleneck for high availability. The connectivity issues incur more failures than hardware or software issues at IoT devices [LLL21]. FCC datasets indicate that, a single carrier's coverage could be as low as 40% in rural areas [FCC21]. Therefore, the single-carrier solution seems to be inadequate for 5G IoT.

5.1.1 Current Practice on 5G IoT Availability

As mentioned in §1, mission-critical and enterprise applications present new challenges for 5G networks. Although 5G deployments have primarily focused on broadband devices, offering high bandwidth and low latency, 5G IoT systems demand even higher standards. These IoT devices are often deployed in more stringent environments, such as underground facilities or remote areas, where connectivity requirements are more rigorous. To ensure high availability for 5G IoT, current practices can be categorized into software replicas and

hardware replicas. Software replicas employ multi-carrier technology to switch the attached carrier network. When the connectivity with the current carrier is lost, the device switches to another carrier network for connectivity. To this end, device profiles from multiple carriers will be stored in a single SIM. This approach is already supported by current SIMs [One23, SIM23]. In summary, this scheme takes a reactive approach to network outages, and searches for an alternative network after an outage. The device cannot scan other alternative carriers without disrupting ongoing data transfer [LPD18]. Consequently, this practice results in lengthy data disruption.

The hardware replicas employ dual modems, each connected to a separate SIM card and a separate carrier network. When the primary modem experiences an outage, the backup modem takes over the data transfer. However, hardware replicas lead to increased power consumption at the device. They are only used on powerful gateways, rather than at front-end IoT devices [Pep23]. Moreover, modems rely on timeout-based outage detection, which identifies outages upon 15-second timeout [Tel23, DIG23]. This timeout incurs prolonged disruptions. The backup modem's data session may have already been released due to inactivity [TKS16, Cis23]; this leads to another data session setup before resuming data transfer.

The fundamental issue is that, current solutions focus on fixing wireless outages only. Multiple carrier access may extend coverage to four (three) nines coverage in urban (rural) areas [FCC21]. However, we will show that, even 100% wireless coverage cannot ensure high availability for 5G IoT service.

5.2 5G IoT Availability

At first glance, it seems that 100% wireless coverage ensures high 5G IoT availability. However, this perception is not always true. We thus use wireless-conditioned availability as the metric to gauge 5G IoT service availability. This metric accounts for data connectivity

outages resulting from both wireless outages and power-saving mechanisms. We show that availability fails to reach a single nine (90%) even with 100% wireless coverage.

5.2.1 Wireless-conditioned Availability

We introduce *Wireless-conditioned Availability (WA)* as the metric to measure the 5G IoT service availability. It is defined as the percentage of time that data transfer is available over the device’s wakeup period, given the frequency of connectivity outages (Equation 5.1). T_{avgDis} is the average disruption time of each connectivity outage. T_{awake} denotes the total wakeup time per day of the device. The connectivity outage frequency f_{outage} is determined by the number of daily outages (wireless outages and device sync-up interruptions).

$$WA(f_{outage}) = \left(1 - \frac{T_{avgDis} \times f_{outage}}{T_{awake}}\right) \times 100\% \quad (5.1)$$

We leverage a real-world cold chain logistics service setting [Pro23] to illustrate wireless-conditioned availability. In this scenario, IoT devices transmit sensory data such as location and temperature every 5 minutes. Each transmission lasts for 5 seconds before the device reverts to the sleep mode for energy conservation. Utilizing a single-carrier solution, public datasets reveal an average connectivity outage disruption time of 37.7 seconds (§5.5.1). With one connectivity outage per day, availability drops to 97.4%. In the driving scenario, data traces indicate 9.6 outages per day, further reducing availability to mere 79.9%.

Our study further identifies two primary sources of unavailability: (1) wireless outages are prevalent, and (2) power-saving mechanisms at the device disrupt synchronization and degrade availability.

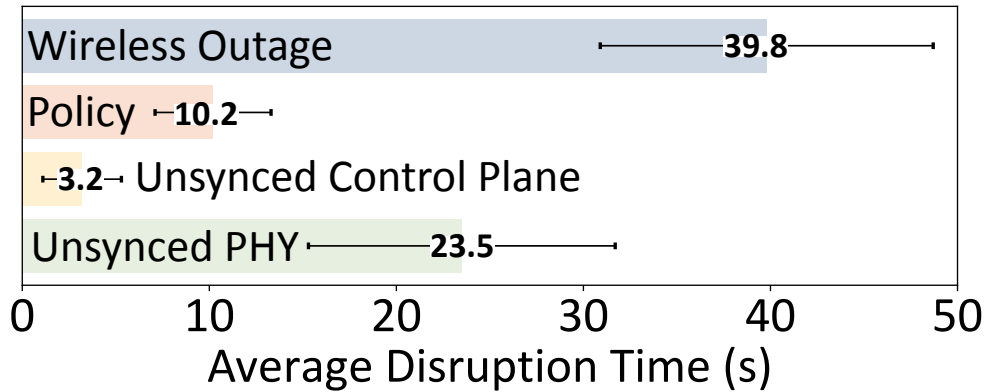


Figure 5.2: Average disruption time (in seconds).

5.2.2 Prevalent Wireless Outage

The widespread wireless outage poses as a major source of 5G IoT unavailability. It signifies the loss of communication coverage from the device to its connected carrier. We study four public datasets with traces from 8 EU and US mobile carriers (dataset details in §5.5.1). On average, wireless outages arise 7.5 times per day with a long 39.8 seconds average disruption, thus reducing the availability to 82.8%. Furthermore, single-carrier coverage could be as low as 40% in rural areas [FCC21], thus exacerbating the issue. Although multiple carriers offer 99.99% coverage in urban areas and 99.9% in rural regions [FCC21], current practices still suffer from lengthy disruptions. This is primarily due to the prolonged carrier-switching process that degrades system availability. Experiments (§5.5.1) reveal that, the average network-switching duration is 21.8 seconds, and availability is only 87.3% with existing multi-carrier solutions. Even when dual modems are used, disruptions still persist due to slow outage detection and response, leading to 19.1 seconds average disruption time; the 88.7% availability fails to reach one nine (90%).

Strong signals indicate available service? Wireless outages definitely arise under weak radio signals. However, strong signals do not mean that the carrier has coverage in the current location. Even with strong signals, access policies may further block service access and hinder data transfer. Such policies are typically configured by operators for various device

categories and services [LPZ21]. They include blocking specific base station access reserved for special user groups, or critical services to IoT devices (such as voice and emergency services [3GP21b]). Consequently, IoT services may remain unavailable even with strong radio signals and good channel quality. When the device connects to a blocked base station, it must disconnect first, reselect and reconnect to another base station; this incurs an average penalty of 10.2 seconds of downtime.

5.2.3 Interrupted Device Sync Up with Power-saving

Unavailability under perfect wireless coverage Given 100% wireless coverage, we next show that, the availability can only reach 89.9% at best. This fails to attain one-nine availability (90%). It is mainly due to the power-saving mechanisms of the IoT device. On average, interrupted sync-ups at a device occur 7.3 times per day, causing disruptions of 22.1 seconds and lowering availability to 89.9%. We next introduce the unavailability caused by power saving.

A 5G IoT device conserves its battery through power-saving mechanisms [3GP21b]. During sleep, the device temporarily shuts off its radio module. Most functions, such as channel measurements and signaling processing, are also suspended for minutes or hours. Despite more energy savings, sleep mode adversely disrupts device synchronization. This affects both the control session and the physical channel, resulting in increased downtime.

Impact on control plane Power-saving mechanisms affect the control plane by disrupting session state synchronization. The device's radio module is shut off upon power saving, silencing network notifications. During sleep, the device's session may be terminated by the infrastructure for various reasons (e.g., load balance [TKS16], session timeout [Cis23], etc.). While broadband 5G devices (e.g., smartphones) receive timely notifications and update their sessions, the IoT devices are unaware that their session has been terminated during sleep [AC19]. Consequently, they must re-establish the session before data transfer, adding

an extra 1.2-7.9 seconds of downtime on top of physical channel disruption.

Impact on physical channel Power-saving schemes can also disrupt physical channel condition sync-up. During sleep, devices are unaware of wireless channel changes. For example, a device’s mobility may cause it to leave the coverage of its prior base station, or enter an area without coverage. Even if the device remains static, the prior channel could become unavailable. A common practice is that, operators adaptively turn base stations on and off at different time-of-the-day to conserve energy [OKL11]. As a result, when the device wakes up, it undergoes a prolonged carrier search. This leads to increased downtime before data transfer can resume. Figure 5.2 shows that, the average unavailable time is 23.5 seconds due to interrupted channel condition sync-up.

5.3 SHIELD Design

We describe SHIELD, a SIM/eSIM-based solution to highly available 5G IoT systems. Using a SIM-centric approach, we reduce data access unavailable time up to $28\times$ while adding only 4.7% extra power consumption to IoT devices. We thus ensure high availability without compromising power savings. We next introduce the design guidelines for SHIELD and its overview, and then elaborate on each component.

Case for SIM/eSIM based minisystem solution The SIM card or eSIM chip naturally exists on 5G IoT devices. It functions as an independent miniature system equipped with processor, storage, and security fences. This allows the SIM/eSIM to securely store user profiles, process data, and authenticate device access. Commercial SIMs have 320KB storage and a 10MHz processor [Emn23]. The resources already suffice to run lightweight applications on SIM/eSIM. Industrial IoT SIMs are also reliable with over-10-year lifespans and can withstand extreme conditions (vibrations, chemicals, corrosion, etc.) [Emn23]. The current practice on SIM is predominantly limited to device identification [SBA21]. Our proposal exploits the SIM/eSIM to offload IoT tasks from the device CPU, resulting in two key

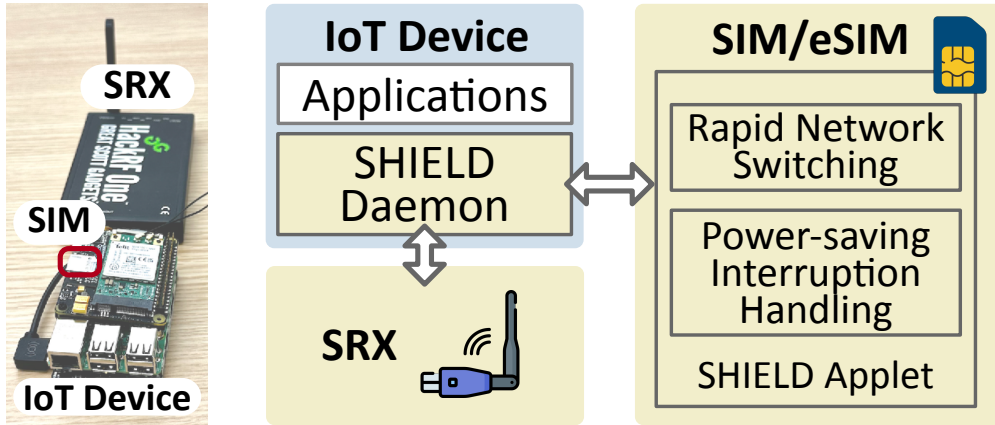


Figure 5.3: SHIELD overview.

benefits: high availability and energy efficiency. We confirm that, the SIM can function as an independent, energy-efficient minisystem within the 5G IoT system, effectively handling IoT tasks including connectivity control, data transfer and security, as well as power-saving management.

5.3.1 Overview

In a nutshell, SHIELD offers a SIM-centric software solution with minimal hardware support. It follows two design guidelines: rapid multi-carrier switching and data-first design.

Rapid Multi-Carrier Switching The single-carrier solutions are limited by restricted coverage [FCC21]. Prior multi-carrier switching [Goo23, One23] also suffers from reactive outage handling, which only addresses connectivity disruptions after outages have occurred, resulting in extended downtime. Moreover, blind search and switching without the knowledge of available networks exacerbate the problem. SHIELD devises a rapid multi-carrier switching scheme. It takes a proactive approach by using a separate, lightweight, software-defined receiver (SRX) to scan and discover alternative networks. SHIELD further verifies the network before switching, eliminating the penalties of blind switching. By leveraging a SIM-based software-hardware co-design, SHIELD ensures rapid multi-carrier switching that

retains high service availability at a negligible energy cost.

Data-First Design Current IoT systems adopt a control-first design, which involves establishing the control plane and data plane sequentially prior to data transfer. However, for IoT services, this session setup procedure results in excessive overhead. The frequent interruptions under the device power-saving mechanism further exacerbate this issue. In contrast, SHIELD adopts a data-first design. It enables data-first operations so that application data can be securely transmitted even without setting up the data plane. SHIELD explores standardized 5G mechanism to embed IoT data in control signaling, and leverages these features without modifying device firmware or the 5G infrastructure. Consequently, SHIELD offers a plug-and-play solution that accelerates data transfer and improves availability.

With both guidelines, SHIELD uses a SIM-centric design to minimize data access unavailability due to wireless outages and interrupted device sync-ups, as depicted in Figure 5.3. This novel approach efficiently addresses challenges in both active and sleep modes. When the modem is active, SHIELD enables rapid network switching in the event of wireless outages, ensuring swift recovery. Meanwhile, during sleep mode, SHIELD effectively reduces power-saving induced interruptions without compromising energy efficiency. We next elaborate on each component.

5.3.2 Rapid Network Switching

Existing multi-carrier switching schemes, e.g., Google Fi [Goo23] and OneSIMCard [One23], take the *disconnect-scan-switch* approach. To initiate switching, the device first disconnects from its current base station to search for alternative carriers. It thus scans all available frequencies to identify and evaluate those alternative carriers. Finally, the device switches to the new carrier network that has strong signal strength.

The current scheme causes extended downtime upon wireless outages, with the average being 39.8 seconds. This is because scanning other networks requires disconnection from

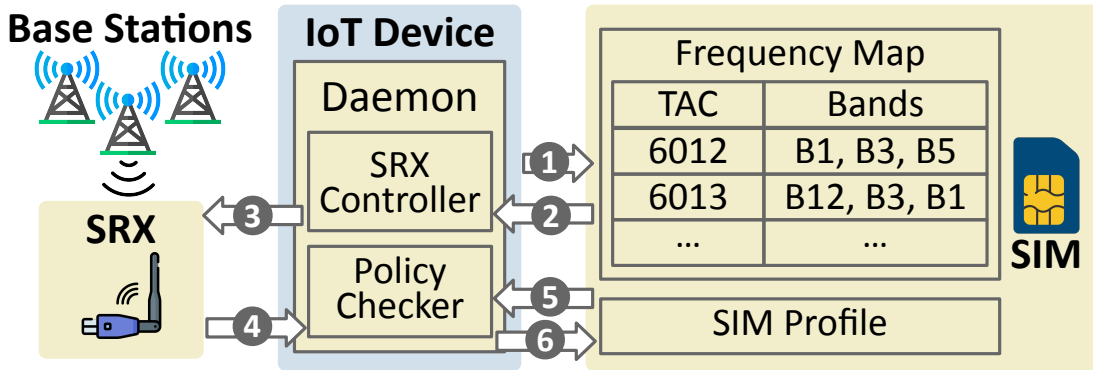


Figure 5.4: Backup network discovery procedure.

the connected base station. This feature thus disrupts ongoing data transfer. Consequently, existing proposals probe alternative networks only during idle states or after outages have occurred [LDP16, One23]. In usage scenarios with continuous bulk data transfer, devices might not have enough idle time to scan backup networks. Thus, the current outage handling is reactive, resulting in prolonged disruptions. Therefore, the following dilemma arises: should the system prioritize the ongoing data transfer, or suspend the current transfer to facilitate rapid scanning before an actual outage occurs?

SHIELD introduces a *retain-discover-verify* scheme for network switching. This approach proactively discovers alternative networks without disrupting ongoing data transfer. The novel non-disruptive discovery is enabled with a software-defined receiver (SRX). The overall procedure has three steps. First, SHIELD retains ongoing data transfer and continuously monitors signal conditions via SIM. Second, SHIELD delegates backup network discovery to SRX, thus enabling swift, non-disruptive discovery. Third, SHIELD checks for policy conflicts using broadcast messages without actual connection to the alternative network. With the retain-discover-verify scheme, SHIELD enables rapid network switching, minimizing downtime upon wireless outages.

We next elaborate on how SHIELD performs the backup network discovery. We further present how SHIELD activates rapid switching while preserving energy efficiency.

5.3.2.1 Discovering Alternative Carrier Network

In addition to the disruptive feature of scanning, current modems use the timeout-based outage detection, which identifies an outage after the 15-second timeout [Tel23, DIG23]. Upon timeout, the modem initiates an exhaustive search to scan all available frequencies in alternative carriers. This brute-force search prolongs the data access downtime during an outage. As devices cannot receive network-side info during such periods, swiftly identifying backup networks without using brute-force search becomes an issue. Moreover, current practices select a network based on the signal strength and then attempt to connect. However, potential blocking policy configurations may render the service to be inaccessible despite strong signals (§5.2.2). Policies are also subject to dynamic changes and may vary across locations. We thus must circumvent policy blocking before trying to establish a connection with the new carrier.

Our approach SHIELD exploits a low-cost SRX to detect alternative, available networks. This allows the device to identify such networks without disconnecting from the current carrier or disrupting ongoing data transfer. SHIELD eliminates blind, exhaustive search by leveraging an in-SIM frequency map. It further verifies the discovered network to avoid policy conflicts before the actual switch. Figure 5.4 shows the procedure on discovering alternative networks in SHIELD. Initially, the SIM learns the current location area from the IoT device (①). It then provides the most probable available frequency bands to the daemon (②). The daemon directs the SRX to initiate network discovery (③). When network information is learned, it is forwarded to the policy checker (④), which assesses potential policy conflicts with the SIM profile (⑤). Finally, the identified backup carrier information is relayed to the SIM (⑥) for network switching.

Preventing blind search SHIELD devises an in-SIM frequency map to prevent the blind search. It facilitates targeted network discovery without exhaustive search. This approach exploits the fact that each IoT network carrier operates on its licensed frequencies, which

can be acquired from publicly available datasets [Cel23]. It does not depend on network-side coordination, and remains operational during outages. The in-SIM frequency map uses the Tracking Area Code (TAC) of network carriers as the key, mapping it to the most probable frequency bands in the location area (e.g., bands 1, 3, 5 for tracking area 6012 in Figure 5.4). Prior to the discovery, the SIM obtains the current TAC from the device (①) and provides the band list (②). Unlike conventional modems, the SRX can perform targeted measurements on these bands (③), thus eliminating exhaustive scanning. When a backup network is found, the SRX immediately returns the results (④). Experiments show that, frequencies provided by the frequency map are available in 98.9% of instances. If no available networks are detected within the candidate frequencies (1.1% of cases), the SRX sequentially attempts other frequencies prioritized by the deployed ratio. On average, the backup discovery process takes about 590 ms, one or two orders of magnitude faster.

Checking for access policy To prevent policy-related access blocking, SHIELD extracts and verifies policies from the broadcast messages of each carrier. 5G base stations wirelessly broadcast these messages. The SRX receives them during network discovery without setting up real connections. SHIELD verifies such messages with the SIM profile to detect access policy blocking (⑤). Specifically, SHIELD examines different levels of policies from the Master Information Block (MIB) message and various types of System Information Block (SIB) messages in 5G IoT. For network-level blocking, SHIELD checks for the network ID embedded in the MIB message. It then verifies whether the ID is on the SIM’s list of accepted network IDs. SHIELD also examines SIB1 for base-station blocking and SIB2 for service-level blocking. If no conflicts are detected and the measured signal is strong, SHIELD provides the frequency and network ID to the SIM (⑥), thus enabling a rapid switch. If policy-based blocking is detected, SHIELD switches to the next candidate frequency and proceeds the discovery process.

5.3.2.2 Retaining Energy Efficiency on IoT

While SHIELD enables rapid network switching, maintaining energy efficiency simultaneously becomes the next issue. SHIELD devises lightweight, SIM-based outage detection. The simplified processing with the SRX further results in marginal, extra energy consumption. With the SIM-based software-hardware co-design, SHIELD ensures high availability while preserving energy efficiency.

SIM-based outage detection SHIELD promptly and proactively detects wireless outages with the SIM. During the active period, the SIM assesses signal conditions and triggers outage alerts for network probing and backup discovery. Figure 5.5 illustrates the SIM-based disruption detection pipeline. The SIM continuously checks the modem every second. It assesses the signal strength (RSRP) and quality (RSRQ) with the current base station (①). An outage alert is triggered based on a preset signal threshold, controlled by a parameter T^2 . When the signal is worse than the threshold, the SIM alerts the device daemon to activate SRX for backup network discovery (②A). Concurrently, it sends probe packets to three preset servers to confirm the connectivity outage (②B). Rapid network switching is initiated, if a backup network is found (③A) and no responses are received from the probing packets. However, if any response is obtained (③B) before discovering a backup network, the identified backup network is cached while server probing continues. Probing continues until either 1) signal strength exceeds the threshold such that the alert ceases, or 2) no response is received. Note that no response from probing indicates an outage and would also initiate rapid network switching. To handle outages under strong signals (e.g., 5G core network failures), the SIM also periodically sends probe packets every 5 seconds. Our study shows that this category contributes about 3.9% of all outages.

Simplified processing with SRX Due to energy constraints, the replication scheme

²The threshold is (RSRP < -90-T dBm or RSRQ < -10-T dB), T=5 by default. An -90dBm RSRP and an -10dB RSRQ indicate comparable signal conditions.

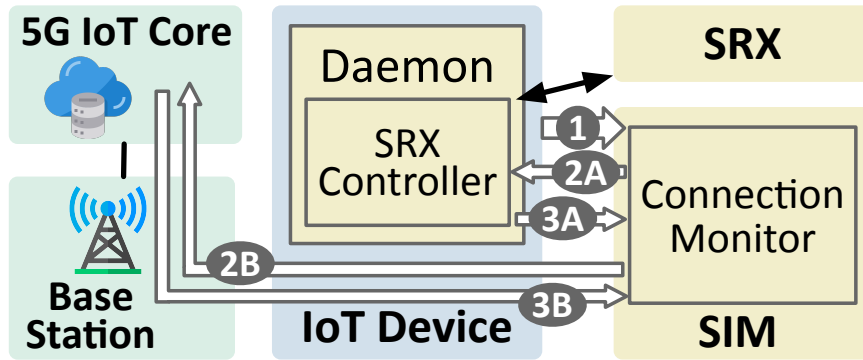


Figure 5.5: Pipeline of SIM-based outage detection.

commonly used in the cloud is not suitable for IoT devices. For example, dual-modem solutions use the replica schemes of one primary and one backup. However, dual-modem solutions consume excessive energy, as the backup modem needs to periodically transmit signals to maintain the backup session. Therefore, such solutions are power-hungry and only used on powerful gateways, rather than IoT devices [Pep23].

The SRX passively monitors without sending any wireless signals. Incorporating the SRX into the design simplifies both hardware and software processing. For hardware, the SRX requires only the radio-frequency (RF) hardware module, rather than the complete set of modem hardware modules (processor, memory, storage, RF, etc.). Software components are also simplified. The host daemon, which processes received signals and decodes 5G messages, focuses solely on the needed tasks. This eliminates additional processing, including transmission, encoding, mobility/session management, and encryption/decryption. In summary, merely 5% of processing is required compared with the commodity modem [GGS16]. Moreover, SHIELD only powers on the SRX when in use, thus conserving device energy even further.

Empirical results Our experiments confirm that, rapid network switching can reduce downtime to just 3.7 seconds, $3.2\text{-}9.2\times$ reduction compared with the current practices (§5.5.1). Together with the concurrent data/control design (§5.3.3.1), disruption is further decreased to 1.3 seconds ($10.8\text{-}28.0\times$ reduction). Furthermore, an unsuccessful connection request due

to policy blocking incurs 10.2 seconds of data access downtime. SHIELD avoids potential blocks by verifying the policy conflicts during discovery. Our experiments show that, SHIELD incurs 4.7% extra energy consumption, enabling rapid network switching with marginal energy overhead.

5.3.3 Handling Power-saving Interruptions

Rapid network switching handles outages when the device is at active state. The power-saving schemes in 5G IoT introduce more issues. We next present how SHIELD manages interruptions when the device is in the sleep mode.

