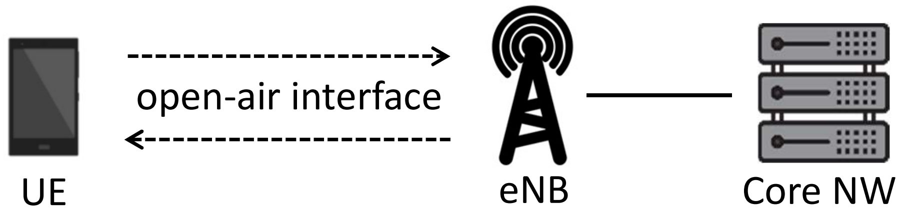
EE515 Fall 2024

LTESniffer: An Open-source LTE Downlink/Uplink Eavesdropper

Tuan D. Hoang, CheolJun Park, Mincheol Son, Teakkyung Oh, Sangwook Bae, Junho Ahn, BeomSeok Oh, and Yongdae Kim
syssec@kaist

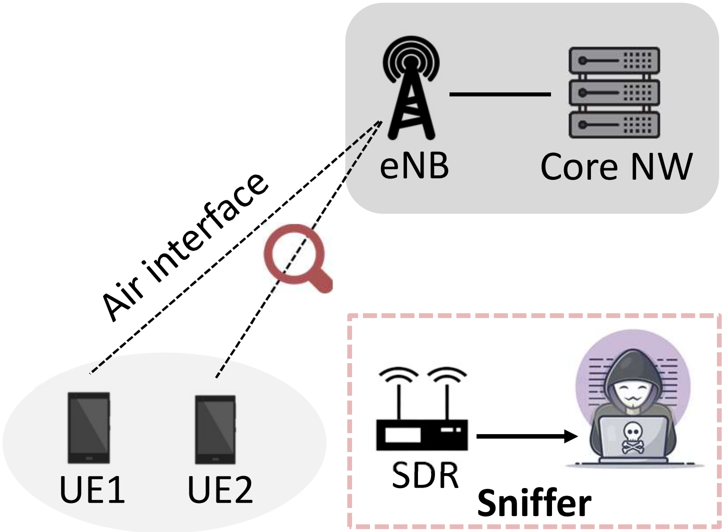
LTE Network

- Three main components: User Equipment (UE), Base Station (eNB), and Core Network
- UEs and eNB communicate over the open-air interface
- Security and analysis research in the air interface
  - Challenging due to its wireless and dynamic nature
  - Requires specialized tools such as **passive sniffers.**



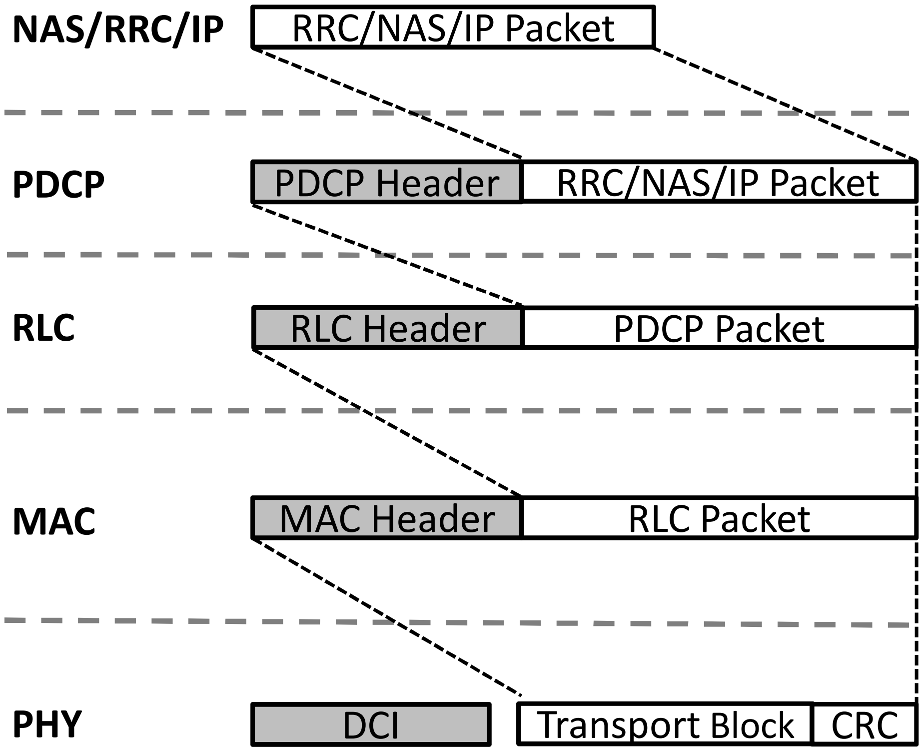
What is an LTE Sniffer?