Power-saving instruments effectively conserve energy, but also introduce data access unavailability, as described in §5.2.3. They disrupt device synchronization with the 5G infrastructure, on both updated control session states and which base station to connect to. For example, the control session can be terminated by the infrastructure while the device is in sleep mode. The device might be unaware of such changes upon waking. Moreover, when moving to a new location (e.g., with the driving truck), the device may seek to connect to the prior base station (before sleep) upon waking. In both cases, power-saving mechanisms lead to state inconsistency between the device and the infrastructure. This results in additional downtime for data access.

SHIELD minimizes data access downtime induced by 5G power-saving operations. It employs a concurrent data/control transfer mode, enabling IoT devices to securely transmit data even without establishing a data plane. Furthermore, SHIELD devises a monitor-wakeup-reconnect mechanism, allowing devices to find the updated base station at the current spot upon mobility. Overall, SHIELD still ensures high availability in the presence of power-saving operations.

5.3.3.1 Handling Control Interruptions

The control session could be released by the network side (e.g., due to insufficient resources or load balancing), when the device is in the sleep mode. When the control session is released, its associated context info (aka bearer info, e.g., IP address) mandated by 5G standards [3GP21b] will also be deleted. This will adversely affect the data session when the device wakes up. As a result, the IoT device must reestablish both the control and data plane before data transfer can start. The slow data plane setup upon device wake-up definitely prolongs the data service downtime.

Concurrent data/control transfer SHIELD takes a concurrent data/control transfer approach, allowing a device to securely transmit its IoT data without establishing a data plane. It exploits the standardized Data-over-NAS (DoNAS) mechanism for 5G networks [3GP21b], thus enabling IoT data to be piggybacked into control-plane messages. This approach reduces data delivery latency and eliminates complex data plane setup procedures, such as data radio bearer setup and gateway session initialization. Upon modem wake-up, the SIM directs it to utilize DoNAS mode for data transfer. This mode is readily available on commodity modems and can be configured with common modem APIs [3GP21c]. The DoNAS mode has also been supported on commercial 5G networks to date [Mob20]. Leveraging the SIM-modem interface, the SIM controls the modem and activates DoNAS mode. Importantly, control plane encryption and integrity protection keys are established prior to embedding data into the control message flow. This guarantees that concurrent data/control transfers are secure. Furthermore, SHIELD does not affect standardized data transfer if the control session remains uninterrupted. By utilizing DoNAS, SHIELD provides a secure and fast data transfer solution, particularly in scenarios involving control-plane interruptions.

Plug-and-play SHIELD's design guarantees compatibility with both commodity IoT devices and 5G infrastructure. It explores the standardized 5G IoT mechanism, without modifying device firmware or base stations. This approach results in a plug-and-play solution

that speeds up data service setup. This way, SHIELD effectively minimizes the unavailability resulting from prolonged session setup in 5G IoT systems.

5.3.3.2 Handling Mobility-induced Out-of-Service

During its sleep mode, the device may move out of the coverage of its connected base station, or enter an area without coverage. Current solutions take a *wakeup-timeout-search* approach. Upon wake-up, despite being beyond the coverage area, the device still seeks to connect to the prior, connected base station by default. If the prior base station cannot be detected, it keeps trying to search base stations belonging to the prior carrier, until timeout (typically 15 seconds [Tel23, DIG23]). If still unsuccessful, the device initiates an exhaustive network search and switches to another network. This scheme results in prolonged data access disruption with an average of 23.5 seconds of downtime.

SHIELD employs a different, *monitor-wakeup-reconnect* strategy. It first instructs the SRX to monitor current channel conditions, allowing the device to detect out-of-service from the connected base station during sleep, without impairing power-saving modes. Upon wake-up, the device checks for any detected out-of-service instances. If found, SHIELD directs the modem to execute fast base station reconnect. To accelerate the reconnect, SHIELD ingeniously leverages the IoT *band-lock* capability to reduce the search space, and achieves fast reconnect. This approach effectively reduces out-of-service downtime caused by the device's mobility under sleep.

Monitoring during power saving During the modem's sleep mode, the device daemon periodically activates the SRX to measure the signal from the previously connected base station. If the base station is undetectable, the modem experiences interruptions upon wake-up from its power-saving mode. SHIELD thus searches for base stations associated with the old carrier. If none is available, SHIELD performs backup network discovery. The daemon caches the available frequency and network ID. Upon wake-up, the daemon sends

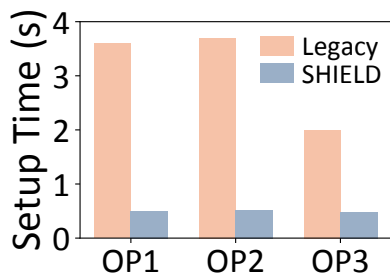


Figure 5.6: Data service setup time of three US operators.

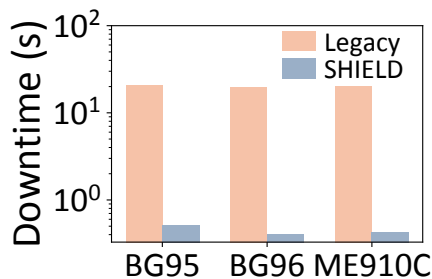


Figure 5.7: Mobility-induced out-of-service downtime.

the cached results to the modem, which initiates fast base station reconnect. Since the SRX is only active for a short period and only receives signals without transmitting signals, its consumed energy is marginal.

The monitoring period can be customized to accommodate diverse mobility patterns. For driving or walking scenarios, configuring a short interval in accordance with traffic patterns facilitates timely outage detection while conserving energy. In contrast, for static IoT settings, a large interval helps minimize unnecessary energy consumption. This allows SHIELD to meet the demands of various IoT applications, ensuring high availability at a marginal energy cost.

Fast reconnect SHIELD uses fast base station reconnect to eliminate long timeout and exhaustive search. Upon wake-up, SHIELD locks the modem to the previously detected frequency if interruptions arise. This locking reduces the search space and enables fast base station search. The current commodity modems have supported *band locking*. Prior efforts used band locking to prevent automatic channel switches under fluctuating signals [Cor23], thus avoiding ping-pong effects [Net23] and enhancing network speed [DLH20]. SHIELD is the first to leverage band locking for fast reconnect to reduce the downtime due to out-of-service. In the event that the previous carrier is unavailable, SHIELD instructs the modem to perform rapid network switching with the discovered backup carrier network ID.

Empirical Results We quantify the data session setup time with three major US 5G op-

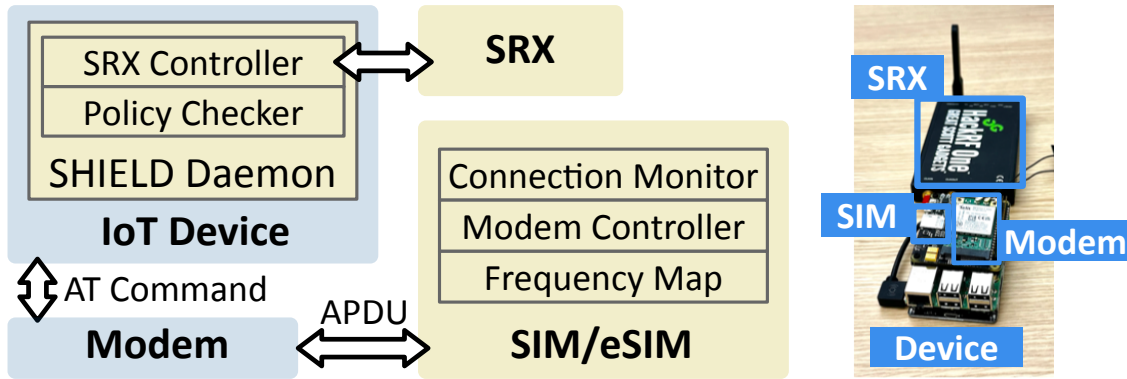


Figure 5.8: Components in SHIELD implementation.

erators. Our study shows that, the current sequential control-data setup requires 3.1 seconds on average before actual data transfer (Figure 5.6). The concurrent data/control transfer enables data service within 497 ms. We also assess the downtime caused by out-of-service from the prior base station during sleep on three commodity IoT modems: Quactel BG95, Quactel BG96, and Telit ME910C. We find that, the wakeup-timeout-search procedure results in an average downtime of 20.2 seconds (Figure 5.7). In contrast, our monitor-wakeup-reconnect scheme reduces the downtime to about 440 ms on average.

5.4 Implementations

Figure 5.8 illustrates the prototype of SHIELD. Its SIM-centric workflow operates as follows. The SIM employs the application protocol data unit (APDU) as its communication protocol [APD23], acting as a byte-stream interface. This allows the SIM to directly control the modem through standardized proactive commands [ETS19b]. While traditionally used for over-the-air SIM profile updates [KR18], SHIELD leverages this channel in an innovative fashion to control the modem for rapid network switching and power-saving induced disruption handling. The APDU also provides a communication channel between the SIM and the host daemon. The host daemon exchanges bytes with the SIM through the modem via AT commands [3GP21c], enabling it to query frequency maps, verify SIM profiles, and

cache discovered network information. SHIELD leverages the commodity hardware HackRF One [Hac23] as its SRX. Based on the SIM’s responses, the daemon instructs the SRX for discovery and monitoring. We next elaborate on the implementation of each component.

SIM Applet The SIM applet comprises three essential modules: the frequency map, the modem controller, and the connection monitor. To store the frequency map within the limited SIM storage, SHIELD compresses the frequency map using compact tables, with each carrier having a corresponding table. SHIELD uses 2 bytes for the tracking-area code (TAC) and a 3-byte representation for the three most commonly used frequencies from the supported carriers. Considering approximately 5,000 tracking areas for three major US carriers [Cel23], the in-SIM frequency map occupies about 25KB storage, and readily fits into the current 320KB SIM storage capacity [Emn23]. The modem controller triggers proactive commands to control the modem to perform essential tasks in rapid network switching and power-saving disruption handling. The tasks include carrier selection, signal condition acquisition, DoNAS mode control, and band locking. The connection monitor enhances outage detection by evaluating signal conditions and initiating probing packets. It further notifies the daemon to activate SRX for alternative network discovery. Notably, the SIM does not actively trigger signal measurements. Instead, the modem measures and caches results automatically, independent of SIM queries. This SIM-based detection ensures marginal energy consumption.

Working with Commercial SIM We developed two versions for the SIM applet implementation. The first aims to validate SHIELD on operational 5G IoT networks. Commercial SIM cards have to be used, but they are closed-sourced and could not be modified. To overcome this limitation, we ingeniously utilize the SIMTrace board [Osm23], originally developed for capturing SIM-modem communications. We modified the board’s firmware to activate APDU injections between the SIM and the modem, thus enabling SHIELD applet functions on commercial SIM cards. This implementation consists of 953 lines of C++ code. The second version aims to validate SHIELD’s feasibility on physical SIM cards. We leverage the publicly available Flora-eSIM [Flo23b], an open-source, Javacard-based eSIM platform.

Javacard is popular for SIM development and shares the same hardware configuration as commercial SIM cards. We implement an applet with 1486 lines of Java code. The second version does not rely on the SIMTrace board and can be deployed via over-the-air software updates on SIM cards or eSIM chips.

Host Daemon The daemon has 1240 lines of C++ code and runs on the 5G IoT device. It has two modules: the SRX controller and policy checker. We implement the SRX controller based on the open-source code of Sonica [DZT21], and extended it to support both Cat-M and NB-IoT. The SRX controller adaptively activates SRX for alternative network discovery and channel monitoring. The policy checker implements a broadcast message decoder to verify potential policy conflicts from incoming broadcast messages, to avoid penalties due to policy blocking. In short, SHIELD is transparent to the IoT application running on the device. The IoT application transfers its data through the networking stack of the device OS as usual without any modification.

SHIELD Power Control The lack of a power switch in SRX poses the new implementation issue of adaptive power control. SHIELD addresses this issue via a new trick. It uses a BIG7 USB hub board [BIG23] to bridge the SRX and the IoT host device. This allows the device to programmatically control the hub board, enabling the daemon to fully power off SRX when not in use. Consequently, SHIELD achieves efficient power management by minimizing energy overhead.

Plug-and-Play The deployment of SHIELD can be plug-and-play on the IoT device. SHIELD design does not modify the device firmware, operating systems, IoT applications, IoT clouds, or 5G IoT infrastructure. The SIM applet could be easily updated through existing over-the-air channels [KR18]. The host daemon and SRX are also plug-and-play on the device.

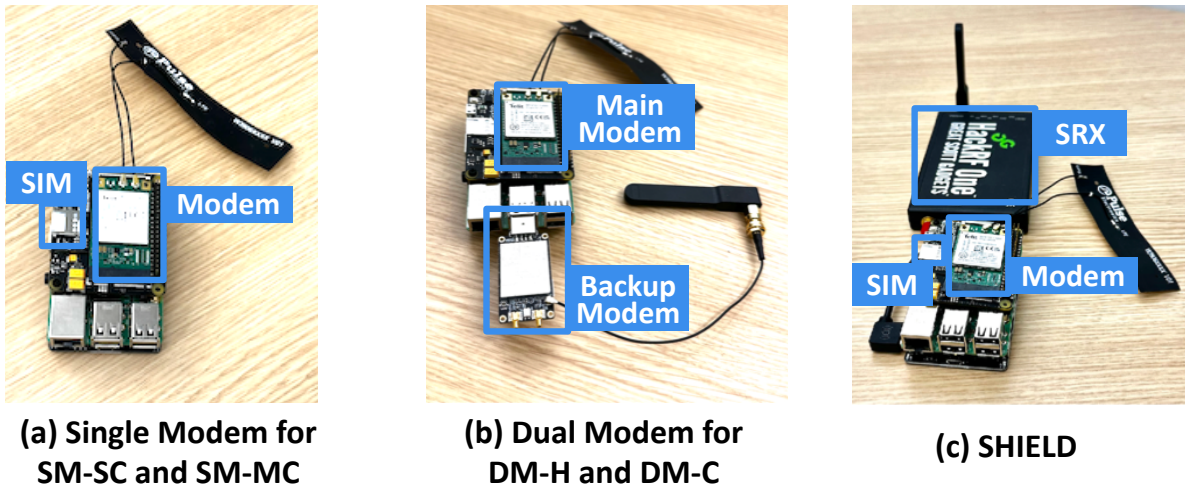


Figure 5.9: Experimental setup.

5.5 Evaluation

This section presents the experimental evaluation of SHIELD and compares its performance with current practices. SHIELD outperforms existing solutions, reducing up to $28\times$ unavailable time. It achieves this high availability while incurring modest 4.7% extra energy consumption. In our showcase IoT applications, SHIELD attains 99.3% availability. These results demonstrate an-order-of-magnitude improvement over existing solutions.

Experimental setup In our experiments, we used the Raspberry Pi 3 Model B+ as the IoT device (Figure 5.9), running Ubuntu 20.04 with 1GB memory and 512MB storage. We employed the HackRF One as the SRX, which was connected to the BIG7 USB hub board, enabling complete power-off when not in use. We use the Telit ME910C1 in the single modem setup and utilize the Quactel BG96 modem as the backup modem in the dual-modem setting for existing solutions. Both modems operated in the 5G Cat-M mode during experiments. To measure the device’s power consumption, we employed the MakerHawk power meter. For the SIM Applet, we utilized two different versions: the commercial network version with the SIMTrace v2 board and the OneSIMCard TCNS-C SIM card supporting T-Mobile, AT&T, and Verizon networks; and the Javacard version deployed on J3R110

Solution	Description
SM-SC	Default: a single modem with a single carrier
SM-MC	Single modem with multiple supported carriers
DM-C	Cold backup modem only wakes up upon outages
DM-H	Active backup transmits data upon outages
SHIELD	Ours: SIM-based mini-system with SRX

Table 5.1: Compared solutions in evaluation.

JavaCard running JCard 3.0.5 with 110KB EEPROM and 4KB RAM. Since commercial 5G IoT devices currently support only 5G Non-Standalone (NSA) signaling, we validate the NSA option on commercial networks. We further confirm the viability of the Standalone (SA) option with the testbed deployed with srsRAN [srs23] running on an Ubuntu 18.04 server with an i7-9700K 8-core CPU.

We evaluate five solutions listed in Table 5.1. Among them, four solutions are existing approaches: single-modem, single-carrier (SM-SC), single-modem, multi-carrier (SM-MC), dual-modem, cold standby (DM-C), and dual-modem, hot standby (DM-H). SHIELD applies a SIM-based software-hardware co-design with SRX. We assess the availability and power consumption for all five solutions. We further evaluate SHIELD on two real-world applications.

5.5.1 Availability Improvement

We use four public datasets to assess the outage frequency f_{outage} of current IoT carrier networks: [KRC22] provides large-scale measurements in Rome, Italy, covering two major IoT network operators. [RQZ18] offers client-side mobile network key performance indicators collected from two major Irish operators. [FHM22] includes datasets of two-year drive tests in Austria, detailing signal conditions, data rates, etc. The fourth dataset is collected by us with over 50GB traces from 2022-Q1 to 2023-Q1 from three major US operators using the public, open-source tool MobileInsight [LPY16]. These datasets contain data traffic details (e.g., timestamps, signal conditions, outages, etc.) covering 1600+ base stations. We assess

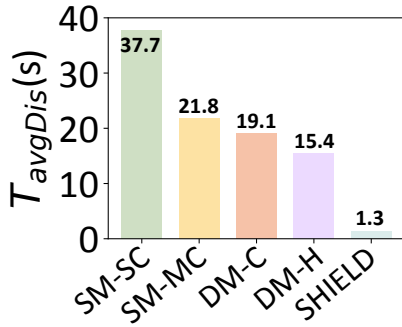


Figure 5.10: Average disruption time of five solutions.

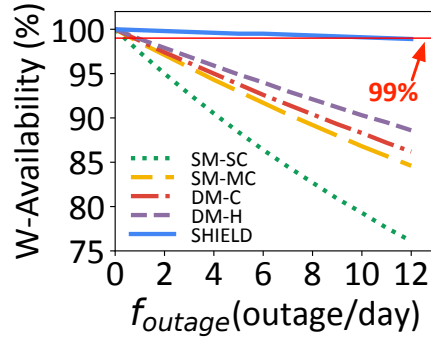


Figure 5.11: Wireless-conditioned availability vs. outage frequencies.

the outage frequencies under three different mobility scenarios covered by the datasets: static, walking, and driving. The results reveal that, the outage frequency f_{outage} for these scenarios is 3.4, 7.3, and 9.6, respectively. We then quantify each solution’s average disruption time T_{avgDis} , and then compare the wireless-conditioned availability (i.e., Equation (5.1) in §3.1) of all five solutions.

We evaluate all five solutions on commercial US carrier networks to determine their average disruption time, T_{avgDis} . For the single-carrier solution (SM-SC), we restrict the IoT device to a single operator in the modem setting. In contrast, the multi-carrier (SM-MC) solution allows the device to freely select among supported carriers. We also implement dual modem solutions with cold (DM-C) and hot (DM-H) standby configurations. We collect and analyze traces for each solution, then assess disruption time during connectivity outages induced by both wireless outages and power-saving interruptions. We use Mobileinsight to quantify the effectiveness of concurrent data/control transfer. We assess the data session setup time with signaling traces from commercial networks. Finally, SHIELD’s effectiveness on the 5G SA option has been confirmed on the testbed.

The results are presented in Figure 5.10. The default SM-SC solution has an average disruption time of 37.7s. SM-MC suffers from slow outage detection and exhaustive network search, resulting in an average disruption time of 21.8s. For DM-C, prolonged timeout-based detection on the main modem leads to extended downtime, and the cold standby modem

	SM-SC	SM-MC	DM-C	DM-H	SHIELD
Static	91.8	95.1	95.7	96.5	99.7
Walking	84.0	90.0	91.2	92.8	99.3
Driving	79.9	87.3	88.7	90.7	99.1

Table 5.2: Wireless-conditioned availability (%) of five solutions under different mobility scenarios.

further suffers from slow session setup, causing 19.1s disruptions. Although DM-H does not require session setup with the hot standby modem, the timeout-based slow detection still results in 15.4s disruption. In contrast, rapid network switching and power-saving interruption handling in SHIELD reduces the average disruption time to about 1.3 seconds, resulting in a reduction ranging from 10.8-28.0 \times compared with existing solutions. These results show that, SHIELD significantly reduces the data access downtime.

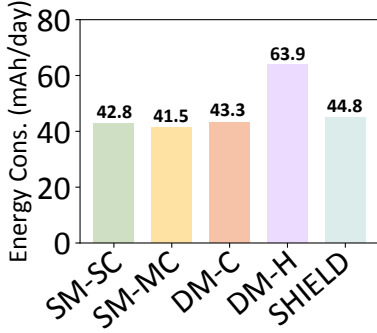


Figure 5.12: Daily energy consumption (mAh/day).

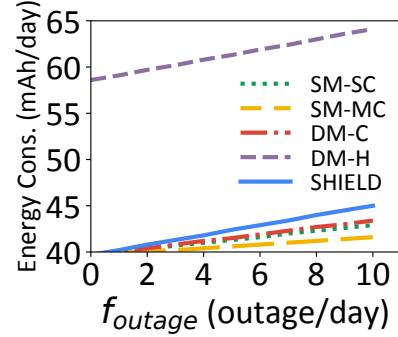


Figure 5.13: Power consumption vs. outage frequencies.

To gauge wireless-conditioned availability, we follow the real-world IoT application setting [Pro23] and quantify the total awake time T_{awake} . The IoT device sends sensory data every 5 minutes, remaining active for 5 seconds for data transfer before going to the sleep mode to save energy. Table 5.2 presents the comparison results for the five solutions across static, walking, and driving scenarios. SHIELD achieves 99.1% availability in driving scenarios, while existing solutions only attain 79.9%-90.7%. Across all three mobility scenarios, SHIELD consistently retains two-nine availability (99.1%-99.7%). In contrast, even under

static scenarios, existing solutions only achieve 91.8%-96.5% availability and fail to achieve two nines.

We also evaluate wireless-conditioned availability under varying outage frequencies, as shown in Figure 5.11. SHIELD ensures two-nine availability with up to 11 outages per day. Under the same outage frequency, existing solutions only achieve 77.6%-89.5%, failing to reach even 90% (one nine). They can only tolerate between 0.4 and 0.9 outages per day, in order to reach two-nine availability. SHIELD offers more than an order of magnitude improvement compared with all prior solutions.

5.5.2 Power Consumption

We evaluate the average daily power consumption for each solution. During the experiments, we employ a power meter to gauge power consumption at various stages, including data transmission, network search, and power saving. We measure power consumption associated with SRX and SIM processing. The daily energy consumption is quantified in the same IoT application setting in §5.2.1, together with traces from the corresponding driving scenario.

Figure 5.12 plots the daily energy consumption for the five solutions. Default SM-SC results in 42.8 mAh/day of energy consumption. In contrast, SM-MC consumes slightly less at 41.5 mAh/day. With its multi-carrier support, SM-MC reduces outage duration, enabling the device to return to the sleep mode more quickly and thereby conserving more energy. DM-C exhibits similar 43.3 mAh/day consumption comparable to SM-SC, as the modem in cold standby awakens only upon outages. On the other hand, DM-H consumes 63.9 mAh/day, 49.3% increase over the SM-SC solution, due to its active backup modem.

SHIELD's energy consumption amounts to 44.8 mAh/day, reflecting modest 4.7% increase in consumed energy³. More frequent outages and longer disruptions not only reduce system availability, but also cause the device to remain active for a longer duration for net-

³SHIELD sets the default threshold T=5 (RSRP<-95 dBm or RSRQ<-15 dB).

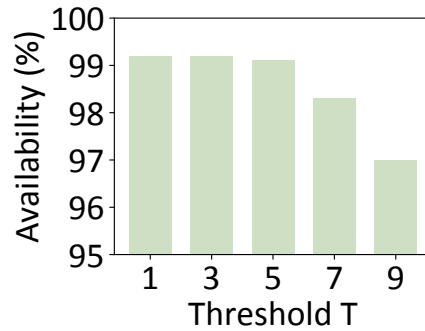
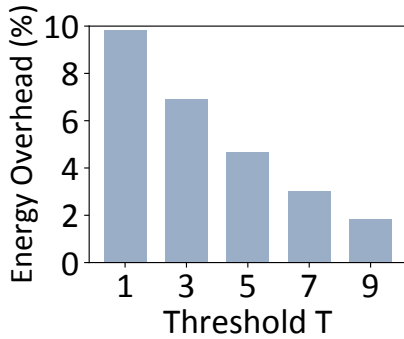


Figure 5.14: Energy overhead vs. threshold T. **Figure 5.15: Availability vs. threshold T.**

work search, thus consuming more energy. Figure 5.13 shows how daily energy consumption evolves with the outage frequency.

5.5.3 Choice of Parameter T

SHIELD offers a customizable parameter T , thus allowing users to configure signal condition thresholds to trigger the outage alert. We study the extra power consumption (Figure 5.14) and availability (Figure 5.15) under various configurations using SHIELD. The energy overhead and availability for different T values are learned. By default, SHIELD sets $T=5$ (i.e., $\text{RSRP} < -95$ dBm or $\text{RSRQ} < -15$ dB), achieving 99.1% availability with 4.7% increase in energy consumption. Traces show that, 96.1% of connectivity outages will trigger the outage alert under this default setting. The SIM’s periodic probing every 5 seconds further detects the remaining 3.9% of failures that occur under strong signals (e.g., 5G core network failures).

The default threshold ($T=5$) balances high availability and energy efficiency in our setting. It can be fine-tuned to meet other IoT requirements. For IoT devices with more constrained battery capacities, a higher threshold, e.g., $T=9$ ($\text{RSRP} < -99$ dBm or $\text{RSRQ} < -19$ dB), can be selected. This option reduces the energy overhead to 1.8% while decreasing availability to 97.0%. A more sensitive threshold improves availability, but increases power

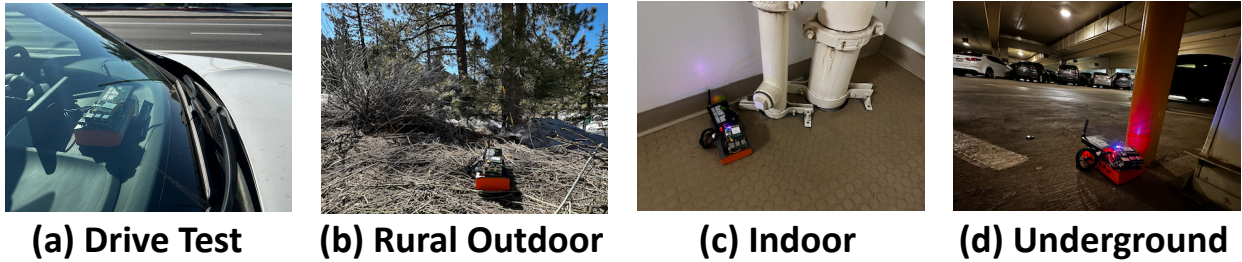


Figure 5.16: Deployments in different scenarios.

consumption due to more frequent discoveries. For example, using $T=1$ ($\text{RSRP} < -91$ dBm or $\text{RSRQ} < -11$ dB) results in 99.2% availability with 9.8% energy overhead. This improvement in availability comes at the cost of increased energy consumption.

5.5.4 Availability on Real-world IoT Applications

We examine the performance of two real-world 5G IoT applications: cold-chain logistics with periodic data transfer, and a gas pipeline fire alarm with event-driven data transfer. The cold-chain app, based on commercial settings [Pro23], sends location and temperature data for 5 seconds every 5 minutes. The gas pipeline IoT app activates the alarm upon fire detection, continuously transmits data until the alarm stops, and reports keep-alive heartbeat messages on an hourly basis. Both apps are developed on Raspberry Pi with sensors, and enter into the sleep mode during inactivity. We compare the multi-carrier (SM-MC) solution and SHIELD in our experiments using two side-by-side IoT devices.

For cold-chain logistics, we performed drive tests in urban and rural areas across the western United States, covering 600 miles of routes. The results indicate that, SHIELD achieves 99.3% availability, outperforming the SM-MC solution with only 89.8% availability. SHIELD reduces unavailability by a factor of 14.6. In terms of energy efficiency, the amortized daily energy consumption for SHIELD is 43.4 mAh, while SM-MC registers at 41.1 mAh, resulting in a marginal increase of 5.5% in extra energy consumption.

We assessed the gas pipeline fire alarm application at five locations: two outdoor rural

areas, one indoor facility, and two underground parking structures. Figure 5.16 shows the deployments in different scenarios. Alarms were triggered randomly throughout the day. Timely IoT data transfer is critical, as a small flame can turn into a major fire in less than 30 seconds [Hom23]. However, the default SM-MC solution failed to ring the alarm within the critical 30-second window in 13.5% of instances. In contrast, SHIELD successfully sends out the alarm within 30 seconds in every case. The daily energy consumption for SHIELD was 5.04 mAh, compared with 4.47 mAh for SM-MC. The primary source for energy overhead came from monitoring during sleep (every 5 hours set for the app). Adaptive monitoring optimization could further reduce this; we plan to explore this option in the future.

5.6 Discussion: The Road to Five Nines

To achieve even higher availability, say, four to five nines to match with the cloud service availability, SHIELD could be extended with a software-defined transceiver with both transmitter (TX) and receiver (RX) capabilities. The slow carrier search and switching on existing modems are the main causes of unavailability. To address it, the transceiver could emulate a mobile device to initialize a session using the concurrent data/control transfer schemes, while the modem is performing the switch. The transceiver uses a backup profile in the SIM card to enable the DoNAS mode. The transceiver continues to piggyback the data with the DoNAS for data transfer, until the modem successfully connects to the new carrier. 5G networks enable the transceiver to access within 6ms [TA19]. Therefore, our calculation shows that, this scheme can achieve 99.999% (five nines) in the static scenario and 99.996% (four nines) in the driving scenario under the measured outage frequencies. The disruption during modem switching could be fully masked.

In this case, although the transceiver needs both RX/TX capabilities, it is still much simpler than a commodity modem. It only requires software support for limited signaling exchanges used in the DoNAS procedures. The flexibility of the transceiver eliminates the

need to complete the entire session setup procedures. Portable transceivers, such as LimeSDR Mini [Mic23c], already support all these required components (RX/TX, clock sync, etc.) to achieve the aforementioned functions.

5.7 Related Work

Multi-carrier switching Current mobile IoT network operators facilitate switching devices between various network carriers [One23, SIM23]. Leveraging multiple carriers has garnered research interests from both the network and device sides. Network-side efforts include sharing infrastructure [PB20, KCY21, SCS16] and radio resources [SKR20, GGD19, QZM20] between carriers for deployment cost reduction. Device-side, multi-carrier efforts focus on enhancing switching performance [Goo23, LDP16, BKS19], improving security [CSB20, OJE20], and resolving policy conflicts [YLL18]. However, achieving rapid multi-carrier switching remains a largely unaddressed issue. Prior methods are reactive and involve exhaustive searches, leading to prolonged disruptions. We introduce SHIELD, a novel SIM-based solution that facilitates rapid multi-carrier switching. Our approach proactively anticipates outages and identifies backup networks in advance, thus making switching faster and more reliable.

High-availability Design There are extensive studies on enhancing cloud system availability, which serves as the back-end for many IoT systems. To ensure high availability, back-end clouds deploy both hardware redundancy and software techniques. Numerous hardware solutions are employed, including backup power supplies [AAB21], RAID storage [KMA22], and hot-swappable components [SHC18], to name a few. However, they do not apply to IoT systems due to intensive processing and energy overhead [MOC14]. The choice of dual modems [Pep23], can result in excessive energy costs. SHIELD addresses this issue by employing a SIM-based software-hardware co-design. Our approach provides high availability with minimal energy overhead. Software techniques include failover mechanisms [KJR21],

data replication [KMA22], load balancing [BTY20], auto-scaling [KLH18], for a few samples. These designs cannot be directly applied to IoT devices due to their complexity. SHIELD introduces a novel data-first design, simplifying the session setup and enabling fast data service recovery.

Device Reliability The IoT devices to date have sufficient hardware capabilities, including the ability to operate over a wide range of temperatures, IP68 waterproofing, and 10-year batteries [EFE23]. These features ensure dependable operations even in harsh environments [MNZ20, OH16]. Prior IoT device reliability research focuses on security protection [WWN22, Has19], firmware verification [SPL19, KKK20], rollback schemes [RCR20], etc. SHIELD is orthogonal to these studies. More recent 5G device reliability research mainly focuses on smartphones [LLL21, ZTX22] and addresses protocol failures without resolving wireless outages. SHIELD facilitates fast outage handling upon wireless outages [YLL18, LLZ20]. Furthermore, the impact of power-saving on availability remains unexplored. To the best of our knowledge, SHIELD is the first to examine the availability impact due to both wireless outage and power-saving induced disruptions for 5G IoT systems.

5.8 Conclusion

Cloud-based 5G IoT systems are on the rise [SKS20] in recent years, for their wide-area coverage, mobility support, and carrier-grade services. Ensuring high system availability is essential to deploying such systems in mission-critical and enterprise application scenarios. Our study reveals that, the 5G IoT network, rather than the cloud and the IoT devices, poses the bottleneck for high availability. In this chapter, we have described our device-side solution, without changing the 5G infrastructure, or device OS and firmware. The key innovation is a SIM-centric software mini-system that exploits the in-device SIM card and a plug-and-play, miniature, software-defined receiver hardware. We rely on pure, device-based operations to enable fast inter-carrier-network switching and data-first handling of mobility

or power-saving induced disruptions. The solution suite is plug-and-play at the device, and works with the 5G infrastructure without any changes. Given the hardware/software configuration choices, we can readily achieve two to three nines (even four to five nines in the immediate future) for 5G IoT, thus eliminating it as the system bottleneck for high availability.

CHAPTER 6

SecureSIM: SIM/eSIM Authentication and Access Control

Ensuring the security of 5G networks is crucial for maintaining high resilience. This dissertation demonstrates that, despite single entities offering security protection, the interactions between the SIM, modem, and 5G/4G infrastructure reveal new vulnerabilities. Attackers can exploit these vulnerabilities to carry out traffic eavesdropping, man-in-the-middle, and impersonation attacks. The root cause lies in the inability of current SIM/eSIM technology to provide fine-grained access control and authentication. To address this issue, we propose a novel certificate-based SIM protection mechanism that enables fine-grained access control and mitigates a wide range of attacks.

This chapter is organized as follows: §6.1 presents the vulnerabilities in current SIM-based 5G/4G authentication schemes. §6.2 elaborates on a novel two-step attack against the existing SIM, which leads to multiple attack vectors. §6.3 introduces our design, SecureSIM, which enables fine-grained access control and ensures proper authentication for SIM/eSIM. The implementation is discussed in §6.4, and the approach is evaluated in §6.5. In §6.6, we examine how new security schemes introduced in cellular networks are unable to prevent attacks and how our solution continues to protect devices in such situations. We present the related work in §6.7 and conclude this chapter in §6.8.

6.1 Vulnerability & Threat Models

6.1.1 Vulnerabilities

The SIM plays a critical role in the 5G/4G connection for authentication and protecting communication. The current SIM design has been prevailing for years due to its simplicity and extendability. Diverse schemes in 5G/4G, such as the fast reattach service, are also designed to improve network performance. However, the interoperability between various entities, such as the SIM, modem, and protocol stacks, exposes vulnerabilities:

First, the PIN-based access control cannot provide enough file protection (**V1**). The SIM cannot differentiate between various entities that access the SIM by PIN verification. Sensitive files, including identities, security-related keys, phonebooks, etc., can be accessed by modems, card readers, OS, or apps. 5G/4G-related files such as the security context are only intended for the modem. Leaking such sensitive files allows attackers to launch eavesdropping, man-in-the-middle (MitM), and impersonation attacks. Furthermore, the current PIN-based authorization is insufficient. Though all the files mentioned above are only accessible after PIN authorization, the SIM only verifies the PIN once, then unlocks access until SIM is unplugged or the device turns off. The attacker without the PIN can still access files during the power cycle, although the user enables the PIN authorization. Even worse, operators usually set the SIM PIN to an identical default number and keep SIM unlocked by default [HP07]. For example, AT&T and Verizon use 1111 and T-Mobile uses 1234 as the default PIN and need users to activate it manually [AT, Ver, T M]. We conducted an informal survey to assess mobile users' sensitivity to SIM security. We collected responses from 103 volunteers from the authors' social networks in the US and 86 responses from anonymous online participants, which were solicited from a US university's public forums on online social networks. The participation was open to everybody and there was no material incentive. The participants cover a wide range of age groups, as shown in Figure 6.1. Among participants, 94.2% of the users set the password for their phones, but only 10.6% set the

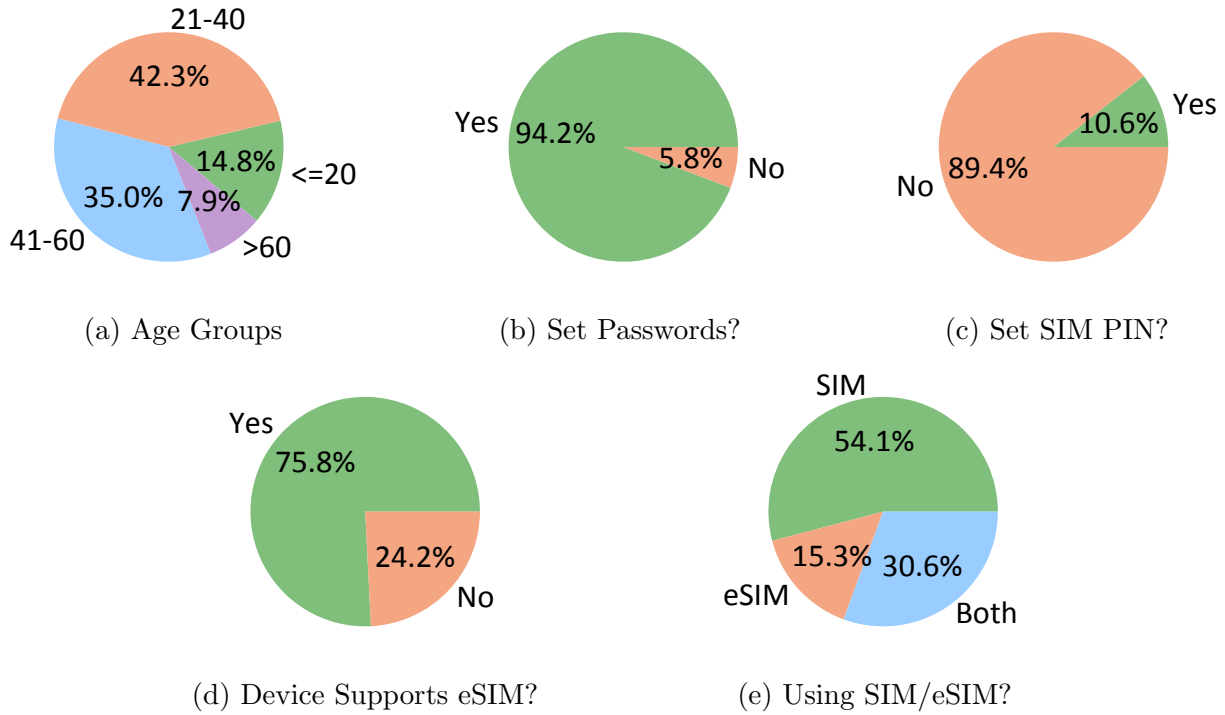


Figure 6.1: SIM/eSIM usage survey.

SIM PIN. Our survey results indicate that the vast majority of mobile users do not enable the SIM PIN, exposing all protected files to attackers.

Second, the SIM and 5G/4G protocol stack interoperation exposes vulnerabilities (**V2**). Storing the security context is mandatory, but the current design cannot provide enough protection. According to the standard, if the SIM supports the fast reattach service, the modem shall store the NAS security context into the SIM when the UE detaches from the network [3GP20b]. The modem would delete the context when it detects the SIM is changed during the power-on state. During the power-off state, however, the modem cannot detect if the SIM has been changed. The attacker can unplug the SIM and acquire the context. Supposing that the attacker inserts the SIM card back when the modem’s power is off, the UE and network will use the same context to perform the fast reattach, which allows various attacks, including traffic eavesdropping, man-in-the-middle (MitM), and impersonation.

Third, the channel between the modem and the SIM is unencrypted (**V3**). It enables

malicious hardware such as a SIM sticker [SIMa] to eavesdrop on and spoof the APDU traffic. The attacker can acquire all the files as plaintext when the modem reads them from the SIM. Multi-layer keys in 5G/4G can also be accessed through the unencrypted APDU. Prior work leveraged CK/IK leakage during the AKA procedure to attack the VoLTE and VoWiFi [CSP17]. We discover that the K_{AMF} leaks when the modem stores the NAS security context into the SIM. Moreover, the attacker hardware can modify the APDU or actively send APDUs to the mobile device to acquire the device location, network conditions, device ID, calling information, etc. The attacker can also leverage APDUs to set up calls or send messages without notifying the victim.

6.1.2 Threat Model

We consider two types of attackers. A *passive attacker* can set up the standard-compliant SDR devices to eavesdrop the radio layer information. The passive attacker could log the ciphered signaling and data messages between the victim UE and gNB/eNB, which remains unperceivable. An *active attacker* can set up forged UE, gNB/eNB, and 5GC/EPC to send standard-compliant signaling and data messages in the communication channel. The active attacker could also forge or alter packets from/to the victim at the forged gNB/eNB and 5GC/EPC. Both types of attackers leverage low-cost SDR devices. We assume both attackers can acquire sensitive files such as the NAS security context from SIM with one-time physical access through customized hardware (such as the SIM sticker or card reader in §6.2.1) or malware installed on the victim’s mobile devices.

Our experiments also observe ethical rules. We ensure that tests on commercial networks do not harm other users. We own the SIM cards used in the experiments. Our passive attacker does not send signals over the licensed band. We use the unlicensed bands for experiments on the MitM and impersonation attacks. We are also informing mobile operators of our findings.

Attack Method	Physical access?	Device rooted?	Need phone password?	Need SIM PIN?	Effective to SIM?	Effective to eSIM?
SIM Sticker	Yes	No	No	No	Yes	No
Card Reader	Yes	No	No	Yes	Yes	No
Malware	No	Yes	No	No	Yes	Yes

Table 6.1: Requirements of hardware and software attacks.

6.2 Attacks

We next devise a novel two-step attack against the current SIM. First, we introduce a component attack that obtains sensitive files, such as the security context, from the SIM. We show that the attacker only needs one-time physical access to acquire sensitive files, which opens a door for other attacks. Second, based on the one-time access attack, we devise three novel attacks: traffic eavesdropping, context-based MitM, and impersonation attack. We present the attack procedures for each attack and validate them on two US mobile operators, denoted as OP-I and OP-II.

6.2.1 One-time Access Attack

Current SIM with PIN-based access control cannot differentiate between access entities. Attackers could leverage cheap hardware or malware to acquire sensitive files, such as the victim’s identities and the security context. The prerequisite for our attacks is to obtain the security keys via one-time physical access to the SIM card or malware on the rooted device. Table 6.1 shows requirements for attacks from different hardware and software. SIM is susceptible to both hardware/software attacks, and eSIM is only susceptible to software attacks.

Attackers can leverage the customized hardware, such as a SIM sticker, to extract sensitive files from the victim’s SIM. The SIM sticker, as shown in Figure 6.2, is a small chip with a processor and storage. The SIM sticker has the same size as a SIM card and can be attached to the SIM. It is used to support multi-IMSI or unlock the carrier lock [Kno, Hei].



Figure 6.2: SIM sticker.

The attacker can extend it to log the sensitive files. To attack the victim with the SIM sticker, the attacker first needs one-time access to the victim's device. The attacker unplugs the SIM, attaches the sticker to the SIM card, and inserts them back into the device. The attack through the sticker does not require the device password or SIM PIN. APDUs between the modem and SIM will go through the sticker. Even if the victim enables the PIN, the sticker could still eavesdrop on all the APDUs as plaintext (**V3**). The attacker does not need to reaccess the device or unplug the sticker to read data. The sticker can forge APDUs and send the captured files through SMS to the attacker's number, enabling the attacker to control the APDU channel and acquire SIM files remotely.

The attacker can also use a SIM card reader to obtain sensitive files. Reading files with the card reader does not require the device password but requires that the SIM PIN is not enabled or the attacker knows the PIN. The attacker needs one-time access to the victim's device first. To acquire the security context files, the attacker triggers the UE detach procedure by enabling airplane mode from the status bar or turning off the device by pressing the power button. It forces the modem to store the security context into the SIM (**V2**). This procedure usually does not require unlocking the phone. The attacker unplugs the SIM, reads files with a card reader (**V1**), then plugs the SIM back into the victim's device.

To achieve the same purpose, the attacker can allure victims to install a malicious ap-

plication on the phone or piggyback malicious codes in SDKs used by mobile applications [LLB17]. Attackers extract sensitive files through system APIs (**V1**) and acquire Identities, phonebooks, SMS if the user grants privileges to the malware. It is feasible since most users cannot make security decisions correctly based on the prompt window's limited information [FHE12, RFW19]. With root privilege, the attacker can further access all files with the PIN access mode by sending APDUs to the SIM or eSIM. The malware can send acquired files from various channels, including Bluetooth, Wi-Fi, 5G/4G, SMS, to name a few.

The various approaches above target different scenarios. The malware, which can bypass the PIN authentication on rooted devices, is able to attack both the SIM and eSIM. For unrooted devices, the attacker could launch the hardware attack to the SIM with one-time access. In addition to smartphones, cellular IoT devices using SIM cards are also vulnerable to such a one-time access attack. By default, the SIM PIN is disabled. The attacker can directly use a card reader to extract sensitive files. Even though the victim enables the PIN, the attacker can still insert the SIM sticker along with the SIM card into the device to capture all APDUs, including the PIN, as plaintexts. After getting the context with the one-time access attack, all the following attacks, including traffic eavesdropping, context-based MitM, and impersonation attacks, are feasible for both SIM and eSIM.

Experimental Validation We validate the feasibility of the one-time access attack with hardware and software. With a card reader, we implement a program with Python that can automatically extract all the files readable with the PIN, including NAS security context, identities, and phonebooks, to name a few. Current commercial SIM stickers do not provide open-source SDK. Thus, we use the SIMTrace [Osm23] board to emulate the SIM sticker's behavior. Results show that it can eavesdrop on all the APDU communication to extract keys, identities, location updates, etc., and further send them out through SMS. We implement the software attacker as an Android application that acquires SIM files on the rooted devices with system APDU interfaces. We test it on Google Pixel, XIAOMI MIX2, and Google Pixel 4a. As shown in Figure 6.3, we can successfully send APDUs to the SIM and

```

11:39:30.128: SEND:00A4000402[6F07]Select IMSI File (6F07)
11:39:30.147: RECV:9000 Success (9000)
11:39:30.149: SEND:00B00000[09]Read IMSI; Length 9B
11:39:30.165: RECV:08 IMSI File Content 9000
11:39:30.167: SEND:00A4000402[6FE4]Select EPSNSC (6FE4)
11:39:30.183: RECV:9000
11:39:30.186: SEND:00B20104[36]Read EPSNSC; Length 54B
11:39:30.202: RECV:

```

EPSNCS File Content 9000

Figure 6.3: Malware attack APDU traces.

acquire sensitive files. We compare performances when acquiring files with a card reader for SIM and with software for both SIM and eSIM. Regarding the average time to read a file, it takes 10.5 ms for the SIM reading with a card reader. The software leveraging the Android API takes a similar time for SIM and eSIM, which are 11.3 ms (SIM) and 10.8 ms (eSIM).

We also conduct a survey to evaluate the SIM/eSIM usage and user security sensitivity. Our survey results from 189 responses show that 75.8% of participants' phones support eSIM, as shown in Figure 6.1. Among all participants, 54.1% only use the SIM, 15.3% only use the eSIM, and 30.6% use both SIM and eSIM for their devices. For the hardware attack, the SIM sticker can only be discovered when the user plugs out the SIM, e.g., by switching the SIM cards for the current phone. However, it is an infrequent behavior for daily usage. 75.6% users plug out the SIM card less than once a year, which makes the sticker attack potentially last for a long time once the one-time access is successful. Also, 53.4% of participants admit that there are chances for other people to access the phone physically in offices, homes, etc., making them vulnerable to the one-time access attack.