- A passive tool capable of capturing the wireless traffic of users
  - Downlink traffic: from base stations to users
  - Uplink traffic: from users to base stations
- Mimics the behavior of both the UE and base station
- UE only decodes its own traffic, sniffer decodes all traffic of all active users



- Components:
  - Hardware: SDR for capturing wireless signals
  - Software: Program running on PC for processing and decoding signals into packets

Unprotected Information in the Air Interface



LTE Protocol Stack

- Signaling messages before AKA, broadcast msg (SIB, paging)
- User's identities used in core network (IMSI, TMSI)
- Encrypted messages after AKA

- PDCP header

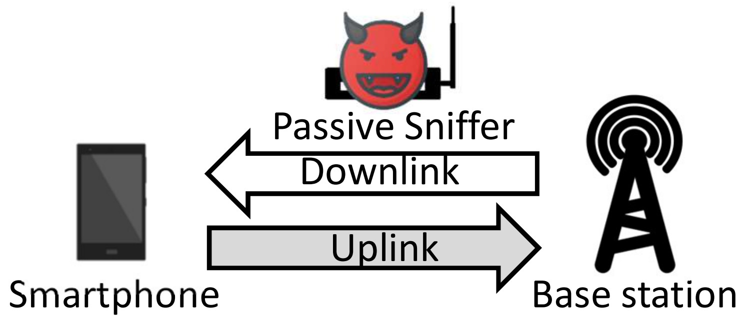
- RLC header

- MAC Control Elements (used to control radio connection)
- All info in Random Access Procedure
- MAC header

- UE's Downlink Control Information (DCI)
  - User radio identity (RNTI), detailed modulation & coding scheme, frequency resources, and time slot of UL/DL
- UL/DL signal properties in time and frequency domains
- Cell information: cell ID, MIB, sync signals

Previous Works (using a sniffer on LTE)

- Unprotected information leads to serious attacks
  - Coarse-grained user location tracking [NDSS'18]
  - Fine-grained user localization [USENIX'22]
  - Collecting and mapping identities [NDSS'18, S&P'19]
  - Video, smartphone fingerprinting [USENIX'22, NDSS'23]
- Encrypted information can be analyzed
  - ReVoLTE attack [USENIX'20]
- Attack model: Passive sniffing
  - The attacker collects over-the-air LTE packets



➢ **All need an LTE sniffer, a tool that can decode over-the-air LTE packets**

Limitations of Existing Sniffers

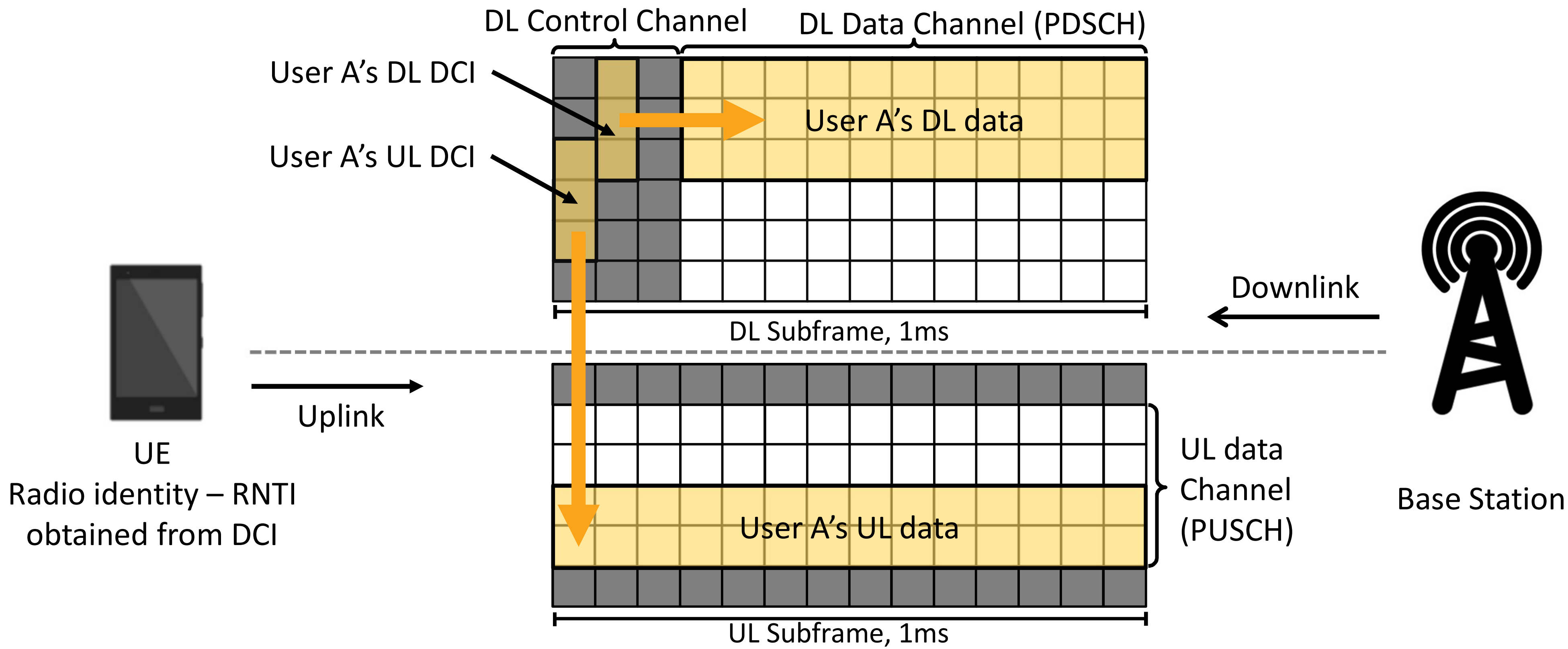
- Open-source LTE sniffers: FALCON, OWL, LTEye
  - Only decode the downlink control information
  - Cannot decode downlink data channel
  - Cannot decode uplink data channel
- Commercial LTE sniffers:
  - AirScope does not support uplink
  - Wavejudge is expensive (~USD 25,000)
  - Cannot modify code, hard to add a new feature