6.2.2 Traffic Eavesdropping

Given the one-time access attack exploiting the interoperation vulnerability (**V2**), the adversary can eavesdrop and decipher victim's data and signaling messages. The attack includes

two stages: the online trace collection stage and offline decoding stage.

At the online trace collection stage, the attacker camps on the victim's connected cell and eavesdrops on the victim's traffic. The attacker can infer the victim's connected cell through monitoring the paging channel [SBA15]. With the inferred cell ID, the attacker camps on the cell based on broadcasted configurations [3GP19]. To locate the victim's traffic in the channel, the attacker needs to know the victim's Cell Radio Network Temporary Identifier (C-RNTI), which can be acquired by monitoring the unciphered signaling messages [KRH19]. Then the attacker could log the ciphered signaling/data traces. At the same time, the attacker collects the bearer configurations for later decryption. Afterward, the attacker performs the one-time access attack. By acquiring the file EPSNSC storing the NAS security context, the attacker can get the K_{AMF} , uplink/downlink NAS counter, and NAS integrity/encryption algorithm identifiers.

At the offline decoding stage, the attacker leverages the NAS security context and logged signaling and data bearer configurations to decrypt the traces. The attacker derives K_{NASenc} , K_{NASint} , and K_{gNB} from the NAS security context. Note the decryption also needs to infer algorithm identifiers for RRC and data decryption. The identifiers are static for the cell, and attackers can infer them by another mobile with software such as MobileInsight [LPY16]. In the worst case, there are three candidate algorithms [3GP13] and the attacker can brute-force them to derive keys for RRC signaling and user data. Combing the keys, the frame counter in unciphered frame headers, and the bearer configurations from the online stage, all the traces eavesdropped can be deciphered offline.

Furthermore, the K_{AMF} can last for a long time, enabling the attack not only before the one-time access but also for later traffic. The K_{AMF} will only change when a new AKA procedure happens, or the 32-bit counter wraps around, which rarely happens due to the low volume of NAS messages. When the UE handover is due to mobility, only the K_{gNB} will change, and the K_{AMF} remain the same. In the fast reattachment procedure, operators decide whether to perform the AKA procedure. If the operator does not run the AKA procedure,

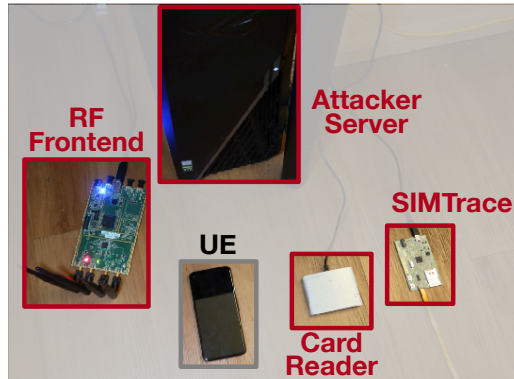


Figure 6.4: Attacker testbed setup.

the network and UE will keep using the same K_{AMF} even the phone restarts.

Experimental Validation We perform the validation on the commercial network and phones. We implement the attacker node upon the open-source srsLTE stack [GGS16] on an Ubuntu 18.04 server with i7-9700K CPU and use USRP B210 as the RF frontend, as shown in Figure 6.4. We deploy the attacker node with OP-I’s configurations. As shown in Figure 6.5, it contains two components: the online tracer and the offline decoder. First, the online tracer monitors the signaling and data frames at the physical layer and logs the ciphered messages as MAC traces. It also collects the bearer configurations for the decryption. Then we perform the one-time access attack. We read the NAS security context file `EPSNSC` from the SIM. The offline decoder automatically parses the file to generate K_{NASenc} , K_{RRCenc} , and K_{UPenc} . It performs the RLC and PDCP decoding and decrypts the UE’s traffic. Experiments show the attacker node can successfully eavesdrop on the victim UE connecting with OP-I network. Figure 6.6 shows the eavesdropped traffic before and after the decryption. The OP-II in the experimental area uses a different modulation and coding scheme in the physical layer, which the current open-source 5G/4G testbed cannot support. We perform a trace analysis and a similar attack procedure can also work with OP-II theoretically.

We perform a user study of the K_{AMF} ’s lifetime on OP-I and OP-II. During the testing, devices follow daily usage scenarios with various mobilities, including static, walking, and driving. For both operators, the same K_{AMF} can last for more than one week. Experiments

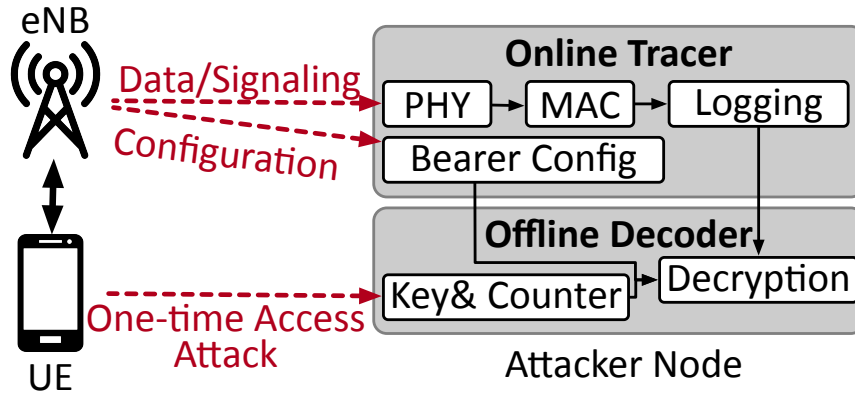


Figure 6.5: Eavesdropping implementation.

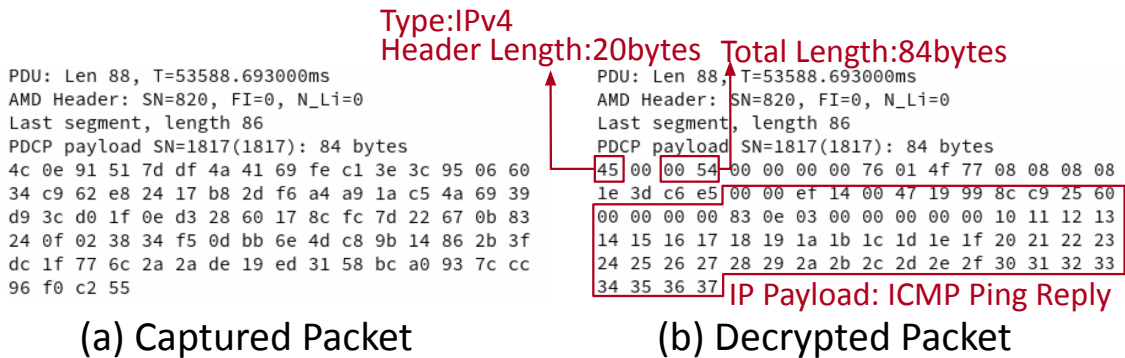


Figure 6.6: Eavesdropped traffic decryption.

show that OP-II does not perform the AKA procedure when the UE reattaches to the gNB/eNB. The UE keeps using the same K_{AMF} after the restart or turning on/off the airplane mode. The attacker can acquire all the traffic without end-to-end encryption, such as DNS, HTTP, or IoT sensory data [IoT].

6.2.3 MitM Attack

The attacker can leverage the SIM vulnerabilities and exploit the fast attachment procedure to perform the MitM attack. Prior work exploited the missing integrity protection of the user plane on the physical layer to introduce bit flips and redirect traffic to another destination [RKH19]. The fake base station (FBS) performs diverse attacks [YBS19, Str07, SBA15],

including denial of service, network downgrade, UE identities, location tracking attacks, etc. They all use a relay node between the victim UE and commercial gNB/eNB, which cannot perform decryption. We propose the context-based MitM, the first attack that can perform the decryption for both the control and data plane and further manipulate signaling and user data. We next introduce our context-based MitM attack.

The attack requires the NAS security context and GUTI, which can be acquired from EPSNSC and EPSLOCI by the one-time access attack. When the victim UE tries to reattach, it scans all supported RF bands, filters out cells of its operator’s Public Land Mobile Network (PLMN), and selects the cell with the strongest signal to attach. Simultaneously, the attacker sets up a fake gNB/eNB and 5GC/EPC with the same PLMN as the victim’s operator and loads the context into it. The attacker node increases the transmit power to be the strongest among all victim candidate cells. The UE will send the Attach Request with GUTI to the attacker node to perform the fast reattachment.

After receiving the Attach Request, the attacker node returns the NAS Secure Mode Command (SMC) with the corresponding 4-byte NAS-MAC derived from the stolen security context. After the UE verifies the NAS SMC’s integrity, it connects the Internet with the fake gNB/eNB and 5GC/EPC. All the data packets are forwarded to the Internet by the P-GW in the fake 5GC/EPC. Attackers can eavesdrop on all the traffic and perform heterogeneous MitM attacks such as IP/ARP/DNS spoofing, SSL hijacking, and SSL stripping, to name a few. It risks the victim’s email, bank, or social media accounts and further causes financial losses [CYT18, JG17]. The attacker can also derive the integrity key so that previous solutions, e.g., adding integrity protection for the 5G/4G data plane, cannot prevent this attack [RKH19, RKH20].

Experimental Validation We implement the attacker node as shown in Figure 6.7 upon the srsLTE, which includes a fake gNB/eNB and 5GC/EPC, the dynamic context loader, and the traffic MitM module. We validate the attack on OP-I and OP-II with commodity phones, including Google Pixel 4a, XIAOMI MIX2, and iPhone 7 as the victim UE. When

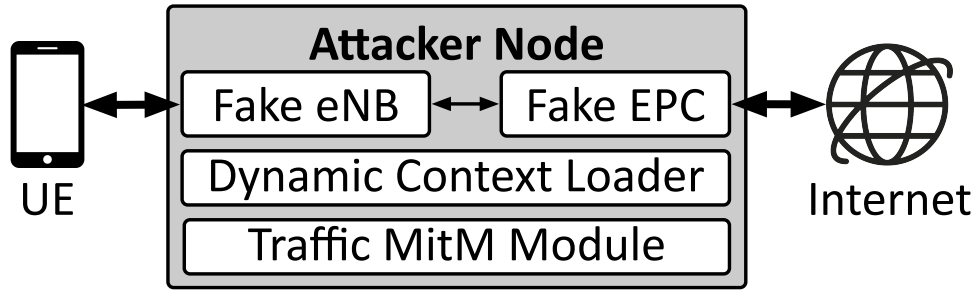


Figure 6.7: Context-based MitM.

performing the one-time access attack, the attacker sets the victim to airplane mode and configures the attacker node with the same PLMN as the victim’s operator on an unlicensed spectrum, following the ethical guidelines. We ensured that experiments do not harm real-world users.

The dynamic context loader loads the acquired EPS_{LOCI} and EPS_{NSC} and updates the attacker node’s cached context. It will also derive the $K_{NAS_{enc}}$ and $K_{NAS_{int}}$. After receiving the Attach Request from the victim UE, the attacker node will generate the NAS SMC integrity protected by the $K_{NAS_{int}}$. Experiments show that the victim validates the NAS SMC’s integrity, and the attacker node passes the UE authentication. The victim completes the following attach procedure and transmits all the data and signaling through the attacker node. With the USRP B210 as the front end, the attacker node can maintain the highest signal strength among all the candidate cells within 10 meters of the victim. We believe the distance can be larger with more powerful RF devices.

The traffic MitM module can perform heterogenous MitM attacks. Results show the victim UE for both OP-I and OP-II will connect to the attacker node. We build the show-case MitM attack, including DNS spoofing and SSL stripping. Leveraging the DNS spoofing, the attacker could forge the DNS packets to make the victim connect to phishing websites. As shown in Figure 6.8(a), the victim will enter the phishing bank website with the correct domain name. The attacker could set up phishing page layouts identical to the real login page, which makes the account information vulnerable. Even for websites that enable

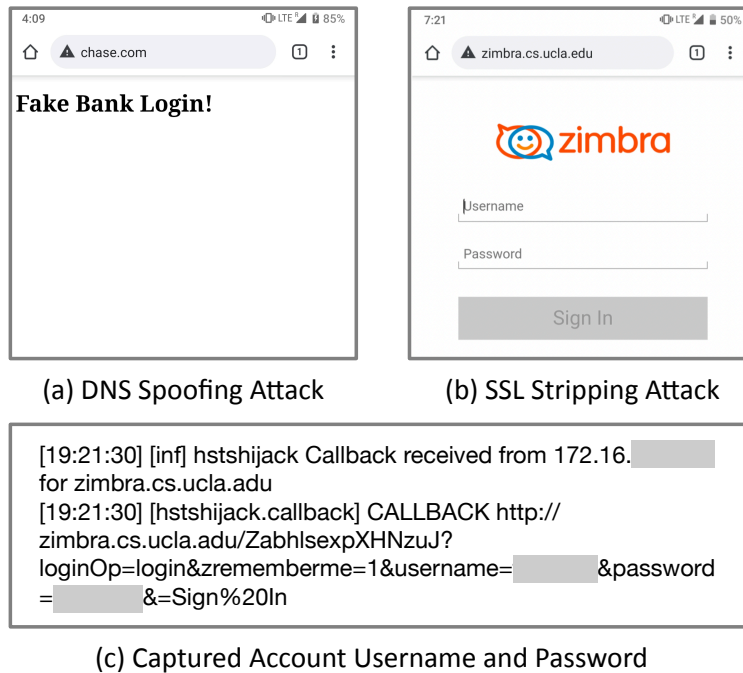


Figure 6.8: MitM attack validation.

HTTPS, it is still feasible to attack the user session with SSL stripping. The SSL stripping could downgrade the connection from HTTPS to HTTP to be able to perform the MitM attack [HPI18]. Figure 6.8(b) shows that the HTTPS connection to the email webpage is downgraded to HTTP. The attacker node could capture the username and password from the victim's request with an HTTP proxy, as shown in Figure 6.8(c).

6.2.4 Impersonation

The attacker can further impersonate the victim's identity by leveraging the vulnerabilities in SIM and fast reattachment service. A recent study reveals that the lack of physical layer integrity protection along with ICMP reflection mechanisms allows the impersonation attack [RKH20]. The prior attack is limited to the user plane and requires the attacker's relay node to be close to the victim. We leverage the SIM vulnerabilities with fast reattachment to impersonate the victim on both the control and user plane. This attack can be launched remotely during the impersonation, thus making it more challenging to be detected.

The impersonation attack includes three steps. First, the attacker acquires the fast reattachment context files (EPSLOCI and EPSNSC) from the SIM by the one-time access attack. Second, the attacker derives the K_{NASint} and K_{NASenc} with K_{AMF} and integrity algorithm identifier from the EPSNSC . The UE fast reattachment requires the victim’s NAS keys, GUTI, and network capability. The network capability of a device is public information. Finally, the attacker sets up a fake UE with SDR devices to send the NAS Attach Request with GUTI from the EPSLOCI , which is integrity protected by the NAS-MAC derived from K_{NASint} . After the AMF/MME authenticates the fake UE and grants it access to the Internet, the attacker can drain the victim’s data plan or attack other entities with the victim’s identity. Some operators verify device access to applications or services through the cellular identity/keys [RKH20]. It protects users when the application or SMS one-time passwords are leaked. However, the SIM impersonation attack could still bypass the checking and cause financial losses. Note that the attacker with the context can launch attacks far from the victim, which makes locating attackers challenging for law enforcement agencies.

Experimental Validation We perform the validation on the SDR testbed with the same hardware configurations before. With one USRP B210 as the gNB/eNB, we use commodity phones with programable SIM cards that support fast reattach service to connect to our private 5G/4G network. We test Google Pixel, Google Pixel 4a, XIAOMI MIX2, and iPhone 7. Through the one-time access, the attacker acquires the EPSLOCI and EPSNSC from the SIM. Another USRP B210 runs srsUE as the rogue UE. The rogue UE loads the context, derives K_{NASint} and K_{NASenc} , and sends NAS Attach Request with the valid GUTI, UE network capability, and NAS-MAC derived from K_{NASint} . The results show that the rogue UE will be authenticated as the victim’s identity and set up the 5G/4G connection. Figure 6.9 shows corresponding logs at the gNB/eNB. We also perform trace analysis on the commercial network successfully. Traces show that OP-II does not run a new AKA procedure when the UE performs the fast reattach, allowing attackers to impersonate the victim’s identity.

```

19:14:29.730205 [S1AP] [I] Received Initial UE message -- Attach Request
19:14:29.730211 [NAS ] [I] Attach request -- M-TMSI: 0xd9bc07a GUTI attach
19:14:29.730213 [NAS ] [I] Attach request -- eNB-UE S1AP Id: 5 request with
19:14:29.730214 [NAS ] [I] Attach request -- Attach type: 2 victim's context
...
19:14:29.730231 [NAS ] [I] Attach Request -- Found previously attached UE.
19:14:29.730238 [NAS ] [I] Integrity check ok. Local: count=6, Received: count=6
19:14:29.730241 [NAS ] [I] GUTI Attach -- NAS Integrity OK. UL count 6, DL count 3
19:14:29.730247 [NAS ] [I] Generating KeNB with UL NAS COUNT: 6
...

```

↓
Successfully attached as the victim

Figure 6.9: Impersonation validation.

6.3 SecureSIM Design

We design SecureSIM to enable flexible access control and ensure proper authentication at SIM. Our solution seeks to address two issues:

- **How to enable fine-grained file access control for heterogeneous entities?** The current SIM includes about 200 files, and different entities have various access requirements for these files. We devise a novel access graph to organize the SIM files. Moreover, we examine common scenarios that require SIM access, including all off-card units and in-card applets. We consequently design a multi-rooted tree in the access graph to offer fine-grained access control to various entities. To improve usability, we design APIs for the access graph, which provides a flexible way for operators to manage file access. Operators specify the intended access policies for various entities. SecureSIM detects conflicts and generates the access graph to perform its fine-grained access control.
- **How does the SIM distinguish various entities and ensure security?** Fine-grained access control itself cannot prevent attackers from acquiring files. Attackers may pretend to be another entity for file access. Proper authentication is thus needed (V1). It should defend against authentication attacks from the card reader, SIM sticker, malware, physical thefts, etc. SecureSIM devises the certificate-based solution to ensure authentication, flexibility, and scalability. We design the certificate chain for SecureSIM to distinguish

various entities. The certificate ID binding protects the SIM from physical theft. To protect the APDU communication channel (**V3**), we propose selective encryption to ensure data secrecy with negligible overhead. With the authentication and APDU channel protection, SecureSIM protects SIM files such as identities, security context, etc., to prevent traffic eavesdropping, MitM, and impersonation attacks (**V2**). Most users do not enable the current PIN-based protection. SecureSIM makes an effort to be more user-friendly. We thus co-design automated verification and revocation with the modem to protect users without degrading usability.

6.3.1 Control SIM Access with Graph

To address the first issue, we leverage two types of relations in our design. The relation among files enables a flexible approach to managing these SIM files. The relation between entities facilitates the access to be fine-grained for various scenarios. We consequently devise a novel access graph to model the relations in the SIM access control.

6.3.1.1 Access Graph

The access graph, as shown in Figure 6.10, has three types of nodes: file node, profile node, and schema node. Each file node denotes a single file stored in the current SIM. Each profile node denotes a category of files. Each schema node contains the policy blueprint for a given entity.

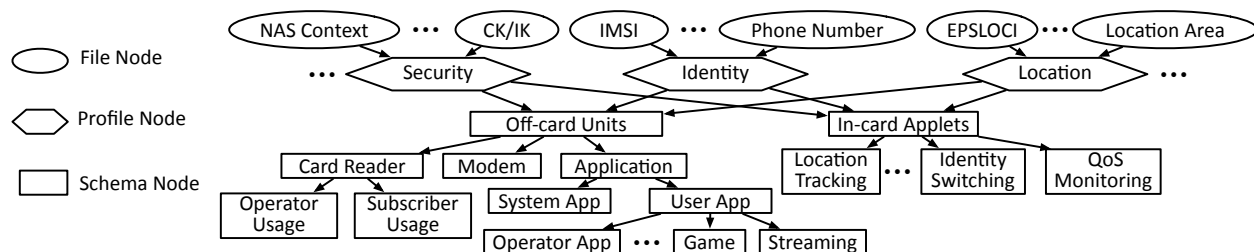


Figure 6.10: Managing SIM file access for off-card units and in-card applets with access graph.

Profile Type	Included Files
Identity	IMSI (6F07), Phone number (6F40), etc.
Security	NAS Security Context (6FE4), CK/IK (6F08), etc.
Location	EPS location Info (6FE3), Location Area (6F7E), etc.
User Information	Phonebook (5F3A), SMS (6F3C), etc.
Operator Information	Operator Name (6F46), Icon (6FDE), etc.
Network Configurations	Services Table (6F56), Access Technology (6F60), etc.

Table 6.2: SIM profile node types.

We design the profile node to organize files into a small number of categories for systematic access management. With more services added to the SIM, the first issue is how to manage diverse SIM files. Different from servers or smartphones, which have enough processing capabilities and storage to manage a large number of files, SIM’s processing speed and storage are limited. It is burdensome to manage each file’s access control for more than 200 files in the SIM, which requires an access control design that fits the hardware capabilities. To better manage files with the same requirements, we examine current SIM files, and classify them into six categories (shown in Table 6.2). Each category is with a profile node. In the access graph, each file node connects to only one profile node. Every profile node contains multiple files sharing the same access control permissions.

We design the schema node to classify heterogeneous entities. Organizing various entities systematically and flexibly is challenging. By reviewing current usage scenarios, we classify entities into off-card units and in-card applets, which are organized as a multi-rooted tree. The schema node provides a convenient and intuitive way to construct the tree based on practical scenarios. All profile nodes connect to the tree to check entities’ privilege. Each schema node contains the node name and access privileges. We prescribe the tree structure with access schema in Table 6.3. Symbol / splits schema nodes into multiple layers. Symbol [] denotes extensible schema nodes. For example, the UE category indicates the modem’s capabilities. A schema node Modem links [UE Category] schema nodes, enabling operators to customize the access control for specific devices such as NB-IoT. The applications contain system applications and user-installed applications, which are further classified based on

Type	Class	Access Schema
Off-card	Modem	/Offcard/Modem/[UE Category]/[Modem ID]
	System App	/Offcard/App/SystemApp/[App ID]
	User App	/Offcard/App/UserApp/[App Type]/[App ID]
	Operator	/Offcard/Reader/Operator/[Operator ID]
	Subscriber	/Offcard/Reader/Subscriber/[Subscriber ID]
In-card	Services	/Incard/[Service Type]/[Applet ID]

Table 6.3: Access schema.

application types. The SIM grants access to applets based on the service type. The finest granularity for all entities is the unique ID, such as the application ID [And, App], the modem ID [Qua], and the applet ID [ISO04]. The schema nodes contain each profile node's privileges, including *Read (R)*, *Update (U)*, *Delete (D)*, etc., which complies with the current standard [ISO13].

The graph's characteristics benefit access control management. Schema nodes follow the least privilege principle: an entity is given the minimum levels of access it needed. Fine-grained entities should have higher or equal privileges than coarse-grained entities [FSG01, AZH12]. The more fine-grained child node can possess higher privileges than its parent. For example, the system application can *Read* identities, but a general application cannot. Each schema node inherits its parent privilege by default. When a schema node's privilege changes, all children's default privileges also change accordingly. When an entity accesses files, SecureSIM first checks the file's profile nodes, then follows schema nodes and grants access based on the corresponding leaf node's privilege.

The access graph is flexible for future extensions. When adding new files, the access graph can link each file to one of the current profile node. If new files cannot fit current profiles, the access graph supports extending new profile nodes for them. The access graph supports heterogeneous scenarios by extensible schema nodes. In rare cases, If the SIM wants to further control the access for a specific file, it can add a new profile node, change the file's profile node to the new one, and set access privileges in schema nodes.

Access Policy	ID	Sec	Loc	UserInfo	OpInfo	NetConf
/Offcard/Modem/Cat(1,3)/*	R	R,U	R,U	R,U	R	R,U
/Offcard/App/SystemApp/*	R	/	R	R,U	R	/
/Offcard/App/UserApp/ Operator/{ID1,ID2}	R	/	R	R	R,U	/
/Offcard/App/UserApp/*	/	/	/	/	/	/
/Offcard/Reader/Operator/OP-I	All ¹	All	All	All	All	All
/Offcard/Reader/Subscriber/0001	R	R	R	R,U	R	R
/Incard/Identity/*	R,U	/	/	/	/	/

Table 6.4: Access policy examples.

6.3.1.2 Access Policy

The access graph provides fine-grained SIM access control. However, directly setting up the privilege for every schema node is error-prone. Privileges may violate the least privilege principle. Duplicate sibling nodes with conflicting privileges may also exist. We provide the access policy as APIs for operators to grant privileges.

Operators need a flexible way to grant the same privilege to many entities. For example, the SIM may grant *Read* identity privilege to all system applications without specifying the application ID. The access policy uses a notation similar to the regular expression. The *** matches any name. *{...}* to represent a list of names. *(a,b)* represents a range from *a* to *b*. Table 6.4 shows an example access policy for SIM files. Note that SecureSIM forbids all applications to acquire the security files to prevent software attackers. The modem plays a critical role in network connection and could read or update most of the files. A wildcard covers all system applications without specifying the ID. The operator can restrict SIM usage to only a range of UE categories. A list of operator applications can read the operator information for add-on services.

The operator grants privileges by access policies with the following steps: First, the operator specifies policies for each profile type; Second, SecureSIM checks policy conflicts.

¹All grant privileges for all operations including Read, Update, Append, Delete, Create, Activate and Deactivate.

The conflicting access control could lead to several damages. A higher privilege to an entity leaks keys or privacy, and an insufficient privilege blocks normal functions. SecureSIM leverages the graph's characteristics to assist the automated privilege checking. SecureSIM checks privileges with the least privilege principle for the parent and children nodes, detects duplicate or conflicting privileges for sibling nodes, and verifies the tree structure correctness with the access schema. Upon detecting conflicts, SecureSIM notifies the operator to fix till no conflict exists; Finally, SecureSIM generates the access graph on the SIM with access policies. With the access graph, SecureSIM controls various entities' access as described in Section 6.3.1.1.

6.3.2 Rethink SIM with Certificates

The second question for the SIM design is: **How does the SIM distinguish various entities and ensure security?**

To identify various entities, the SIM requires authentication before the entity access files or performs functions such as AKA procedure. The current SIM takes password-based authentication for years, and the eSIM follows the same approach. The PIN verification is easy to deploy considering the SIM's limited storage and computation capability. SIM only allows limited tries (usually three times), making it robust to brute force and side-channel attacks. However, the current PIN only contains 4-8 digits and lacks enough strength. Furthermore, simply extending the PIN mode to multiple modes cannot distinguish various entities. It is not flexible for heterogeneous entities.

Another solution is to improve the strength with a shared key between the SIM and each entity. Nevertheless, the shared-key authentication is not scalable. Entities must own different keys to be distinguished. The SIM needs to store all keys and corresponding entity names for the fine-grained access control. Although the SIM with 50K free memory could hold more than one hundred keys, managing the keys is still challenging. Whenever the operator adds a new entity, all SIM cards need to update keys remotely. The cumber-

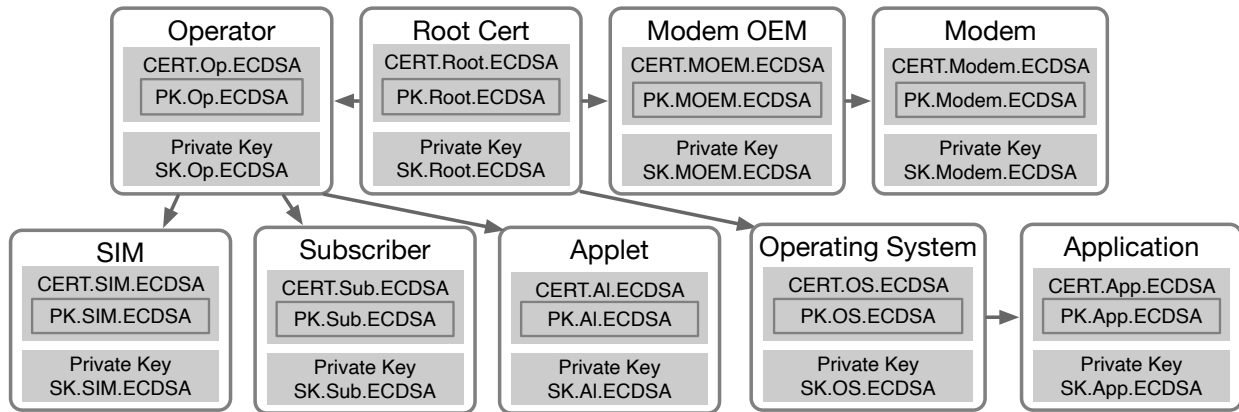


Figure 6.11: SIM certificate chain.

some distribution procedure limits the scalability and cannot perform fine-grained identity checking and access control.

SecureSIM takes the certificate-based authentication, which is much more scalable than the password or shared key authentication. The SIM does not have to store keys for each entity but only a single certificate or several certificates. The certificate is flexible for heterogeneous entities. We design the certificate chain for SecureSIM with ID binding to ensure security and flexibility. To protect the plaintext APDU channel, we design selective encryption for SIM-modem communication with negligible overheads. Considering SIM’s limited processing capability, performing all tasks on the SIM will affect the user experience. We leverage the modem to accelerate SecureSIM. We devise the automated certificate verification and revocation procedure to improve security without hindering usability. SecureSIM design cannot be directly applied to existing phones and require operators and modem manufacturers to update the SIM applets and modem firmware. All functions can be updated over the air and do not need extra hardware. We discuss how SecureSIM could leverage the current infrastructure for each scheme to be deployed gradually. Now we introduce the details of certificate-based SecureSIM design.

6.3.2.1 Certificate Chain with ID Binding

We design the certificate chain as shown in Figure 6.11 for fine-grained access control (**V1**). In the certificate chain, the root certificate is self-signed and organized by a trustworthy third party such as GSMA [GSM]. Operators collaborate with SIM vendors to manufacture SIM cards and sign certificates for SIMs, subscribers, and applets. Independent of operators, the modem original equipment manufacturer (OEM) manufactures modems according to 3GPP standards. The root certificate signs the modem OEM certificates, and each OEM signs the modem. The current eSIM system also leverages the certificates to validate operators' SIM packages and eSIM chips. GSMA defines specific Public Key Infrastructure (PKI) for eSIM remote provisioning, and signs certificates to operators and OEMs [GSM20]. The current eSIM infrastructure can be extended to support the SecureSIM certificate chain. The root certificate also signs certificates for operating systems, which further sign applications. Current operating systems, such as Android, require the application to be signed before installation. For example, when the developer publishes the application to the google play store, Google provides the platform to sign and publish the application [Goo]. SecureSIM could leverage the platform pipeline to sign applications automatically.

The certificates include a customized attribute *SchemaName* that follows the access schema for access control. SecureSIM leverages the *ModemID* in the *SchemaName* to prevent physical thefts. Each modem possesses a unique modem ID. The user needs to send the modem ID to the operator when requesting the SIM for the first time. The operator sets the ID into a whitelist in the SIM. Only whitelisted modems can pass the authentication. If the user wants to change the device, he can add more modem IDs to the list through authenticated devices or request the operator. For IoT devices, the SIM can store a modem ID pattern as the access policy to whitelist a collection of modems. Through the Modem ID binding, the SIM can prevent physical thefts from stealing the data plan or the identity. In current eSIM deployments, operators also require users to submit the eSIM chip ID before activating the eSIM usage. The modem ID could also be submitted simultaneously.

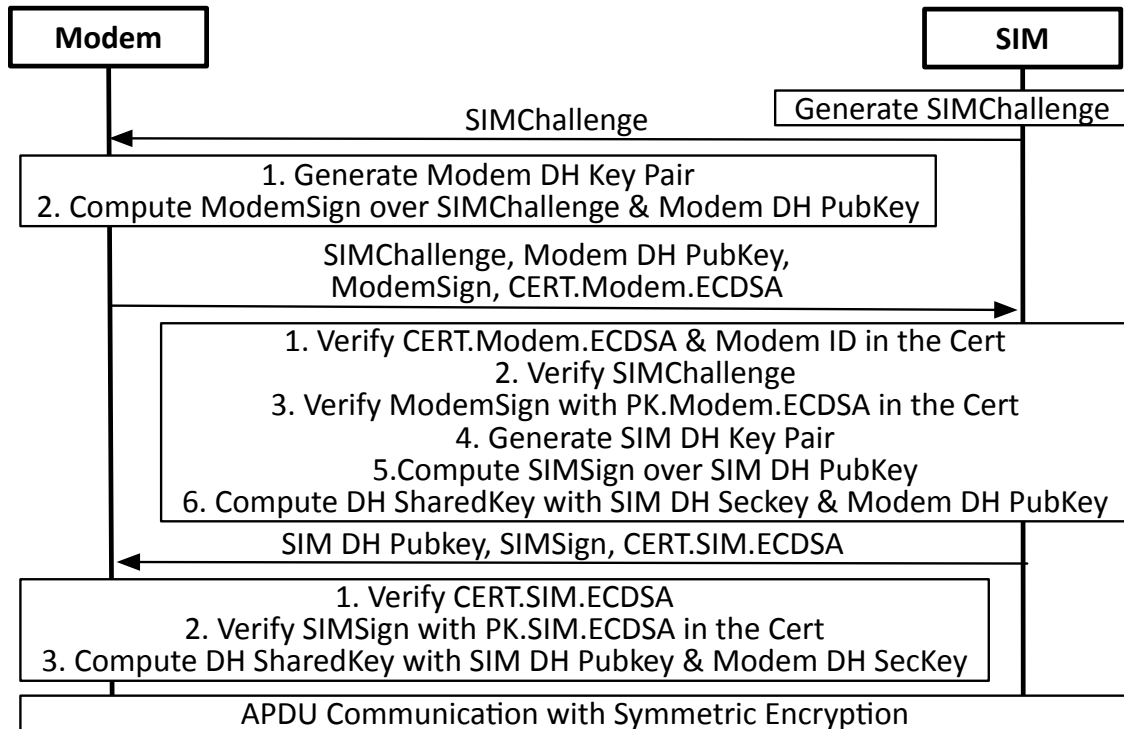


Figure 6.12: SIM-Modem APDU encryption.

Regarding usability, registering the Modem ID does not require extra steps compared with the existing deployment.

6.3.2.2 Selective Encryption

Limiting the SIM access to certified roles cannot prevent eavesdropping on the plaintext APDU channel (**V3**). We design the APDU encryption mechanism between the SIM and modem to prevent the SIM sticker attack. Generally, asymmetric encryption takes relatively more time than symmetric encryption [PNN16, Mar01]. Considering SIM's limited resources, we first perform a Diffie-Hellman (DH) key exchange and later use the generated key to perform the symmetric encryption for later APDU traffic.

As shown in Figure 6.12, the SIM generates a random SIMChallenge and sends it to the modem. The modem generates the DH Key pair and uses its secret key of the certificate to

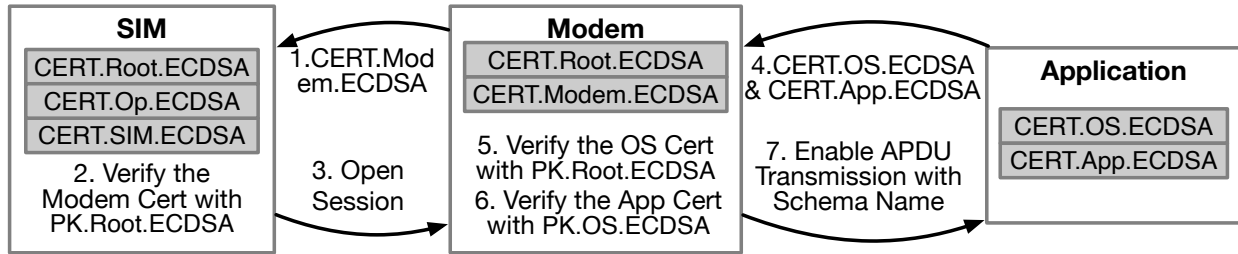


Figure 6.13: Automated verification procedure

sign the SIMChallenge and the modem DH public key. Then the modem sends the signature ModemSign with signed information and its certificate to the SIM. The SIM prevents the replay attack by a random SIMChallenge every time. After receiving the information, the SIM first verifies the modem certificate and SIMChallenge, then authenticates the ModemSign with the certificate's public key. If the modem passes all the checking, the SIM follows a similar way to send its DH public key with the signature SIMSign and its certificate. After the modem validates the SIM, both sides derive the key from the shared secrets and encrypt all later APDUs with the symmetric encryption algorithm such as AES. We further devise a selective scheme to reduce the overhead. Most of the APDUs sent by the modem are standardized and do not contain sensitive information, such as selecting a file. These APDUs can be plaintext so that the SIM can eliminate the decryption latency. The selective encryption only ciphers the sensitive information, including file contents returned by the SIM, updated file contents sent by the modem, and authentication procedures such as 5G/4G AKA. The selective encryption ensures security with marginal overheads. Current eSIM performs the package decryption and certificate verification inside the eSIM chip. Considering the modem's hardware is more powerful than eSIM chips, it is feasible to deploy the functions inside a modem by firmware updates.

6.3.2.3 Automated Verification

SecureSIM enables an automated verification scheme, as shown in Figure 6.13. Considering the limited RAM and computation resources, performing all certificate authentication

in the SIM increases the processing latency and affects the user experience. To speed up the verification procedure, SecureSIM offloads the certificate validation to the modem after authenticating its certificates. Compared with OS or applications, which are vulnerable on root devices, the attacker is hard to compromise a modem. Thus, the modem can be a trustworthy device for application authentication. After the modem validates the application's certificate, it sends the application's *SchemaName* to the SIM and enables the APDU communication with selective encryption. For in-card applets, SecureSIM verifies their certificates automatically before they access files. Through automated verification, SecureSIM can distinguish various entities with negligible overheads.

For both the SIM and the modem, SecureSIM's design does not change the file contents or the APDU interfaces, or require extra hardware. All functions can be updated through SIM or modem firmware updates over the air. SecureSIM schemes could be enabled based on the SIM/modem's capability, which provides backward compatibility for current devices and can be rolled out gradually. Considering the usability, the current SIM PIN is disabled by default and requires users to enable it manually. SecureSIM does not require users to manually enable the PIN or enter the PIN every time the phone starts. All the verifications are automated through certificates and do not affect users' daily usage.

6.3.3 Security Analysis

We analyze the security of SecureSIM in comparison to the current SIM design.

5G/4G Key Leakage The 5G/4G key leakages allow attackers to perform eavesdropping, MitM, and impersonation attacks. SecureSIM provides fine-grained access control and certificate verification. Only whitelisted modems can acquire the security-related keys. The attacker cannot read them by hardware such as a card reader or a random modem out of the whitelist. The malware cannot pass the automated verification even with the root privilege.

APDU MitM SecureSIM protects the APDU communication with selective encryption

between the SIM and modem. Both sides sign the APDUs in the key agreement procedure so that the attacker cannot forge or manipulate them. The symmetric encryption for APDUs protects all sensitive files.

Physical Thefts SecureSIM leverages the modem whitelist to restrict SIM usage on credible devices. Physical thefts cannot use the stolen SIM on other devices for data service or impersonate the victim's identity.

SIM Cloning The 2G/3G SIM algorithm's weakness allows SIM cloning attack [SRL13, RRS02]. 5G/4G SIM cards improve the algorithm security, but attackers can still get the master K to clone the SIM from side-channel attacks like differential power analysis [LYS15, IIT02]. Improved implementations with anti-cloning mechanisms can defend these attacks [AT07, HOM06]. SecureSIM only grants whitelisted devices to perform AKA authentication, increasing the difficulty of collecting traces for compromising. Even if the master K leaks, attackers cannot compromise the certificates and clone the SIM.

SIM Swapping SIM swapping occurs when the attacker tricks the operator into porting the victim's phone number to the attacker's SIM. For example, the attacker acquires the victim's personal information through phishing emails or social engineering and claims that they have lost their phone. While SecureSIM prevents the attacker from accessing the SIM from unauthorized devices, defending against SIM swap attacks still requires the operator to improve the security checking like a mandatory strong account password.

Abused Modem If attackers extract modem keys and impersonate a certified modem, the SIM security is risked. However, it is hard to be achieved in practice. First, the modem firmware is usually closed-sourced and hard to be attacked. Modems also provide separate hardware and software stacks from the host device, making it difficult to execute malicious codes and be abused. Second, even if the attacker abuses a specific modem, critical files such as keys in SecureSIM could only be accessed by whitelist modems. Abused modem with other Modem ID cannot access keys successfully.

6.4 Implementation

We implement SecureSIM with Javacard, one of the most prominent platforms for smart card development. Javacard has similar hardware capabilities to the commercial SIM. Operators could build SIM with smart cards and install an applet running SIM functions. Javacard defines a subset of the Java programming language and provides APIs for smart cards [Jav]. We develop a SIM card applet as a platform that can support standard SIM file operations and the MILENAGE algorithm to perform the 5G/4G AKA mutual authentication. It consists of 5263 lines of Java code. To the best of our knowledge, there is no open-source SIM implementation on the smart card. We release the applet at [Flo].

We deploy the SecureSIM based on our SIM applet. We develop an access policy parser with Python. It parses access policies provided by operators, checks for conflicts, and generates the corresponding access graph for the SIM applet. The SIM applet holds files as standardized in [3GP20a]. In the standard, each file contains a header, including file format, length, etc. We extend the header with the file's profile category. For each profile, schema nodes encode granted operations with one byte, which complies with the standard [ISO13]. When an entity starts a new APDU session, the SIM matches the *SchemaName* from the certificate verification procedure. By checking the schema tree, SecureSIM could acquire the entity's privileges for each profile type. When the entity performs file operations such as Read or Update, SecureSIM grants access based on the file's profile.

SecureSIM adopts certificate-based authentication and communication. We generate the certificate chain with X.509 v3 standards and add customized attribute *SchemaName*. For certificate algorithms, we select ECDSA for its smaller key size compared with RSA. The current phone modem firmware is closed-source. Thus, we deploy a module with Java in an Android application to simulate the modem's selective encryption, automated verification, and certificate revocation procedure. We use the Elliptic-curve Diffie-Hellman (ECDH) over the secp256k1 curve to perform the key agreement described in section 6.3.2.2 and gener-

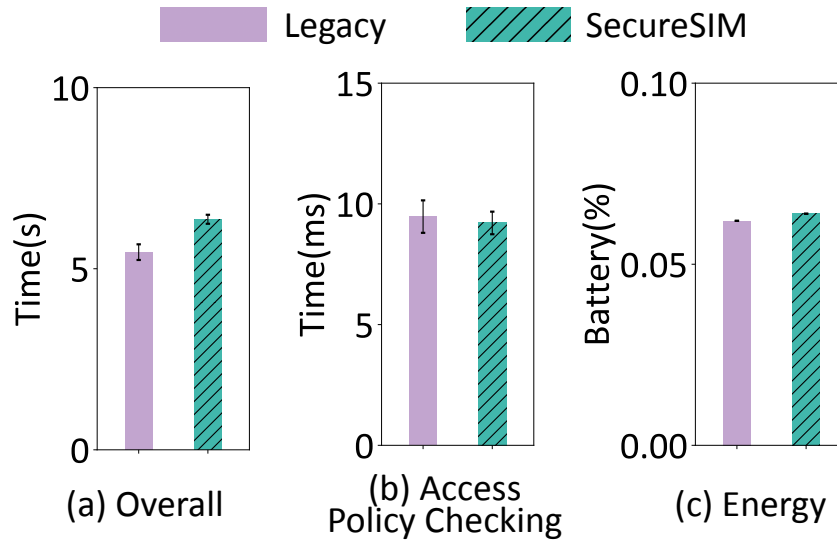


Figure 6.14: Performance comparison.

ate shared secrets. The later symmetric encryption uses the AES-128-CBC algorithm. For automated verification, the SIM first authenticates the module with the modem certificate `CERT.Modem.ECDSA`, then the module handles all the other certificate verifications automatically. The module also collaborates with SIM for storing and updating the certificate revocation list.

6.5 Evaluation

We evaluate the performance of SecureSIM on J3R180 Javacard supporting Java Card 3.0.5 specifications with 180KB EEPROM and 8KB RAM. We assess the performance with Google Pixel 4a with Qualcomm Snapdragon 730G running Android 10. We first compare the overall performance of the commercial SIM card (Legacy) and SecureSIM. We then evaluate each module’s performance, including certificate verification, selective encryption, and access policy checking. We further measure the energy overhead introduced by new mechanisms in SecureSIM. The results show that the overhead of SecureSIM is marginal compared with the current SIM design.

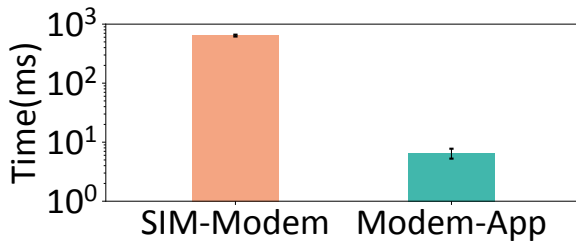


Figure 6.15: Certificate verification

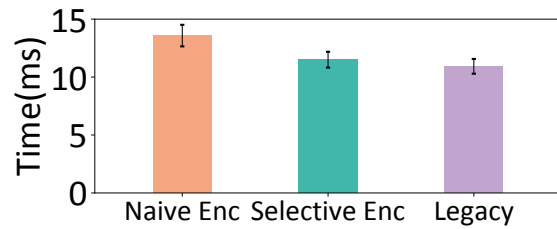


Figure 6.16: Encryption.

Overall Evaluation We use SIMTrace to record all the APDUs when the modem loads SIM files once the SIM is inserted into the phone. Then we replay the APDUs and measure the overall performance at the SIM initial stage between the commercial SIM card and SecureSIM. The results show that SecureSIM takes $6.3 \pm 0.1s$, while the commercial SIM takes $5.4 \pm 0.2s$, as shown in Figure 6.14(a). The execution time increases mainly because of certificate verification. It only happens once at the start of the APDU session and will not incur overhead for follow-up communication. Our implementation is based on the generic smart card API. It cannot directly control the hardware resource. Customized devices could further improve execution efficiency. Moreover, the eSIM chip with more powerful computational resources can also reduce the overhead.

Certificate Verification We measure the time for the automated verification procedure. SecureSIM delegates its authentication task to the modem after the SIM validates its certificates. We compare the SIM-modem and modem-application verification time. As shown in Figure 6.15, the SIM-modem authentication time is $639 \pm 23ms$. It only happens once when the modem starts the APDU session. The session can last within the SIM power cycle, which can be hours or days. SecureSIM leverages the modem to accelerate application authentication. The automated verification can reduce the modem-application verification time to $6.5 \pm 1.2ms$. It only happens once when the app requires SIM access and incurs marginal overhead.

Encryption We measure APDUs that read files from the SIM with and without selective encryption. With symmetric encryption, reading a file needs on average $11.5 \pm 0.6ms$,

comparable with the Legacy without encryption ($10.9 \pm 0.6ms$), as shown in Figure 6.16. Furthermore, we compare the performance of encrypting all APDUs (naive encryption) and selective encryption in SecureSIM. Experiments show that selective encryption can reduce 15% of the average time (from $13.6 \pm 0.9ms$ to $11.5 \pm 0.6ms$). We measure the benefits of selective encryption at the initial stage. The results show that selective encryption could reduce the average package loading time by 1.1s (from $7.4 \pm 0.1s$ to $6.3 \pm 0.1s$).

Access Policy Checking SecureSIM matches the entity’s *SchemaName* with the access graph once and gets corresponding access policies. Then SecureSIM only needs to check the file’s profile type to grant access. We measure the operation execution time of SecureSIM access policy checking and the Legacy with access mode checking. Figure 6.14(b) shows that SecureSIM ($9.2 \pm 0.4ms$) will not introduce extra overhead compared with Legacy ($9.5 \pm 0.7ms$).