➢ Researchers have limited tools available for capturing over-the-air LTE packets.

➢ **Goal: Develop an open-source LTE sniffer capable of decoding uplink/downlink control/data channels**

Decode LTE Traffic

- Utilizes unencrypted Downlink Control Information (DCI) in DL control channel (PDCCH)
  - DCIs indicate how and where to decode/send data in the DL/UL data channels (PDSCH/PUSCH)



Problem and Approach [P1-A1]

- Problem [P1]: Obscure modulation scheme for each UE
  - The parameter for determining modulation schemes is transmitted via an encrypted message
- Approach [A1]: Inferencing the correct parameter per UE
  - Tries all potential parameters in the first packet, and stores the correct parameter for the subsequent packets from same RNTI

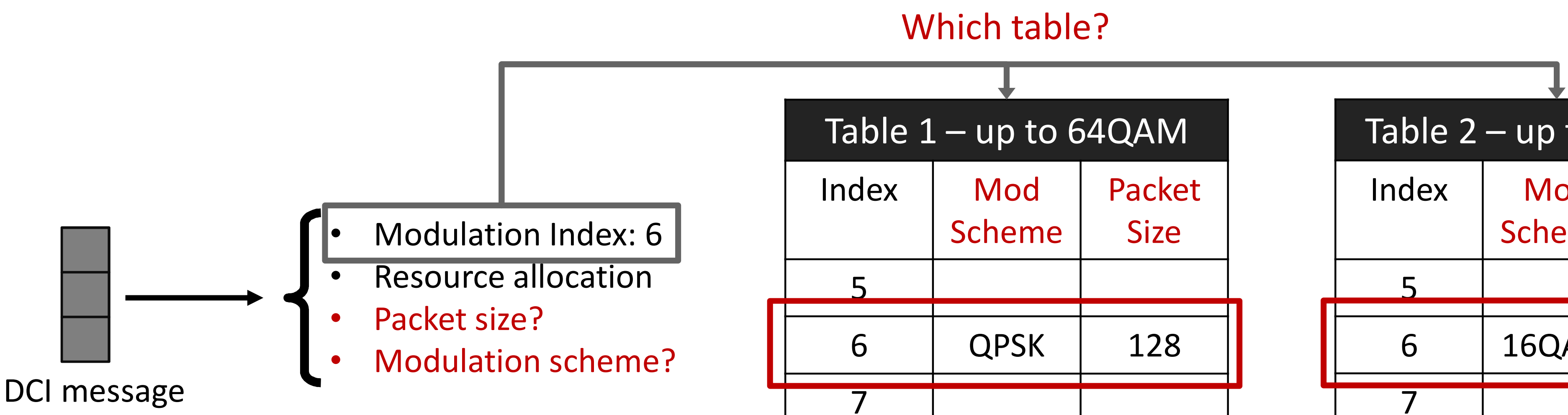
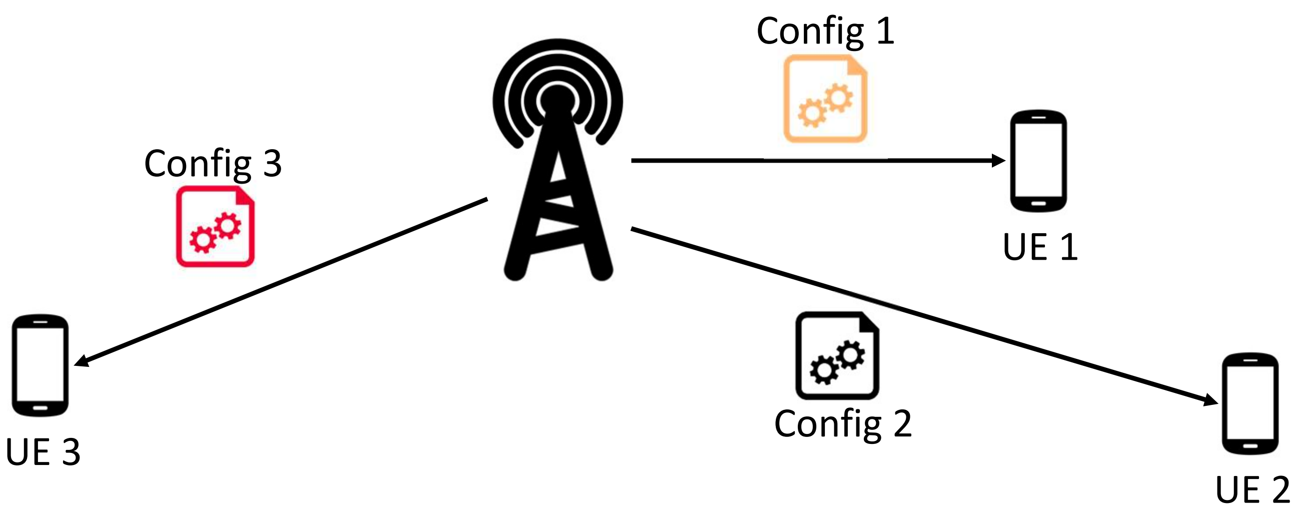