Energy Consumption We further measure the energy consumption of SecureSIM and Legacy. We measure the 1-hour energy consumption of an application that reads SIM files through APDU every 50ms. The results show that SecureSIM consumes comparable power (0.064%) as Legacy (0.062%), as shown in Figure 6.14(c). The increased energy consumption (3.2%) caused by encryption is marginal. We measure the modem’s daily APDU traffic pattern with SIMTrace. During the whole session, the modem sends 0.2 APDU/s. The power consumption will be more insignificant. We also measure the energy overhead caused by automated verification. Our test application launches the certificate verification every 100ms. The 1-hour test consumes 1.1% of the power. Usually, the application only needs to authenticate once at the start, thus incurring negligible amortized energy overhead.

6.6 Discussion

Here we discuss whether our findings apply to new security schemes in 5G networks. The 5G SIM does not transmit the permanent identifier, such as IMSI, during the attachment

to improve privacy protection. However, it still stores the security context and temporary identities for fast reattachment. 5G defines three authentication methods: 5G-AKA, EAP-AKA, and EAP-TLS. The 5G-AKA and EAP-AKA utilize a similar pre-shared key scheme as the 5G/4G AKA. 5G introduces EAP-TLS for private networks or IoT environments to handle the devices without SIM to perform the AKA authentication, such as laptops or IoT sensors. EAP-TLS sets up the TLS connection based on the devices' certificates. EAP-TLS does not rely on the symmetric master key K but generates keys from the TLS session. However, current vulnerabilities still exist in 5G. All three methods derive a shared key called K_{AMF} , which is still stored in the SIM. Furthermore, the 5G SIM adds support to store intermediate authentication keys. 5G also supports fast reattachment with the cached security context. These files are under current PIN-based protection and put 5G users at risk of traffic eavesdropping, MitM, and impersonation attacks.

The 5G/4G trust models require the long-term symmetric key. All the keys depend on the master key K , which exposes high operational risk during the multi-layer key derivation and transmission. In future work, certificates in the SIM make a case to redesign the 5G authentication scheme. Certificates can derive one-time keys to set up security between the UE, serving network, and home network. Multi-layer keys can be independent and flexible for heterogeneous devices and services.

6.7 Related Work

The current SIM design has been largely inherited from the legacy 2G/3G networks, where end devices do not assume much intelligence. Prior studies have exploited its implementation vulnerabilities to clone SIM cards [LYS15], track user location [Simb], or attack VoLTE calls [CSP17]. Recent efforts exploit the vulnerability in the remote SIM provisioning procedure to exhaust the memory at eSIM [MQS18, CZA18]. In contrast, we look into the vulnerability of in-card file access for both SIM and eSIM, and use the obtained information to reconstruct

the keys for 5G/4G attacks. We thus allow for attacks against both the victim device and the 5G/4G infrastructure within the standards. Therefore, we differ from existing studies that use the 5G/4G design or implementation vulnerabilities to attack the victim user [LLL20, RKH19, RKH20, SBA15] or the network [LTP15, XTL20, HYA18].

To the best of our knowledge, we report the first effort on authentication and access control for SIM/eSIM. We take a novel approach with the access graph to enable fine-grained access control on SIM files. It differs from the recent graph-based vulnerability analysis in account access [HRS19]. Prior efforts analyzed the mobile operating system’s access control and vulnerability for both Android [SST18] and iOS [DCB20]. They further proposed access control designs for mobile operating systems [RCC12, BHS13, NKZ10] and IoT applications [SST18, ZML18]. Such OS or IoT access control schemes cannot prevent information leakage at SIM/eSIM.

6.8 Conclusion

The security of SIM/eSIM is critical to both mobile users and 5G/4G infrastructure operations, since it contains various IDs, security keys, and personal information of a mobile user. Despite its importance, SIM/eSIM security is a largely unaddressed topic in the research community. This is partly because SIM/eSIM is widely considered as a closed module only accessible to mobile operators and vendors. However, as recent initiatives seek to open up SIM and make it more programmable (e.g., adding new over-the-air services into SIM/eSIM [ETS19a]), the conventional wisdom probably will not hold. As a result, the current SIM protection is not sufficient. The fundamental problem is that, SIM/eSIM does not offer proper authentication to various in-SIM applets and off-card units. Moreover, it does not provide fine-grained access control to hundreds of in-SIM files that store keys and personal data. Consequently, various attacks are made feasible. In this chapter, we uncover three such vulnerabilities, devise new attacks against both the victim device and the 5G/4G in-

frastructure, and propose solutions. Our prototype and evaluation have validated both the attacks and our defenses.

CHAPTER 7

Highly Resilient Open-Source 5G Platform

We establish a testbed for demonstrating the highly resilient 5G system designs. We develop the first open-source, Javacard-based eSIM platform. Moreover, we have designed an open-source mobile AR/VR platform to showcase the incoming 5G applications. The testbed utilizes commercially available 5G and 5G IoT devices equipped with our Javacard-based eSIM and executes the mobile AR/VR application, allowing us to assess resilience on both 5G devices and networks. This chapter is organized as follows: we first introduce the eSIM platform in §7.1. We then describe the 5G/4G mobile AR/VR platform in §7.2.

7.1 eSIM Platform

The open-source SIM card promotes transparency, innovation, and accessibility in mobile communication technologies. The importance of open-source SIM cards stems from their ability to offer users increased control and customization, enabling them to switch carriers or select the best network profiles with ease. Furthermore, open-source solutions foster collaboration and innovation, allowing developers to build upon existing technology and address the specific needs of different user segments. Despite its importance, there has not been a widely-adopted open-source SIM card project. This may be due to the complex nature of telecommunications technology, and the dominance of proprietary solutions by major manufacturers and network operators.

To address this gap, We develop the first open-source eSIM platform [Flo23b]. We

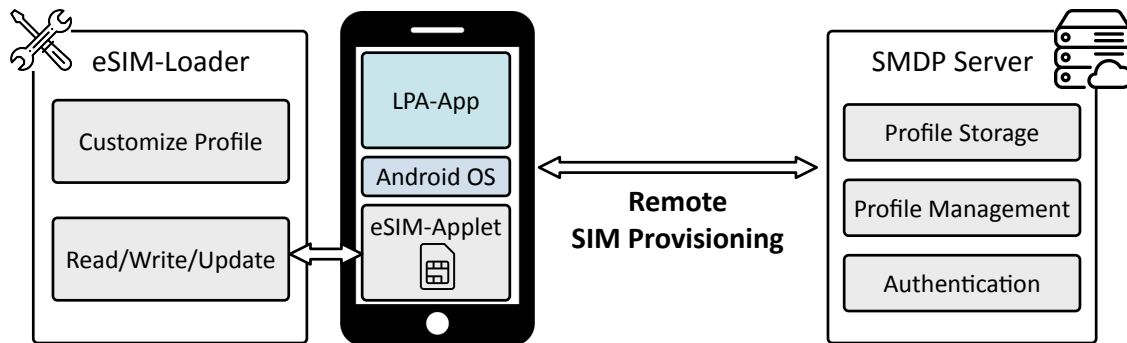


Figure 7.1: eSIM platform.

leverage Javacard, which is a widely-used, secure, and flexible platform that has been employed for various applications, including smart cards, secure tokens, and SIM cards. By combining the versatility of Javacard with eSIM technology, the platform aims to provide an open-source solution for both single and multi-profile SIM cards. This applet features a basic SIM file structure, SIM authentication, and customizable APDU handling functions, making it compatible with a wide range of mobile devices and network configurations.

The eSIM platform consists of four critical components, as illustrated in Figure 7.1, which work in tandem to provide a seamless and efficient eSIM experience: (1) an applet that enables basic SIM file structure for single and multi-profile, SIM authentication, and customized APDU handling functions; (2) a SIM profile loader program designed to build the eSIM multi-profile structure on the eSIM-Applet, as well as read, write, and customize all fields and parameters within the profile; (3) an Android app that facilitates profile downloading, installation, switching, and deletion on mobile devices by integrating Javacard with the eSIM-Applet, and enables automatic carrier selection, presenting two cases for auto-selection based on latency and throughput, while offering APIs for users to deploy their own algorithms; (4) a simplified SM-DP+ server that supplies profile download functions to the LPA-App, allowing dynamic profile downloading from a cloud server and subsequent installation as a local profile with WING-SMDP.

eSIM-Applet The eSIM applet, supported by the Javacard-eSIM, offers an extensive

feature set catering to the varied needs of users and developers. It enables read/write operations on standardized and customized SIM structures, ensuring compatibility with diverse network profiles. The applet also incorporates the MILENAGE algorithm for robust SIM authentication, which is vital for secure and reliable mobile communications. Additionally, the multi-profile storage feature allows for smooth transitions between network profiles, granting users the flexibility to choose the most suitable carrier and plan. The eSIM applet facilitates profile installation, deletion, and switching, while also supporting basic proactive commands for managing mobile communications. Finally, the dynamic profile downloading feature, in conjunction with the LPA-App and WING-SMDP, enables users to effortlessly access and update their eSIM profiles as required.

eSIM-Loader eSIM-Loader is a Python tool crafted to customize the SIM profile. This loader constructs multi-profile structures for the eSIM-Applet and supports reading and writing standardized SIM/USIM EF/DF/ADF fields, such as IMSI, ICCID, Key, OPc, and others, in compliance with the 3GPP specifications [3GP20a]. Additionally, it accommodates customized SIM/USIM EF/DF/ADF fields, offering versatility in profile management.

LPA-App An Android local profile assistants (LPA) App enables the communication between Android phones and SIM cards. This app allows reading and writing standardized and customized SIM structures directly from mobile devices. It also streamlines eSIM profile downloading from the WING-SMDP server, as well as eSIM installation, deletion, and profile switching. Moreover, the LPA-App provides dynamic profiling, and automatic carrier selection, and supports customized algorithms for carrier selection, delivering a personalized user experience.

WING-SMDP The WING-SMDP (Subscription Manager for Data Preparation) server serves as a central hub for remote profile storage and management. This server supports basic authentication functions with Android phones and the eSIM-Applet, ensuring secure communication. As a cloud database for SIM profiles, WING-SMDP facilitates profile down-

load functions for the LPA-App, simplifying the process of managing and updating eSIM profiles on mobile devices.

Our eSIM platform delivers convenience, flexibility, and security for 5G/4G communication. Compliant with established standards, the SIM is designed to function seamlessly with a variety of 5G/4G devices, including smartphones and IoT modems. By offering an open-source solution, our platform fosters innovation and facilitates future advancements in SIM/eSIM design and deployment. Our platform complements the open-source RAN and Core infrastructure, effectively bridging the final gap in the open-source mobile cellular network ecosystem.

7.2 Mobile AR/VR Platform

We develop an open-source mobile AR/VR platform [Flo23a] to demonstrate the potential of emerging 5G applications. This platform delivers a high-fidelity, real-time AR/VR experience designed for 5G/4G networks, enabling users to access AR/VR content anywhere and anytime. It operates without root or system privileges, ensuring compatibility with a wide array of commercially available 5G and 5G IoT devices.

7.2.1 System Components

The platform, as shown in Figure 7.2, consists of two primary components: the Android AR/VR Controller Application and the Edge Server. The Android AR/VR Controller Application is responsible for providing real-time user data to the AR/VR edge server. This data includes gyroscope information for adjusting viewpoints, GPS location for mapping real-world locations onto virtual scenes, camera streams for real-time AR applications, and user controls that facilitate game actions (e.g., light control, scene selection).

The edge server supports various AR/VR tasks and comprises the following components:

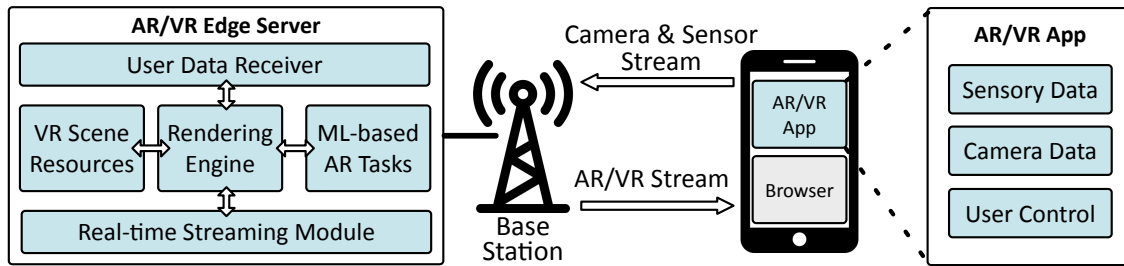


Figure 7.2: Mobile AR/VR platform.

User Data Receiver This component continuously updates the virtual camera based on real-time user movements, such as viewpoint and position changes, camera streams, and game controls. This synchronization ensures that the virtual environment remains consistent with user actions and movements. The receiver also processes user camera streams for an array of downstream AR tasks.

ML-based AR Module The AR module utilizes advanced machine learning algorithms for tasks like face and skeleton recognition, real-time multi-camera stitching, and 3D point cloud rendering. This functionality allows our platform to offer an interactive user experience, expanding the possibilities within mobile AR.

Frame Rendering Engine This engine renders the latest AR/VR frame, based on either the virtual camera’s field of view for VR or machine learning outcomes for AR. This approach ensures that users are presented with a visually captivating and immersive experience, continuously updated in response to their movements and interactions.

Real-time Streaming Module The streaming module utilizes WebRTC to deliver real-time game views, ensuring low-latency streaming ($\leq 50\text{ms}$) compared to traditional RTSP or RTMP solutions ($\geq 200\text{ms}$). It supports high-quality VR streaming, such as 1080p-60FPS and 4K-30FPS. Users can simply access the AR/VR stream through their web browsers, facilitating easy access to AR/VR content.

7.2.2 Enabling AR with 5G IoT

We further design an AR application that operates using 5G IoT connectivity. 5G IoT provides more extensive coverage and longer battery life than traditional 5G broadband networks. Specifically, the increased range of NB-IoT enables the transmission of messages from remote rural areas, extending over ten times the distance of 5G/4G networks. However, delivering AR/VR services under the constraints of limited bandwidth, which can be as low as 100 kbps, presents a significant challenge. To address this issue, we have developed an AR intrusion detection application that demonstrates the feasibility of AR on NB-IoT devices.

Our AR system offers a solution for detecting unwanted intruders that may pose a threat to remote facilities such as pipelines, farms, and ranches. IoT cameras in these areas identify the skeletons of potential intruders in real-time. The data is compressed and transmitted through the NB-IoT network to the cloud, where it is reconstructed with 3D characters to provide a clear understanding of the intruder's actions. This process enables rapid detection and monitoring of potential attacks on facilities. To deploy the system, we employed Raspberry Pi as the front-end IoT device, connected to commercial NB-IoT modems for network transmission. We further demonstrate the feasibility of our approach by leveraging software-defined radio to enable a standard-compliant base station and core network. Our open-source application provides a new benchmark for AR/VR services on cellular IoT devices. This solution allows remote facilities to be monitored and protected against intruders, ensuring the safety and security of crucial infrastructure.

In summary, our open-source mobile AR/VR platform enhances the AR/VR experience over 5G and 5G IoT networks. The applications include built-in metrics for quantifying 5G network resilience, such as throughput, latency, and disruption times. This platform serves as a benchmark for future resilient 5G network developments.

CHAPTER 8

Conclusion and Future Work

The global rollout of 5G mobile systems is advancing at an unprecedented pace, primarily targeting mission-critical and enterprise applications such as autonomous vehicles, smart factories, and remote health care that necessitate high service resiliency. As 5G deployments continue to burgeon, they achieve universal coverage and enhanced performance via higher radio spectrums. Nevertheless, challenges beyond wireless technology itself, including protocol failures, connectivity outages, and security vulnerabilities, impede the realization of highly resilient 5G services.

A key contributor to this diminished resiliency is the “smart core, dumb terminal” design philosophy prevalent in 5G systems. This paradigm emphasizes intelligent infrastructures while relegating end-user devices to passive roles, thereby neglecting the potential of devices to respond autonomously during service disruptions.

This dissertation demonstrates that augmenting device-side intelligence with the use of SIM/eSIM offers a novel approach to bolstering resiliency. SIMs facilitate simple yet effective device-side management, leading to enhanced resilience through the incorporation of lightweight operations, such as multi-tier resets and rapid multi-carrier switching. These operations empower SIMs to expedite recovery from protocol failures, enhance connectivity outage handling, and strengthen security against vulnerabilities. By harnessing the potential of device-side intelligence, SIMs actively contribute to the resilience of both 5G and 5G IoT systems, laying the foundation for more robust and reliable communication networks in the future.

In the following sections, we summarize our results in §8.1, discuss the lessons learned in §8.2, and envision future directions in §8.3.

8.1 Summary of Results

This dissertation reveals that the roadblocks toward highly resilient 5G systems are caused by a complex interplay of factors, including protocol stacks, device mobility, power-saving designs, and security mechanisms. To address these issues, we propose a novel SIM/eSIM-based approach that enhances resiliency in 5G and 5G IoT systems. By using the SIM card and eSIM chip as an independent miniature system, we offer plug-and-play, highly resilient 5G services without requiring modifications to device firmware, operating systems, or base stations. The main contributions are as follows:

SEED: A SIM-Based Solution to 5G Failures We present SEED, our first effort in diagnosing and handling 5G failures using a novel SIM-based solution. SEED exploits the available error codes in standardized 5G signaling messages to infer root causes. It further improves diagnosis with a simple, domain-specific machine-learning algorithm. Upon identifying the cause of failure, SEED takes adaptive, multi-tier reset/redo actions (resetting protocol operations, refreshing outdated configurations, reloading profiles, etc.), enabling swift diagnosis and handling of 5G protocol failures.

SHIELD: SIM-Based Rapid Connectivity Outage Handling We introduce SHIELD, a SIM-based solution aimed at highly available 5G and 5G IoT systems. The key innovation is a SIM-centric software mini-system that leverages the in-device SIM card and a plug-and-play, miniature, software-defined receiver hardware. Our approach relies on pure, device-based operations to facilitate rapid inter-carrier-network switching and data-first designs to handle mobility and power-saving-induced disruptions. Our designs readily achieve two to three nines (potentially four to five nines in the near future) of availability for 5G services, effectively eliminating connectivity outages as a system bottleneck.

SecureSIM: Enhanced 5G Security with SIM/eSIM We identify three vulnerabilities in the current SIM/eSIM practice. We demonstrate that PIN-based access control may expose in-SIM data to adversaries through both hardware and software means. Once compromised, this in-SIM data can be utilized to reconstruct various keys used for device authentication, data encryption, and more, enabling attacks such as traffic eavesdropping, man-in-the-middle attacks, and impersonation. We propose a new solution that provides both authentication and fine-grained access control to hundreds of in-SIM files for various in-card applets and off-card units. By redesigning SIM/eSIM security, we address the security vulnerabilities in current SIM designs and enhance 5G service resiliency against diverse attacks.

8.2 Lessons Learned

8.2.1 Emphasizing Device-side Intelligence for 5G Resiliency

The pursuit of highly resilient 5G systems is often hindered by the prevailing cellular design convention, which adheres to a “smart core, dumb terminal” philosophy. This approach prioritizes intelligent infrastructure, while neglecting the potential for end-user devices to play more active roles. The root cause of this imbalance can be traced back to historical design choices that emphasized centralized control, leaving little room for distributed intelligence. As a result, resiliency design has predominantly focused on network-side mechanisms, creating a single point of failure and limiting the devices’ capacity to respond autonomously during service interruptions.

It is through recognizing the inherent limitations of this design philosophy that we can begin to explore alternative approaches. This dissertation proposes a SIM-centric design, which aims to enhance device intelligence and unlock the latent potential of end-user devices to contribute to network resilience. The SIM is uniquely situated within the 5G ecosystem, being both produced and managed by the network, while residing on cellular devices. This

distinctive role empowers the SIM to act as an ambassador, enabling device-side intelligence while leveraging the network knowledge and ensuring high resiliency in 5G networks.

For example, SEED enables fine-grained diagnosis with multi-tier reset for fast failure handling, even under disruptions (§4). Additionally, SHIELD facilitates rapid carrier switching and a data-first design at the device level during connectivity outages (§5). By capitalizing on the SIM’s unique position, we can effectively challenge the status quo of “dumb terminal” designs and usher in a new era of device-side resiliency that is both simple and effective. This shift in design philosophy not only addresses the limitations of the current approach but also paves the way for a more robust and adaptive 5G infrastructure.

8.2.2 High Resiliency Is Beyond Enhancing Wireless

The notion that disruptions in cellular networks result primarily from weak signal conditions is an oversimplification. This dissertation reveals that multiple factors contribute to the lack of resiliency in 5G networks, even under optimal signal conditions. The complex interplay among protocols, mobility and power-saving schemes, and security vulnerabilities collectively constrains the potential for high resiliency.

This work offers insights into both current 5G operations and future deployments. The analysis presented herein allows for a more accurate diagnosis of network failures by distinguishing between wireless and non-wireless issues. Understanding these causes can guide efforts to develop more resilient and reliable 5G systems. Additionally, the solutions proposed in this dissertation, such as multi-tier reset mechanisms (§4.3.4), rapid multi-carrier switching (§5.3.2), and fine-grained access control (§6.3), effectively address both wireless and non-wireless challenges at the 5G device. These designs can be readily deployable to enhance the resiliency of 5G services on existing 5G devices.

8.2.3 Optimization Can Be Plug-and-Play

The 5G mobile ecosystem encompasses numerous mobile network operators, device vendors, OS vendors, and application developers. The heterogeneity of this ecosystem complicates the optimization process, and the proprietary nature of core components, such as modems, base stations, etc., further exacerbates the challenge of implementing new designs. Traditionally, optimization efforts have necessitated root privileges [LPZ21, TDZ21a] or modifications to device firmware or operating systems [LLZ20, ZLL22], often leading to lengthy deployment times or introducing potential security threats.

This dissertation demonstrates that standardized interfaces hold significant potential for optimization within the 5G ecosystem. For example, widely available SIM-device interfaces facilitate high-availability designs by enabling data-first designs and fast reconnect (§5.3.3). Additionally, the SIM leverages supported APIs to handle failures (§4.3.4), obtain real-time network conditions (§5.3.2.2), and authenticate applications (§6.3.2). The enhanced resilient designs can be implemented as plug-and-play solutions on commodity devices, without the need to modify firmware, applications, operating systems, or 5G infrastructure. In the future, leveraging readily available APIs in conjunction with standardized interfaces will provide new opportunities to repurpose existing functions to achieve more novel optimization objectives.

8.3 Looking Forward

As we stand at a crucial juncture in the evolution of mobile networking, with rapid advancements in both 5G and beyond-5G (B5G) systems, it is important to recognize the challenges that accompany this progress. While significant strides have been made in improving wireless technologies and implementing more flexible network frameworks like SDN and NFV, these solutions are insufficient to fully exploit the potential of client-side capabilities or ensure resilient network operations.

It is critical for 5G/B5G to create high-performance, highly resilient networks that can

support mission-critical services, such as Industrial IoT, autonomous vehicles, and digital twins. However, the current 5G systems have become excessively complex due to the convoluted protocol stacks and heterogeneous schemes introduced incrementally between devices and infrastructure components. We believe that the key to overcoming these challenges lies in harnessing the potential of SIM/eSIM technology to develop simplified, resilient cellular designs. By capitalizing on the SIM’s unique position within the ecosystem, we can unveil new opportunities for innovative cellular designs in the 5G/B5G core, radio access networks, and devices.

In this section, we explore future research that delves into how SIM-centric designs can bolster 5G/B5G systems. We introduce the concept of the SIM Trust Core, a secure framework that streamlines cellular designs by positioning the SIM at the heart of the network. The SIM/eSIM enables a trusted and simplified 5G network, fostering a more efficient and secure architecture across various network entities: the trusted state for core network design (§8.3.1), trusted access for wireless fusion (§8.3.2), and trusted AI/ML for cellular devices (§8.3.3). By embracing the SIM Trust Core, we can address the root cause of complexity and pave the way for a more reliable, adaptable, and high-performing mobile networking landscape.

8.3.1 State Snapshot for Resilient 5G/B5G Core Design

As 5G and beyond-5G (B5G) networks continue to evolve, addressing their inherent complexities and vulnerabilities become essential. The current design, which relies on multiple states at different protocol levels, such as RRC/NAS, and corresponding gateway status management, results in complications and inconsistencies during mobility and power-saving scenarios. Additionally, the centralized network-side management of states introduces a new single point of failure, compromising network resilience.