Table 1 – up to 64QAM		
Index	Mod Scheme	Packet Size
5		
6	QPSK	128
7		

Table 2 – up to 256QAM		
Index	Mod Scheme	Packet Size
5		
6	16QAM	256
7		

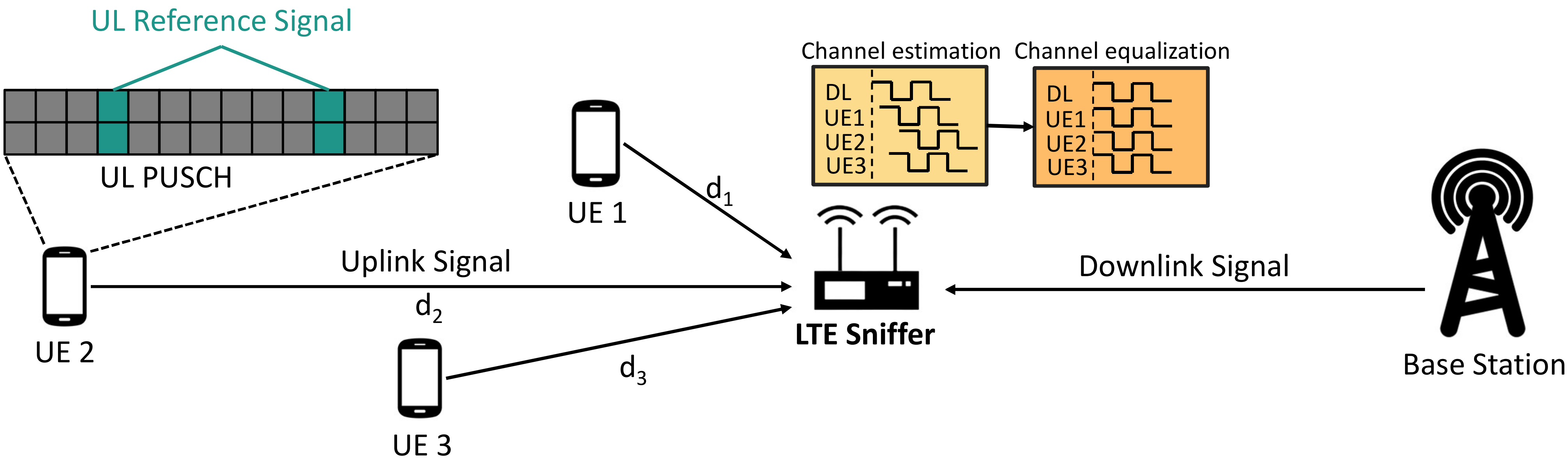
Problem and Approach [P2-A2]

- Problem [P2]: Diverse radio configuration for UEs
  - The base station assigns radio configuration differently for UEs, based on channel quality
- Approach [A2]: Adopting UE-specific configurations
  - Continuously monitoring initial radio setup procedure to obtain configuration per UE



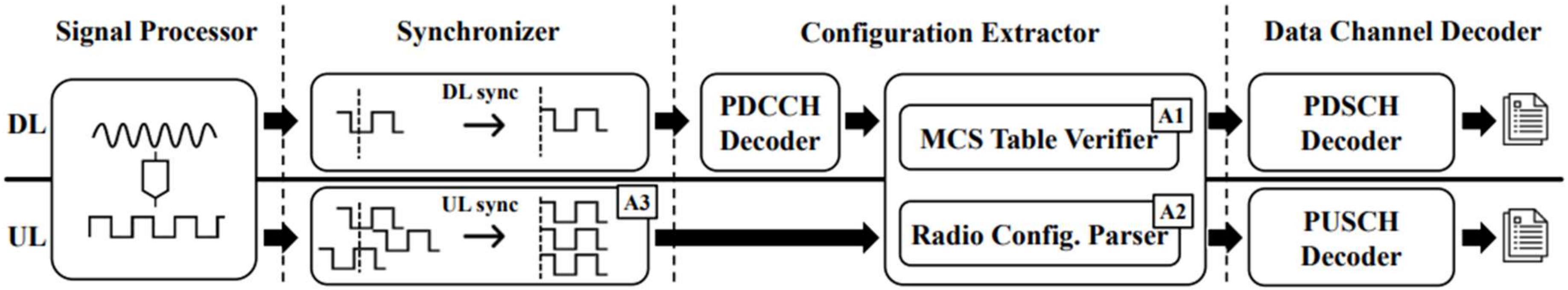
Problem and Approach [P3-A3]

- Problem [P3]: Different signal propagation delays from UEs in uplink
- Approach [A3]: Compensates for time delay for each UE
  - Utilizes uplink reference signal to calculate time delay by channel estimation
  - Applies channel equalization to compensate for the delay



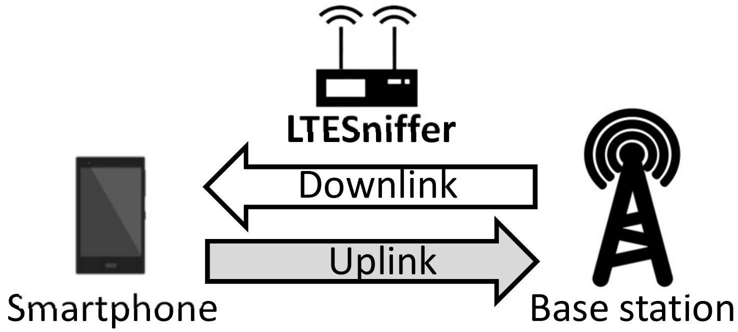
Design Overview

- ❖ Adopts behaviors of both UE/base station in downlink/uplink
- ❖ Applies three approaches [A1-A3] to the design
- ❖ Implemented on top of FALCON with the help of the srsRAN library
- ❖ C/C++



LTESniffer's Capabilities

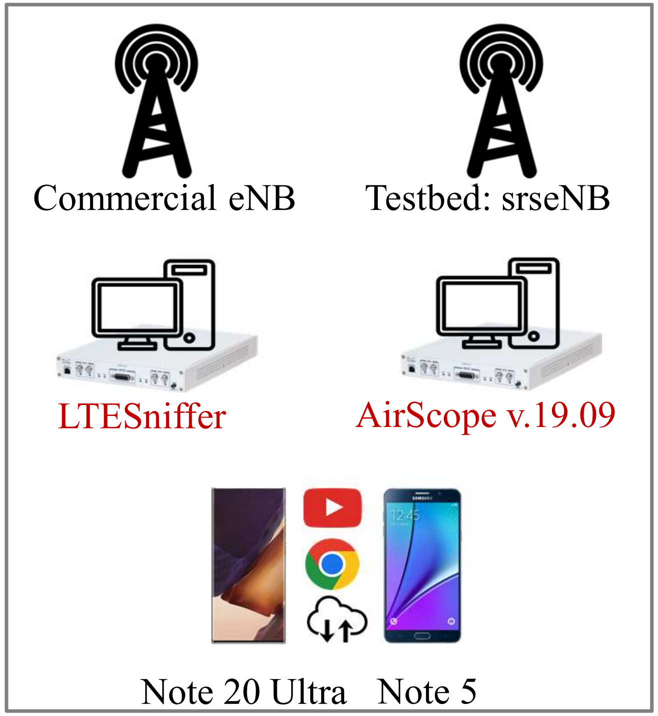
- Decoding LTE uplink-downlink control-data channels
  - Downlink: PDCCH, PDSCH (up to 256QAM)
  - Uplink: PUSCH (up to 256QAM)
- Storing decoded packets in Pcap files for further analysis
- Supporting a security API with three functions
  - 1) Identity mapping 2) IMSI collecting 3) UE Capability Profiling
- Multiple hardware options
  - For DL sniffing: most SDRs are capable
  - For UL sniffing:
    - Single USRP X310
    - Two USRP B210s with GPSDOs