Our future work aims to tackle these issues by proposing the SIM card as a trusted storage medium for a state snapshot, encompassing all the necessary states within the core

network. Upon state generation by the core, the snapshot will be securely transmitted to and stored in the SIM card. In the event of mobility or failures, the devices can leverage standard-compliant methods to provide the stored state copy, allowing for rapid and efficient recovery of lost control and data plane status. This approach will enable not only swift state restoration but also ensure secure state handling.

To this end, the trusted state snapshot achieves a distributed and scalable design, effectively eliminating the single point of failure inherent in current systems. By decentralizing state management, we will be able to enhance network resilience and reliability while also catering to the ever-growing demands of 5G and B5G core networks.

8.3.2 Trusted Access for Resilient Wireless Fusion

5G networks feature intricate wireless designs, marked by complex physical resource management across various spectrums at the physical layer and sophisticated control and scheduling at the MAC layer. The current designs give rise to considerable complexities and create potential security vulnerabilities. For future cellular networks that are both efficient and secure, it is crucial to simplify design and reinforce security.

Wireless fusion offers a compelling approach to alleviate these complexities by merging the cellular protocol stack with the physical layer of other wireless technologies, such as Wi-Fi. The efficient use of unlicensed spectrum could streamline the PHY and MAC designs without compromising connectivity. Furthermore, wireless fusion promotes the seamless integration of diverse wireless technologies, laying the groundwork for a unified and adaptable communication framework. However, providing secure access within such a converged environment remains an essential challenge.

Our proposed solution centers on the SIM Trust Core, which acts as a unified security anchor on devices, storing access keys for various wireless technologies. By harnessing the SIM as a centralized, secure element, devices can directly connect to different access technolo-

gies, such as Wi-Fi, 5G, and B5G, in a trusted manner. This SIM-based design heightened protection against device malware and diminished vulnerability to hacking attempts. As we progress toward a future of wireless fusion, the SIM Trust Core emerges as a vital component for secure and consolidated access management.

8.3.3 Resilient AI/ML Services for Cellular Devices

The significance of privacy continues to grow, fueling the demand for solutions that safeguard user data while still harnessing the power of AI and ML technologies. Federated learning has emerged as an approach to meet these needs, enabling AI advancements without sacrificing user privacy. By distributing ML tasks across numerous devices, federated learning allows for collaborative model training, with each device retaining its data locally—effectively reducing privacy risks.

The SIM card, traditionally employed for secure storage and communication, can be employed as a secure execution environment to support federated learning on mobile devices. Utilizing the inherent security features, it can offer a protected environment for the storage and processing of lightweight user data. Furthermore, the SIM can act as an authentication gatekeeper, ensuring that only authorized devices participate in the federated learning process. This approach paves the way for the seamless integration of privacy-preserving AI and ML techniques within the mobile ecosystem.

Looking ahead to the future of AI and mobile devices, the SIM card holds immense potential for striking a balance between intelligence and privacy. By enabling devices to securely engage in federated learning, the SIM lays the foundation for a new era in which AI development and deployment prioritize privacy. This paradigm shift will unlock unprecedented opportunities for AI and ML applications on mobile devices, while maintaining user control over their data—ultimately fostering a more secure, private, and intelligent digital landscape.

Appendix A

Supplementary Materials for Chapter 4

A.1 Standardized Failure Causes Related to Configuration Issues

Control plane management failures with the required configurations from the infrastructure:

- #26-Non-5G authentication unacceptable: Supported RAT
- #27-N1 mode not allowed: Supported RAT
- #31-Redirection to EPC required: Supported RAT
- #62-No network slices available: Suggested S-NSSAI
- #72-Non-3GPP access to 5GCN not allowed: supported RAT
- #91-DNN not supported or not subscribed in the slice: Suggested DNN
- #95-Semantically incorrect message: Invalid/missed config
- #96-Invalid mandatory information: Invalid/missed config
- #100-Conditional IE error: Invalid/missed config

Data plane management failures with the required configurations from the infrastructure:

- #27-Missing or unknown DNN: Suggested DNN
- #28-Unknown PDU session type: Suggested session type

- #33-Requested service option not subscribed: Suggested DNN
- #39-Reactivation requested: Suggested DNN
- #41-Semantic error in the TFT operation: Suggested TFT
- #42-Syntactical error in the TFT operation: Suggested TFT
- #43 –Invalid PDU session identity: Activated PDU session
- #44-Semantic errors in packet filter(s): Suggested packet filter
- #45-Syntactical error in packet filter(s): Suggested packet filter
- #54 –PDU session does not exist: Activated PDU session
- #59-Unsupported 5QI value: Suggested 5QI
- #68-Not supported SSC mode: Suggested packet filter
- #70-Missing or unknown DNN in a slice: Suggested DNN
- #83-Semantic error in the QoS operation: Suggested packet filter
- #84-Syntactical error in the QoS operation: Suggested packet filter
- #95-Semantically incorrect message: Invalid/misssed config
- #96-Invalid mandatory information: Invalid/misssed config
- #100-Conditional IE error: Invalid/misssed config

A.2 AT Commands List for Fast Failure Handling

- Modem reset: AT+CFUN
- PLMN selescion: AT+COPS

- Control-plane reattachment: AT+CGATT
- Data session setting: AT+CGDCONT
- Data plane reset: AT+CGACT

REFERENCES

- [3GP13] 3GPP. “TS24.301: Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS); Stage 3.”, Jun. 2013.
- [3GP19] 3GPP. “TS36.213: Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures.”, Sep. 2019.
- [3GP20a] 3GPP. “TS31.102: Characteristics of the Universal Subscriber Identity Module (USIM) application.”, Dec. 2020.
- [3GP20b] 3GPP. “TS33.401: 3GPP System Architecture Evolution (SAE); Security architecture.”, Jul. 2020.
- [3GP20c] 3GPP. “TS33.501: Security architecture and procedures for 5G System.”, Dec. 2020.
- [3GP20d] 3GPP. “TS38.331: NR; Radio Resource Control (RRC); Protocol specification.”, Dec. 2020.
- [3GP21a] 3GPP. “TS24.008: Mobile radio interface Layer 3 specification; Core network protocols; Stage 3.”, Apr. 2021.
- [3GP21b] 3GPP. “TS24.501: Non-Access-Stratum (NAS) protocol for 5G System (5GS); Stage 3.”, Sep. 2021.
- [3GP21c] 3GPP. “TS27.007: AT command set for User Equipment (UE).”, Oct. 2021.
- [AAB21] Kazi Main Uddin Ahmed, Manuel Alvarez, and Math HJ Bollen. “Reliability analysis of internal power supply architecture of data centers in terms of power losses.” *Electric Power Systems Research*, **193**:107025, 2021.
- [AC19] Alaa AR Alsaedy and Edwin KP Chong. “Mobility management for 5G IoT devices: Improving power consumption with lightweight signaling overhead.” *IEEE Internet of Things Journal*, **6**(5):8237–8247, 2019.
- [AD22] Dharun Anandayavaraj and James C Davis. “Reflecting on Recurring Failures in IoT Development.” *arXiv preprint arXiv:2206.13560*, 2022.
- [AFP17] Mario Almeida, Alessandro Finamore, Diego Perino, Narseo Vallina-Rodriguez, and Matteo Varvello. “Dissecting dns stakeholders in mobile networks.” In *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies*, pp. 28–34, 2017.

- [AHB21] Amir Afaq, Noman Haider, Muhammad Zeeshan Baig, Komal S Khan, Muhammad Imran, and Imran Razzak. “Machine learning for 5G security: Architecture, recent advances, and challenges.” *Ad Hoc Networks*, **123**:102667, 2021.
- [AJI20] Mukhtiar Ahmad, Syed Usman Jafri, Azam Ikram, Wasiq Noor Ahmad Qasmi, Muhammad Ali Nawazish, Zartash Afzal Uzmi, and Zafar Ayyub Qazi. “A low latency and consistent cellular control plane.” 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. 648–661, 2020.
- [And] “Android Application ID.” <https://developer.android.com/studio/build/application-id>.
- [And22a] “Android Runtime API.” <https://developer.android.com/reference/java/lang/Runtime>, Jan. 2022.
- [And22b] “Android Service API.” <https://developer.android.com/reference/android/app/Service>, Jan. 2022.
- [And22c] “Android TelephonyManager.” <https://developer.android.com/reference/android/telephony/TelephonyManager>, Jan. 2022.
- [And22d] “Connectivity Diagnostics API.” <https://source.android.com/devices/tech/connect/connectivity-diagnostics-api>, Jan. 2022.
- [And22e] Android. “Data Connection Tracker.” <https://cs.android.com/android/platform/superproject/+master:frameworks/opt/telephony/src/java/com/android/internal/telephony/dataconnection/DcTracker.java>, Jan. 2022.
- [And22f] Android. “DataFailCause.” <https://developer.android.com/reference/android/telephony/DataFailCause>, Jan. 2022.
- [And22g] Android. “UICC Carrier Privileges.” <https://source.android.com/devices/tech/config/uicc>, Jan. 2022.
- [APD23] “Smart card application protocol data unit.” https://en.wikipedia.org/wiki/Smart_card_application_protocol_data_unit, Mar. 2023.
- [App] “Apple Bundle ID.” https://developer.apple.com/documentation/appstoreconnectapi/bundle_ids.
- [ARA18] Abdulraqueb Alhammadi, Mardeni Roslee, Mohamad Yusoff Alias, Ibraheem Shayeaa, and Saddam Alraih. “Dynamic handover control parameters for LTE-A/5G mobile communications.” In *2018 Advances in Wireless and Optical Communications (RTUWO)*, pp. 39–44. IEEE, 2018.

- [AT] “AT&T PIN Setting.” <https://www.att.com/support/article/wireless/KM1000485/>.
- [AT07] Sarang Aravamuthan and Viswanatha Rao Thumparthy. “A parallelization of ECDSA resistant to simple power analysis attacks.” In *2007 2nd International Conference on Communication Systems Software and Middleware*, pp. 1–7. IEEE, 2007.
- [AT20] “DNS Server Issue.” <https://forums.att.com/conversations/att-internet-equipment/dns-server-isnt-responding/5e7a4e47fd0835122f9dd15c>, Mar. 2020.
- [AT22] “Suggestions to recover mobile data.” <https://www.att.com/support/article/wireless/KM1349609/>, Jan. 2022.
- [AWS23a] “AWS Service Level Agreements (SLAs).” <https://aws.amazon.com/legal/service-level-agreements/>, Mar. 2023.
- [AWS23b] “AWS IoT Core.” <https://aws.amazon.com/iot-core/>, Mar. 2023.
- [AY19] Md Abdulla Al Mamun and Mehmet Rasit Yuce. “Sensors and systems for wearable environmental monitoring toward IoT-enabled applications: A review.” *IEEE Sensors Journal*, **19**(18):7771–7788, 2019.
- [AZH12] Kathy Wain Yee Au, Yi Fan Zhou, Zhen Huang, and David Lie. “Pscout: analyzing the android permission specification.” In *Proceedings of the 2012 ACM conference on Computer and communications security*, pp. 217–228, 2012.
- [Azu22] “Azure for operators.” <https://azure.microsoft.com/en-us/industries/telecommunications/>, Jan. 2022.
- [BAM20] Alcardo Alex Barakabitzze, Arslan Ahmad, Rashid Mijumbi, and Andrew Hines. “5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges.” *Computer Networks*, **167**:106984, 2020.
- [BHS13] Sven Bugiel, Stephen Heuser, and Ahmad-Reza Sadeghi. “Flexible and fine-grained mandatory access control on android for diverse security and privacy policies.” In *22nd USENIX Security Symposium (USENIX Security 13)*, pp. 131–146, 2013.
- [BIG23] “BIG7: 7-Port USB hub for Raspberry Pi.” <https://www.uugear.com/product/7-port-usb-hub-for-raspberry-pi/>, Mar. 2023.
- [BKS19] Tristan Braud, Teemu Kämäräinen, Matti Siekkinen, and Pan Hui. “Multi-carrier measurement study of mobile network latency: The tale of hong kong and helsinki.” In *2019 15th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN)*, pp. 1–6. IEEE, 2019.

- [BTY20] Tom Barbette, Chen Tang, Haoran Yao, Dejan Kostic, Gerald Q Maguire Jr, Panagiotis Papadimitratos, and Marco Chiesa. “A High-Speed Load-Balancer Design with Guaranteed Per-Connection-Consistency.” In *NSDI*, pp. 667–683, 2020.
- [Cel23] Cellmapper. “Cellmapper.” <https://www.cellmapper.net/map>, Feb. 2023.
- [Chr22] “Chrome.” <https://www.google.com/chrome/>, Jun. 2022.
- [Cis22] Cisco. “Failure Disconnect Reasons Reference.” https://www.cisco.com/c/en/us/td/docs/wireless/ucc/smf/2021-01-0/SMF_Metrics_Ref/b_ucc-5g-smf-metrics-ref_2021-01/m_failure-disconnect-reasons-reference.html, Jan. 2022.
- [Cis23] Cisco. “Ultra Cloud Core 5G Session Management Function.” https://www.cisco.com/c/en/us/td/docs/wireless/ucc/smf/2021-02-0/SMF_Config_Admin/b_ucc-5g-smf-config-and-admin-guide_2021-02/m_nas_timers_support.html, Jan. 2023.
- [Com21] T mobile Community. “Good signal/bars but no network connections.” <https://community.t-mobile.com/network-coverage-5/good-signal-bars-but-no-network-connection-many-times-daily-36756>, Jul. 2021.
- [Cor23] Infinite Internet Corporation. “Infinite Modem Band Locking Guide.” <https://infiniteic.com/band-locking-guide>, Mar. 2023.
- [CSB20] Gino Carrozzo, M Shuaib Siddiqui, August Betzler, José Bonnet, Gregorio Martinez Perez, Aurora Ramos, and Tejas Subramanya. “AI-driven zero-touch operations, security and trust in multi-operator 5G networks: A conceptual architecture.” In *2020 European conference on networks and communications (EuCNC)*, pp. 254–258. IEEE, 2020.
- [CSP17] Sreepriya Chalakkal, H Schmidt, and S Park. “Practical attacks on volte and vowifi.” *ERNW Enno Rey Netzwerke, Tech. Rep*, 2017.
- [CYT18] Kang Leng Chiew, Kelvin Sheng Chek Yong, and Choon Lin Tan. “A survey of phishing attacks: Their types, vectors and technical approaches.” *Expert Systems with Applications*, **106**:1–20, 2018.
- [CZA18] S Chitroub, N Zidouni, H Aouadia, D Blaid, and R Laouar. “SIM Card of the Next-Generation Wireless Networks: Security, Potential Vulnerabilities and Solutions.” In *2018 2nd European Conference on Electrical Engineering and Computer Science (EECS)*, pp. 502–509. IEEE, 2018.

- [DCB20] Luke Deshotels, Costin Carabas, Jordan Beichler, Răzvan Deaconescu, and William Enck. “Kobold: Evaluating decentralized access control for remote NSXPC methods on iOS.” In *2020 IEEE Symposium on Security and Privacy (SP)*, pp. 1056–1070. IEEE, 2020.
- [DIG23] “Digi IX15 User Guide.” http://store.express-inc.com/pdf/X15_UG.pdf, Jan. 2023.
- [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.
- [Dow22a] DownDetector. “Failures reported on T-Mobile.” <https://downdetector.com/status/t-mobile/>, Jan. 2022.
- [Dow22b] DownDetector. “Failures reported on Verizon.” <https://downdetector.com/status/verizon/>, Jan. 2022.
- [DZT21] Boyan Ding, Jinghao Zhao, Zhaowei Tan, and Songwu Lu. “Sonica: an open-source NB-IoT prototyping platform.” In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, pp. 868–870, 2021.
- [EB22] Saad El Jaouhari and Eric Bouvet. “Secure firmware Over-The-Air updates for IoT: Survey, challenges, and discussions.” *Internet of Things*, **18**:100508, 2022.
- [EFE23] “Wireless hydrogen sulfide (H₂S) logger.” <https://getefento.com/product/wireless-hydrogen-sulfide-h2s-logger-lte-m-nb-iot/>, Jan. 2023.
- [Emn23] Emnify. “Emnify IoT SIM Card.” <https://www.emnify.com/global-iot-sim>, Feb. 2023.
- [ETS19a] ETSI. “TS 102 223: Smart Cards;Card Application Toolkit (CAT) .”, Jul. 2019.
- [ETS19b] TS ETSI. “102 223:“Smart Cards.” *Card Application Toolkit (CAT)*, Jul. 2019.
- [FCC21] “FCC Coverage Map.” <https://fcc.maps.arcgis.com/apps/webappviewer/index.html?id=6c1b2e73d9d749cdb7bc88a0d1bdd25b>, May. 2021.
- [FHE12] Adrienne Porter Felt, Elizabeth Ha, Serge Egelman, Ariel Haney, Erika Chin, and David Wagner. “Android permissions: User attention, comprehension, and behavior.” In *Proceedings of the eighth symposium on usable privacy and security*, pp. 1–14, 2012.
- [FHM22] Stefan Farthofer, Matthias Herlich, Christian Maier, Sabrina Pochaba, Julia Lackner, and Peter Dorfinger. “CRAWDAD srfg/lte-4g-highway-drive-tests-salzburg.”, 2022.

- [Fie21] “Network slicing inches closer to reality.” <https://www.fiercewireless.com/5g/marek-s-take-network-slicing-inches-closer-to-reality>, Nov. 2021.
- [FLM20] Chongrong Fang, Haoyu Liu, Mao Miao, Jie Ye, Lei Wang, Wansheng Zhang, Daxiang Kang, Biao Lyv, Peng Cheng, and Jiming Chen. “VTrace: Automatic Diagnostic System for Persistent Packet Loss in Cloud-Scale Overlay Network.” 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. 31–43, 2020.
- [Flo] “Flora eSIM Applet.” <https://github.com/Project-Flora/Flora-eSIM>.
- [Flo23a] “Flora: Flexible Mobile Network Platform.”, Apr. 2023.
- [Flo23b] “Flora-eSIM.” <https://github.com/Project-Flora/Flora-eSIM>, Mar. 2023.
- [FPE17] Xenofon Foukas, Georgios Patounas, Ahmed Elmokashfi, and Mahesh K Marina. “Network slicing in 5G: Survey and challenges.” *IEEE Communications Magazine*, **55**(5):94–100, 2017.
- [FR21] Xenofon Foukas and Bozidar Radunovic. “Concordia: teaching the 5G vRAN to share compute.” In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, pp. 580–596, 2021.
- [FSG01] David F Ferraiolo, Ravi Sandhu, Serban Gavrila, D Richard Kuhn, and Ramaswamy Chandramouli. “Proposed NIST standard for role-based access control.” *ACM Transactions on Information and System Security (TISSEC)*, **4**(3):224–274, 2001.
- [GGD19] David M Gutierrez-Estevez, Marco Gramaglia, Antonio De Domenico, Ghina Dandachi, Sina Khatibi, Dimitris Tsolkas, Irina Balan, Andres Garcia-Saavedra, Uri Elzur, and Yue Wang. “Artificial intelligence for elastic management and orchestration of 5G networks.” *IEEE wireless communications*, **26**(5):134–141, 2019.
- [GGS16] Ismael Gomez-Miguel, 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.
- [Goo] “Google Play App Signing.” <https://developer.android.com/studio/publish/app-signing#generate-key>.
- [Goo22] “Google Map.” <https://www.google.com/maps>, Jun. 2022.

- [Goo23] “Google Fi.” <https://fi.google.com/about/>, Mar. 2023.
- [GSM] “GSMA.” <https://www.gsma.com/esim/gsma-root-ci/>.
- [GSM20] GSMA. “SGP.22 Technical Specification v2.2.2.”, Jul. 2020.
- [GYX15] Chuanxiong Guo, Lihua Yuan, Dong Xiang, Yingnong Dang, Ray Huang, Dave Maltz, Zhaoyi Liu, Vin Wang, Bin Pang, Hua Chen, et al. “Pingmesh: A large-scale system for data center network latency measurement and analysis.” In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pp. 139–152, 2015.
- [GZL19] Xiaohu Ge, Rong Zhou, and Qiang Li. “5G NFV-based tactile Internet for mission-critical IoT services.” *IEEE Internet of Things Journal*, **7**(7):6150–6163, 2019.
- [Hac23] “HackRF One.” <https://greatscottgadgets.com/hackrf/one/>, Mar. 2023.
- [Has19] Wan Haslina Hassan et al. “Current research on Internet of Things (IoT) security: A survey.” *Computer networks*, **148**:283–294, 2019.
- [Hei] “Heicard SIM.” <https://www.heicard.com/en/>.
- [HOM06] Christoph Herbst, Elisabeth Oswald, and Stefan Mangard. “An AES smart card implementation resistant to power analysis attacks.” In *International conference on applied cryptography and network security*, pp. 239–252. Springer, 2006.
- [Hom23] “Home Fires.” <https://www.ready.gov/home-fires>, Feb. 2023.
- [HP07] Sheng He and Ing Christof Paar. “SIM card security.” In *Seminar Work, Ruhr-University of Bochum*, 2007.
- [HPI18] Md Shohrab Hossain, Arnob Paul, Md Hasanul Islam, and Mohammed Atiquzzaman. “Survey of the Protection Mechanisms to the SSL-based Session Hijacking Attacks.” *Netw. Protoc. Algorithms*, **10**(1):83–108, 2018.
- [HRS19] Sven Hammann, Saša Radomirović, Ralf Sasse, and David Basin. “User account access graphs.” In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1405–1422, 2019.
- [HYA18] Limei He, Zheng Yan, and Mohammed Atiquzzaman. “LTE/LTE-A network security data collection and analysis for security measurement: A survey.” *IEEE Access*, **6**:4220–4242, 2018.
- [IIT02] Kouichi Itoh, Tetsuya Izu, and Masahiko Takenaka. “Address-bit differential power analysis of cryptographic schemes OK-ECDH and OK-ECDSA.” In *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 129–143. Springer, 2002.

- [IoT] IoTNow. “Why 98% of IoT traffic is unencrypted.” <https://www.iot-now.com/2020/06/04/103245-why-98-of-iot-traffic-is-unencrypted/>.
- [IoT22] “IoT connectivity report.” <https://blog.consoleconnect.com/what-are-the-different-types-of-iot-connectivity>, Sep. 2022.
- [ISO04] ISO/IEC. “Identification cards - Integrated circuit cards - Part 5: Registration of application providers.” *International Organization for Standardization/International Electrotechnical Commission. ISO/IEC 7816-5*, 2004.
- [ISO13] ISO/IEC. “Identification cards—Integrated circuit cards—Part 4: Organization, security and commands for interchange.” *International Organization for Standardization/International Electrotechnical Commission. ISO/IEC 7816-4*, 2013.
- [Jav] “Java Card Platform, Classic Edition 3.0.5.” <https://docs.oracle.com/javacard/3.0.5/>.
- [JG17] Ankit Kumar Jain and Brij B Gupta. “Phishing detection: analysis of visual similarity based approaches.” *Security and Communication Networks*, **2017**, 2017.
- [KCY21] Maya Kassis, Salvatore Costanzo, and Mohamad Yassin. “Flexible multi-operator ran sharing: Experimentation and validation using open source 4g/5g prototype.” In *2021 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*, pp. 205–210. IEEE, 2021.
- [KJR21] Jongyul Kim, Insu Jang, Waleed Reda, Jaeseong Im, Marco Canini, Dejan Kostić, Youngjin Kwon, Simon Peter, and Emmett Witchel. “LineFS: Efficient SmartNIC offload of a distributed file system with pipeline parallelism.” In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*, pp. 756–771, 2021.
- [KK20] Abhishek Khanna and Sanmeet Kaur. “Internet of things (IoT), applications and challenges: a comprehensive review.” *Wireless Personal Communications*, **114**:1687–1762, 2020.
- [KKK20] Mingeun Kim, Dongkwan Kim, Eunsoo Kim, Suryeon Kim, Yeongjin Jang, and Yongdae Kim. “Firmae: Towards large-scale emulation of iot firmware for dynamic analysis.” In *Annual computer security applications conference*, pp. 733–745, 2020.
- [KLH18] Vasiliki Kalavri, John Liagouris, Moritz Hoffmann, Desislava Dimitrova, Matthew Forshaw, and Timothy Roscoe. “Three steps is all you need: fast, accurate, automatic scaling decisions for distributed streaming dataflows.” In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, pp. 783–798, 2018.