SysSec
System Security Lab

Performance Evaluation

- Tested in testbed/commercial environments with two test smartphones
- Success rate = $\frac{\text{\# successfully decoded UL/DL messages}}{\text{\# detected DCI for UL/DL}}$



Success rate (%) of LTESniffer / AirScope*							
Environment	SNR (dB)	Galaxy Note 5			Galaxy Note 20 Ultra		
		Web	Video	Data	Web	Video	Data
Testbed	30	91 / 50	75 / 36	71 / 19	83 / 1	72 / 1	68 / 1
Commercial	30	92 / 92	83 / 74	61 / 40	93 / 1	70 / 1	60 / 1
	25	83 / 53	75 / 40	52 / 23	73 / 1	28 / 1	19 / 1
	22	52 / 46	31 / 19	12 / 9	44 / 1	19 / 1	5 / 1

* AirScope v. 19.09. Current v. 21.11.0 supports 256QAM in downlink with enhanced performance

Security Application

❖ LTESniffer includes an API with three functions proposed by previous works

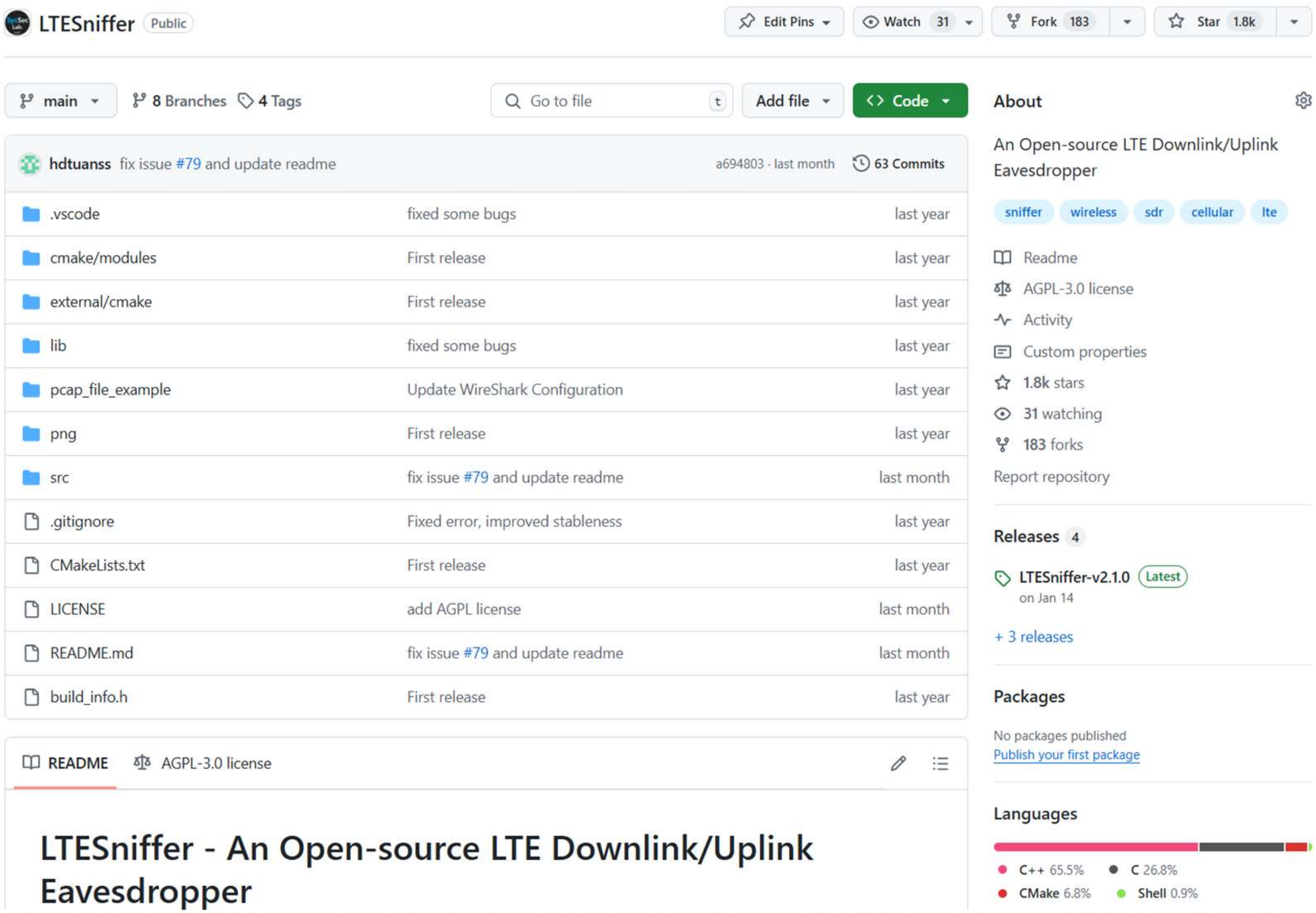
API Function	Implication
Identity mapping: RNTI - TMSI	Location tracking, website/video fingerprinting
Permanent identity (IMSI) collecting	Surveillance, privacy issues
UE capability profiling	UE model fingerprinting

SF	Detected Identity	Value	RNTI	From Message
164 -2	RandomValue	a7b611b6	70	RRC Connection Request
164 -8	Contention Resolution	a7b611b6	70	RRC Connection Setup
166 -8	TMSI	452e6684	70	Attach Request
168 -8	IMSI	901550000050918	70	Identity Response
182 -8	-	-	70	UECapability

Demo

The demonstration was conducted using an Amarisoft eNB inside a Faraday cage

LTESniffer on Github



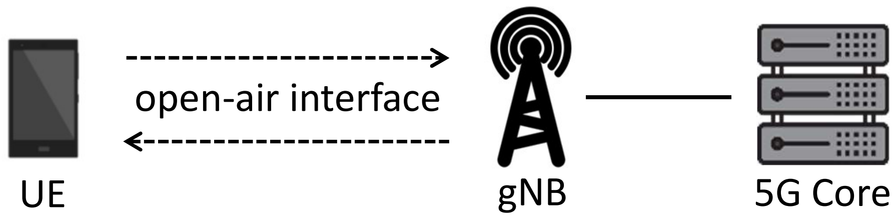
SCAN ME

https://github.com/SysSec-KAIST/LTESniffer

Developing 5G Sniffer

- 5G Overview
  - Similar architecture as LTE
  - Similar mechanism to decode DCI and data channel
    - Decode DCI first and data in PDSCH/PUSCH later
    - DCI is still unencrypted
  - ➢ **Developing 5G Sniffer is possible**

- However, there are several challenges
  - 5G physical channel is complicated
  - Unknown parameter for decoding DCI
  - Real-time decoding issue
  - Limited SDR hardware capabilities
  - Lack of supporting open-source tools

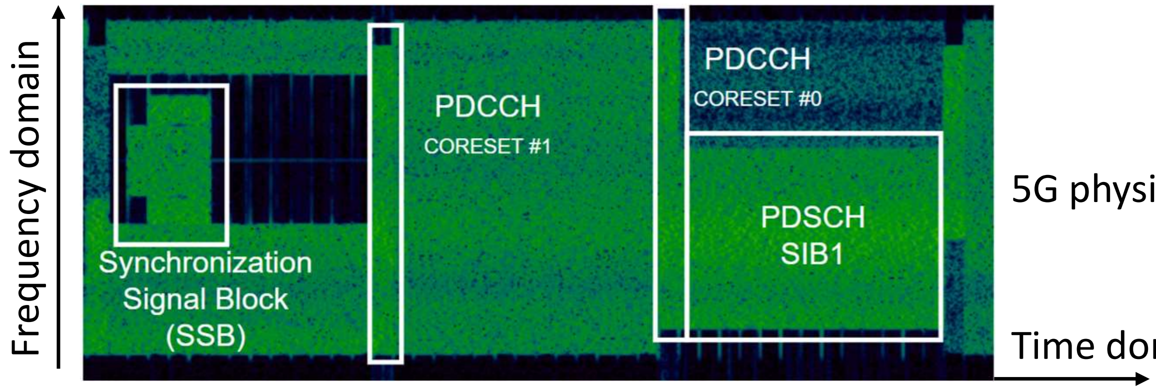


SysSec
System Security Lab

5G Sniffer Challenges

❖ (1) The complexity of new physical channel in 5G

PHY Property	LTE	5G
Subframe duration	1 ms	1/0.5/0.25 ms
Synchronization signal	Fixed at center of bandwidth	Configurable location within bandwidth
Subframe radio resources	Shared for all users	Divided into many smaller areas (bandwidth parts)
Subcarrier spacing	15 kHz	Different spacing for different areas (15,30,..,240)
PDCCH Location	Fixed, single location within subframe	Configurable, multiple locations (Coresets)
DCI Search Space	2 search spaces within 1 PDCCH area	Many search spaces within many Coresets