- [KMA22] Saurabh Kadekodi, Francisco Maturana, Sanjith Athlur, Arif Merchant, KV Rashmi, and Gregory R Ganger. “Tiger:Disk-Adaptive Redundancy Without Placement Restrictions.” In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, pp. 413–429, 2022.
- [Kno] “KnowRoaming Travel SIM Sticker.” <https://nakedsecurity.sophos.com/2019/02/19/thousands-of-android-apps-bypass-advertising-id-to-track-users/>.
- [KR18] Deepu George Koshy and Sethuraman N Rao. “Evolution of SIM Cards—What’s Next?” In *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1963–1967. IEEE, 2018.
- [KRC22] Konstantinos Kousias, Mohammad Rajiullah, Giuseppe Caso, Usman Ali, Ozgu Alay, Anna Brunstrom, Luca De Nardis, Marco Neri, and Maria-Gabriella Di Benedetto. “A Large-Scale Dataset of 4G, NB-IoT, and 5G Non-Standalone Network Measurements.”, 2022.
- [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.
- [KRM17] Tai-hoon Kim, Carlos Ramos, and Sabah Mohammed. “Smart city and IoT.”, 2017.
- [KSK15] Changhoon Kim, Anirudh Sivaraman, Naga Katta, Antonin Bas, Advait Dixit, and Lawrence J Wobker. “In-band network telemetry via programmable data-planes.” In *ACM SIGCOMM*, volume 15, 2015.
- [KSK21] Grigorios Kakkavas, Adamantia Stamou, Vasileios Karyotis, and Symeon Papavassiliou. “Network Tomography for Efficient Monitoring in SDN-Enabled 5G Networks and Beyond: Challenges and Opportunities.” *IEEE Communications Magazine*, **59**(3):70–76, 2021.
- [LDL16] Yuanjie Li, Haotian Deng, Jiayao Li, Chunyi Peng, and Songwu Lu. “Instability in distributed mobility management: Revisiting configuration management in 3g/4g mobile networks.” In *Proceedings of the 2016 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science*, pp. 261–272, 2016.
- [LDP16] Yuanjie Li, Haotian Deng, Chunyi Peng, Zengwen Yuan, Guan-Hua Tu, Jiayao Li, and Songwu Lu. “icellular: Device-customized cellular network access on commodity smartphones.” In *13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16)*, pp. 643–656, 2016.

- [LLB17] Li Li, Daoyuan Li, Tegawendé F Bissyandé, Jacques Klein, Yves Le Traon, David Lo, and Lorenzo Cavallaro. “Understanding android app piggybacking: A systematic study of malicious code grafting.” *IEEE Transactions on Information Forensics and Security*, **12**(6):1269–1284, 2017.
- [LLL20] Yu-Han Lu, Chi-Yu Li, Yao-Yu Li, Sandy Hsin-Yu Hsiao, Tian Xie, Guan-Hua Tu, and Wei-Xun Chen. “Ghost calls from operational 4G call systems: IMS vulnerability, call DoS attack, and countermeasure.” In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, pp. 1–14, 2020.
- [LLL21] Yang Li, Hao Lin, Zhenhua Li, Yunhao Liu, Feng Qian, Liangyi Gong, Xianlong Xin, and Tianyin Xu. “A nationwide study on cellular reliability: measurement, analysis, and enhancements.” In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, pp. 597–609, 2021.
- [LLZ20] Yuanjie Li, Qianru Li, Zhehui Zhang, Ghufuran Baig, Lili Qiu, and Songwu Lu. “Beyond 5g: Reliable extreme mobility management.” 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. 344–358, 2020.
- [LoR23] “LoRa Alliance.” <https://lora-alliance.org/>, Mar. 2023.
- [LPD18] Yuanjie Li, Chunyi Peng, Haotian Deng, Zengwen Yuan, Guan-Hua Tu, Jiayao Li, Songwu Lu, and Xi Li. “Device-customized multi-carrier network access on commodity smartphones.” *IEEE/ACM Transactions on Networking*, **26**(6):2542–2555, 2018.
- [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.
- [LTP15] Chi-Yu Li, Guan-Hua Tu, Chunyi Peng, Zengwen Yuan, Yuanjie Li, Songwu Lu, and Xinbing Wang. “Insecurity of voice solution volte in lte mobile networks.” In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 316–327, 2015.

- [LVY20] Xing Liu, Christina Vlachou, Mao Yang, Feng Qian, Lidong Zhou, Chendong Wang, Lifei Zhu, Kyu-Han Kim, Gabriel Parmer, Qi Chen, et al. “Firefly: Un-tethered Multi-user VR for Commodity Mobile Devices.” In *2020 USENIX Annual Technical Conference USENIX ATC 20*, pp. 943–957, 2020.
- [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.
- [LYS15] Junrong Liu, Yu Yu, François-Xavier Standaert, Zheng Guo, Dawu Gu, Wei Sun, Yijie Ge, and Xinjun Xie. “Small tweaks do not help: Differential power analysis of milenage implementations in 3G/4G USIM cards.” In *European Symposium on Research in Computer Security*, pp. 468–480. Springer, 2015.
- [M2M] “M2M Applets.” <https://1ot.mobi/resources/blog/iot-hacking-series-6-what-is-a-sim-applet-and-why-is-it-important-for-iot-m2m>.
- [Mag22a] “Magma Core.” <https://www.magmacore.org/>, Jan. 2022.
- [Mag22b] “Magma NMS.” https://docs.magmacore.org/docs/nms/nms_arch_overview, Jan. 2022.
- [Mag22c] “Magma Orchestrator REST API.” https://docs.magmacore.org/docs/next/orc8r/dev_rest_api, Jan. 2022.
- [MAQ21] Mohammed Najah Mahdi, Abdul Rahim Ahmad, Qais Saif Qassim, Hayder Natiq, Mohammed Ahmed Subhi, and Moamin Mahmoud. “From 5G to 6G technology: meets energy, internet-of-things and machine learning: a survey.” *Applied Sciences*, **11**(17):8117, 2021.
- [Mar01] Konstantinos Markantonakis. “Is the performance of smart card cryptographic functions the real bottleneck?” In *IFIP International Information Security Conference*, pp. 77–91. Springer, 2001.
- [Mic23a] “Microsoft Azure Service Level Agreements (SLA) for Online Services.” <https://www.microsoft.com/licensing/docs/view/Service-Level-Agreements-SLA-for-Online-Services>, Mar. 2023.
- [Mic23b] “Azure IoT Hub.” <https://azure.microsoft.com/en-us/products/iot-hub>, Mar. 2023.
- [Mic23c] Lime Microsystems. “LimeSDR Mini.” <https://limemicro.com/products/boards/limesdr-mini/>, Feb. 2023.
- [MIL22] “MILab.” <http://milab.cs.purdue.edu/>, Jan. 2022.

- [MLC20] Roberto Minerva, Gyu Myoung Lee, and Noel Crespi. “Digital twin in the IoT context: A survey on technical features, scenarios, and architectural models.” *Proceedings of the IEEE*, **108**(10):1785–1824, 2020.
- [MNZ20] Samuel J Moore, Chris D Nugent, Shuai Zhang, and Ian Cleland. “IoT reliability: a review leading to 5 key research directions.” *CCF Transactions on Pervasive Computing and Interaction*, **2**:147–163, 2020.
- [Mob20] “Mobile-IoT Roaming Testing.” <https://www.gsma.com/newsroom/wp-content/uploads/NG.112-v2.0-7.pdf>, May. 2020.
- [MOC14] Toni Mastelic, Ariel Oleksiak, Holger Claussen, Ivona Brandic, Jean-Marc Pierson, and Athanasios V Vasilakos. “Cloud computing: Survey on energy efficiency.” *Acm computing surveys (csur)*, **47**(2):1–36, 2014.
- [MQS18] Maxime Meyer, Elizabeth A Quaglia, and Ben Smyth. “Attacks Against GSMA’s M2M Remote Provisioning (Short Paper).” In *International Conference on Financial Cryptography and Data Security*, pp. 243–252. Springer, 2018.
- [Net22a] “NetMotion.” <https://help.netmotionsoftware.com/support/docs/Diagnostics/450/help/DiagnosticsHelp.htm>, Jun. 2022.
- [NET22b] “NETSCOUT.” <https://www.netscout.com/>, Jan. 2022.
- [Net23] Teltonika Networks. “Band Locking.” https://wiki.teltonika-networks.com/index.php?title=Band_Lock, Mar. 2023.
- [NKZ10] Mohammad Nauman, Sohail Khan, and Xinwen Zhang. “Apex: extending android permission model and enforcement with user-defined runtime constraints.” In *Proceedings of the 5th ACM symposium on information, computer and communications security*, pp. 328–332, 2010.
- [OH16] Sharief MA Oteafy and Hossam S Hassanein. “Resilient IoT architectures over dynamic sensor networks with adaptive components.” *IEEE Internet of Things Journal*, **4**(2):474–483, 2016.
- [OJE20] Asuquo Okon, Nishant Jagannath, Ibrahim Elgendi, Jaafar MH Elmirghani, Abbas Jamalipour, and Kumudu Munasinghe. “Blockchain-enabled multi-operator small cell network for beyond 5G systems.” *IEEE Network*, **34**(5):171–177, 2020.
- [OKL11] Eunsung Oh, Bhaskar Krishnamachari, Xin Liu, and Zhisheng Niu. “Toward dynamic energy-efficient operation of cellular network infrastructure.” *IEEE Communications Magazine*, **49**(6):56–61, 2011.
- [One23] “OneSIMCard.” <https://www.onesimcard.com/>, Mar. 2023.

- [Osm23] “Osmocom SIMtrace 2.” <https://osmocom.org/projects/simtrace2/wiki>, Feb. 2023.
- [PA20] Reza Poorzare and Anna Calveras Augé. “Challenges on the way of implementing tcp over 5g networks.” *IEEE access*, **8**:176393–176415, 2020.
- [PB20] Jolly Parikh and Anuradha Basu. “Technologies assisting the paradigm shift from 4g to 5g.” *Wireless Personal Communications*, **112**:481–502, 2020.
- [PEC18] Anand Padmanabha Iyer, Li Erran Li, Mosharaf Chowdhury, and Ion Stoica. “Mitigating the latency-accuracy trade-off in mobile data analytics systems.” In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, pp. 513–528, 2018.
- [Pep23] “MAX HD2 Dual 4G LTE Mobile Router.” <https://www.onesimcard.com/>, Mar. 2023.
- [PKA10] Jani Puttonen, Janne Kurjenniemi, and Olli Alanen. “Radio problem detection assisted rescue handover for LTE.” In *21st Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, pp. 1752–1757. IEEE, 2010.
- [PLK20] Hyun-Seo Park, Yuro Lee, Tae-Joong Kim, Byung-Chul Kim, and Jae-Yong Lee. “Faster recovery from radio link failure during handover.” *IEEE Communications Letters*, **24**(8):1835–1839, 2020.
- [PNN16] Priyadarshini Patil, Prashant Narayankar, DG Narayan, and S Md Meena. “A comprehensive evaluation of cryptographic algorithms: DES, 3DES, AES, RSA and Blowfish.” *Procedia Computer Science*, **78**:617–624, 2016.
- [Pro23] Procuero. “Continuous Temperature Monitoring.” <https://procuro.com/continuous-temperature-monitoring-2/>, Feb. 2023.
- [Qua] “Qualcomm Modem ID.” <https://thingspace.verizon.com/documentation/apis/qualcomm-cat-m1-lte-modem-device-api.html>.
- [QZM20] Bo Qian, Haibo Zhou, Ting Ma, Kai Yu, Quan Yu, and Xuemin Shen. “Multi-operator spectrum sharing for massive IoT coexisting in 5G/B5G wireless networks.” *IEEE Journal on Selected Areas in Communications*, **39**(3):881–895, 2020.
- [RB14] John P Rula and Fabian E Bustamante. “Behind the curtain: Cellular dns and content replica selection.” In *Proceedings of the 2014 Conference on Internet Measurement Conference*, pp. 59–72, 2014.

- [RCC12] Giovanni Russello, Mauro Conti, Bruno Crispo, and Earlence Fernandes. “MOSES: supporting operation modes on smartphones.” In *Proceedings of the 17th ACM symposium on Access Control Models and Technologies*, pp. 3–12, 2012.
- [RCR20] Alexandru Radovici, Ioana Culic, Daniel Rosner, and Flavia Oprea. “A model for the remote deployment, update, and safe recovery for commercial sensor-based iot systems.” *Sensors*, **20**(16):4393, 2020.
- [RFW19] Joel Reardon, Álvaro Feal, Primal Wijesekera, Amit Elazari Bar On, Narseo Vallina-Rodriguez, and Serge Egelman. “50 ways to leak your data: An exploration of apps’ circumvention of the android permissions system.” In *28th USENIX Security Symposium (USENIX Security 19)*, pp. 603–620, 2019.
- [RKH19] David Rupperecht, 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.
- [RKH20] David Rupperecht, Katharina Kohls, Thorsten Holz, and Christina Pöpper. “Imp4gt: Impersonation attacks in 4g networks.” In *Symposium on Network and Distributed System Security (NDSS). ISOC*, 2020.
- [RQZ18] Darijo Raca, Jason Quinlan, Ahmed Zahran, and Cormac Sreenan. “Beyond Throughput: a 4G LTE Dataset with Channel and Context Metrics.”, June 2018.
- [RRS02] Josyula R Rao, Pankaj Rohatgi, Helmut Scherzer, and Stephane Tinguely. “Partitioning attacks: or how to rapidly clone some GSM cards.” In *Proceedings 2002 IEEE Symposium on Security and Privacy*, pp. 31–41. IEEE, 2002.
- [RVS20] SR Jino Ramson, S Vishnu, and M Shanmugam. “Applications of internet of things (iot)—an overview.” In *2020 5th international conference on devices, circuits and systems (ICDCS)*, pp. 92–95. IEEE, 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.
- [SBA21] Catarina Silva, João Paulo Barraca, and Rui Aguiar. “eSIM suitability for 5G and B5G enabled IoT verticals.” In *2021 8th International Conference on Future Internet of Things and Cloud (FiCloud)*, pp. 210–216. IEEE, 2021.
- [SCS16] Konstantinos Samdanis, Xavier Costa-Perez, and Vincenzo Sciancalepore. “From network sharing to multi-tenancy: The 5G network slice broker.” *IEEE Communications Magazine*, **54**(7):32–39, 2016.

- [SHC18] Yizhou Shan, Yutong Huang, Yilun Chen, and Yiying Zhang. “Legoos: A disseminated, distributed {OS} for hardware resource disaggregation.” In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, pp. 69–87, 2018.
- [SHH20] Martin Serror, Sacha Hack, Martin Henze, Marko Schuba, and Klaus Wehrle. “Challenges and opportunities in securing the industrial internet of things.” *IEEE Transactions on Industrial Informatics*, **17**(5):2985–2996, 2020.
- [SIMa] “RebelSIM.” <http://osmocom.org/projects/simtrace/wiki/RebelSIM>.
- [Simb] “Simjacker Technical Paper.” <https://simjacker.com/>.
- [SIM19] “What is a SIM Applet and Why is it Important for IoT & M2M?” <https://iot.mobi/resources/blog/iot-hacking-series-6-what-is-a-sim-applet-and-why-is-it-important-for-iot-m2m>, Jul. 2019.
- [SIM22] “SIMbae SIM Based Application Engine.” <https://abledevice.com/simbae/>, Jun. 2022.
- [SIM23] “SIMETRY.” <https://www.simetry.com/>, Feb. 2023.
- [SKR20] Nina Slamnik-Kriještorac, Haris Kremo, Marco Ruffini, and Johann M Marquez-Barja. “Sharing distributed and heterogeneous resources toward end-to-end 5G networks: A comprehensive survey and a taxonomy.” *IEEE Communications Surveys & Tutorials*, **22**(3):1592–1628, 2020.
- [SKS20] Kinza Shafique, Bilal A Khawaja, Farah Sabir, Sameer Qazi, and Muhammad Mustaqim. “Internet of things (IoT) for next-generation smart systems: A review of current challenges, future trends and prospects for emerging 5G-IoT scenarios.” *Ieee Access*, **8**:23022–23040, 2020.
- [SPL19] Prashast Srivastava, Hui Peng, Jiahao Li, Hamed Okhravi, Howard Shrobe, and Mathias Payer. “Firmfuzz: Automated iot firmware introspection and analysis.” In *Proceedings of the 2nd International ACM Workshop on Security and Privacy for the Internet-of-Things*, pp. 15–21, 2019.
- [SRL13] Jaspreet Singh, Ron Ruhl, and Dale Lindskog. “GSM OTA SIM cloning attack and cloning resistance in EAP-SIM and USIM.” In *2013 International Conference on Social Computing*, pp. 1005–1010. IEEE, 2013.
- [srs23] “srsRAN Project.” <https://www.srsran.com//5g>, Mar. 2023.
- [SST18] Roei Schuster, Vitaly Shmatikov, and Eran Tromer. “Situational access control in the internet of things.” In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1056–1073, 2018.

- [Str07] Daehyun Strobel. “IMSI catcher.” *Chair for Communication Security, Ruhr-Universität Bochum*, **14**, 2007.
- [T M] “T-Mobile PIN Setting.” <https://www.t-mobile.com/support/devices/android/t-mobile-revvlry-plus/sim-pin-t-mobile-revvlry-plus>.
- [T M21a] T-Mobile. “DNS Failures on T-mobile devices.” <https://community.t-mobile.com/tv-home-internet-7/dns-failures-tonight-south-carolina-on-whitebox-lte-modem-router-34319>, Jan. 2021.
- [T M21b] T-Mobile. “UDP blocking issue in 5G.” <https://community.t-mobile.com/network-coverage-5/network-blocking-udp-36746>, Jun. 2021.
- [T M22] “Internet and data issues.” <https://www.t-mobile.com/support/devices/device-troubleshooting/internet-and-data-issues>, Jan. 2022.
- [TA19] Jayashree Thota and Adnan Aijaz. “On performance evaluation of random access enhancements for 5G uRLLC.” In *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–7. IEEE, 2019.
- [tal23] “Cellular Fleet Tracker.” <https://tallymatics.com/product/tw400/>, Mar. 2023.
- [TDZ21a] Zhaowei Tan, Boyan Ding, Zhehui Zhang, Qianru Li, Yunqi Guo, and Songwu Lu. “Device-centric detection and mitigation of diameter signaling attacks against mobile core.” In *2021 IEEE Conference on Communications and Network Security (CNS)*, pp. 29–37. IEEE, 2021.
- [TDZ21b] 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.
- [Tel23] “Telit Cinterion IoT Developer Community.” <https://iot-developer.thalesgroup.com/threads/gprs-attach-retry-mechanism>, Jan. 2023.
- [the20] “Configure Outdated APN Settings.” <https://thecellguide.com/configure-apn-settings-galaxy-s8-3506>, Jul. 2020.
- [TKS16] Tarik Taleb, Adlen Ksentini, and Bruno Sericola. “On service resilience in cloud-native 5G mobile systems.” *IEEE Journal on Selected Areas in Communications*, **34**(3):483–496, 2016.
- [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.

- [Twi22] “Twitch.” <https://www.twitch.tv/>, Jan. 2022.
- [TZD23] Zhaowei Tan, Jinghao Zhao, Boyan Ding, and Songwu Lu. “CellDAM: User-Space, Rootless Detection and Mitigation for 5G Data Plane.” In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, 2023.
- [TZL21] Zhaowei Tan, Jinghao Zhao, Yuanjie Li, Yifei Xu, and Songwu Lu. “Device-Based LTE Latency Reduction at the Application Layer.” In *NSDI*, pp. 471–486, 2021.
- [U b22] U-blox. “TOBY-R2 series.” <https://www.u-blox.com/en/product/toby-r2-series>, Jan. 2022.
- [UIC] “UICC Carrier Privileges.” <https://source.android.com/devices/tech/config/uicc>.
- [Ver] “Verizon PIN Setting.” <https://www.verizon.com/support/knowledge-base-206865/>.
- [Ver22a] “My Verizon App.” https://play.google.com/store/apps/details?id=com.vzw.hss.myverizon&hl=en_US&gl=US, Jan. 2022.
- [Ver22b] Verizon. “Supported device brands.” <https://www.verizon.com/support/brands/>, Jan. 2022.
- [Ver22c] “T-mobile App.” <https://www.t-mobile.com/apps/download-t-mobile-app>, Jan. 2022.
- [Wi 23] “Wi-Fi Alliance.” <https://www.wi-fi.org/>, Mar. 2023.
- [WM 23] “Smart electricity metering on cellular.” <https://m2mserver.com/en/smart-electricity-metering-on-cellular/>, Mar. 2023.
- [WWN22] Fei Wang, Jianliang Wu, Yuhong Nan, Yousra Aafer, Xiangyu Zhang, Dongyan Xu, and Mathias Payer. “{ProFactory}: Improving {IoT} Security via Formalized Protocol Customization.” In *31st USENIX Security Symposium (USENIX Security 22)*, pp. 3879–3896, 2022.
- [XTL20] Tian Xie, Guan-Hua Tu, Chi-Yu Li, and Chunyi Peng. “How Can IoT Services Pose New Security Threats In Operational Cellular Networks?” *IEEE Transactions on Mobile Computing*, 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.
- [YLL18] Zengwen Yuan, Qianru Li, Yuanjie Li, Songwu Lu, Chunyi Peng, and George Varghese. “Resolving policy conflicts in multi-carrier cellular access.” In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, pp. 147–162, 2018.
- [You22] “Youtube.” <https://www.youtube.com/>, Jan. 2022.
- [YZH20] Deliang Yang, Xianghui Zhang, Xuan Huang, Liqian Shen, Jun Huang, Xiangmao Chang, and Guoliang Xing. “Understanding power consumption of nb-iot in the wild: tool and large-scale measurement.” In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, pp. 1–13, 2020.
- [ZDG21] Jinghao Zhao, Boyan Ding, Yunqi Guo, Zhaowei Tan, and Songwu Lu. “SecureSIM: rethinking authentication and access control for SIM/eSIM.” In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, pp. 451–464, 2021.
- [Zig23] “Zigbee.” <https://csa-iot.org/all-solutions/zigbee/>, Mar. 2023.
- [ZKC15] Yibo Zhu, Nanxi Kang, Jiaxin Cao, Albert Greenberg, Guohan Lu, Ratul Mahajan, Dave Maltz, Lihua Yuan, Ming Zhang, Ben Y Zhao, et al. “Packet-level telemetry in large datacenter networks.” In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pp. 479–491, 2015.
- [ZLL22] Zhehui Zhang, Yuanjie Li, Qianru Li, Jinghao Zhao, Ghufraan Baig, Lili Qiu, and Songwu Lu. “Movement-Based Reliable Mobility Management for Beyond 5G Cellular Networks.” *IEEE/ACM Transactions on Networking*, 2022.
- [ZML18] Wei Zhang, Yan Meng, Yugeng Liu, Xiaokuan Zhang, Yinqian Zhang, and Haojin Zhu. “Homonit: Monitoring smart home apps from encrypted traffic.” In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1074–1088, 2018.
- [ZTX22] Jinghao Zhao, Zhaowei Tan, Yifei Xu, Zhehui Zhang, and Songwu Lu. “SEED: a SIM-based solution to 5G failures.” In *Proceedings of the ACM SIGCOMM 2022 Conference*, pp. 129–142, 2022.

- [ZZL14] Xuan Zhou, Zhifeng Zhao, Rongpeng Li, Yifan Zhou, Tao Chen, Zhisheng Niu, and Honggang Zhang. “Toward 5G: When explosive bursts meet soft cloud.” *IEEE Network*, **28**(6):12–17, 2014.
- [ZZY19] Guosheng Zhu, Jun Zan, Yang Yang, and Xiaoyun Qi. “A supervised learning based QoS assurance architecture for 5G networks.” *IEEE Access*, **7**:43598–43606, 2019.