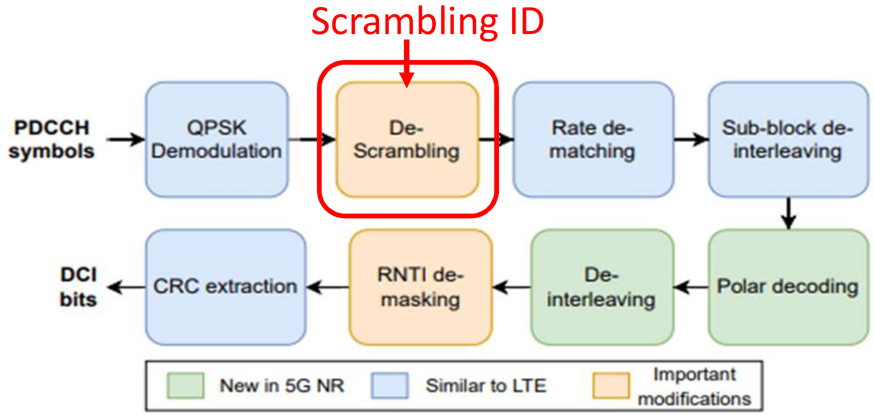
5G physical channels [1]

[1] Ludant, Norbert et. al. "From 5g sniffing to harvesting leakages of privacy-preserving messengers." *2023 IEEE S&P*

5G Sniffer Challenges

- (2) Unknown parameter for decoding procedure
  - Scrambling ID is required to decode DCI
    - In LTE, Scrambling ID is fixed to Cell ID for all UEs; in 5G, it is UE-specific parameter
  - However, this parameter is sent to UE via encrypted RRC Connection Reconfiguration msg
  - Totally, we need to brute force: 16-bit Scrambling ID in all (bandwidth parts + all locations of PDCCH + all search spaces + all DCI formats + all users) → **Huge number of attempts**

- (3) Real-time decoding issue
  - 5G peak data rate: up to 20/10 Gbps for DL/UL
  - Requires a lot of computational power
  - General-purpose CPUs might not be capable



DCI Decoding Procedure [1]

[1] Ludant, Norbert et. al. "From 5g sniffing to harvesting leakages of privacy-preserving messengers." *2023 IEEE S&P*

5G Sniffer Challenges

- (4) Limited SDR hardware capabilities
  - Limited TX-RX antennas: most SDRs do not support 4x4 MIMO
  - Limited frequency range: most SDRs do not cover FR2 (24-52 GHz)

- (5) Lack of supporting open-source tools
  - srsUE: Does not support TDD, which is main configuration in 5G
  - OpenAirInterface UE: Debugging is highly complex and challenging.

USRP X410 with 4 TX-RX antennas (~USD 30K)

srsRAN

OPEN AIR INTERFACE

Two popular open-source tools for 5G

Conclusion

- LTESniffer:
  - An open-source sniffer
  - Supports decoding uplink/downlink control/data channels
  - Supports a security API with three functions

- Developing 5G Sniffer is possible, but there are several challenges

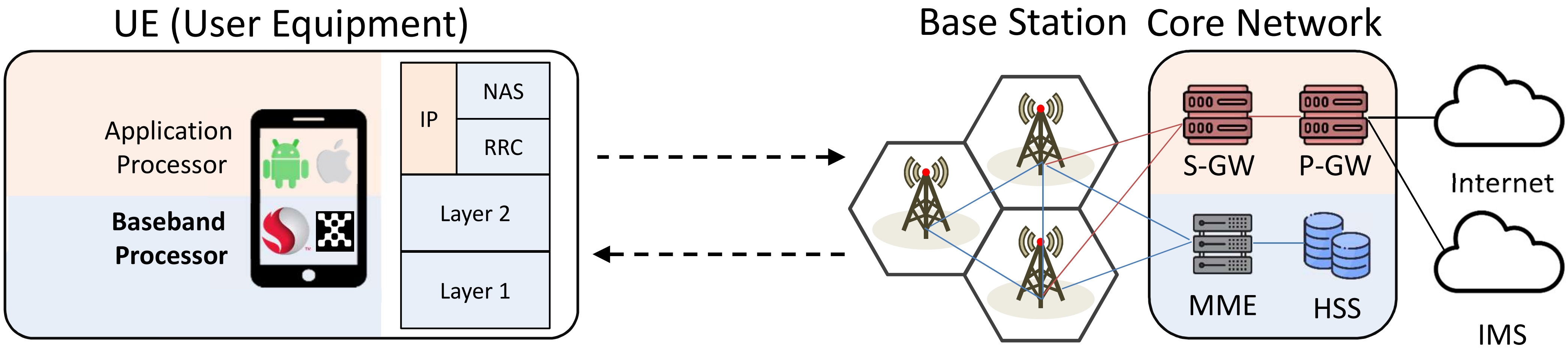
	LTEye	OWL	FALCON	AirScope	Wavejudge	LTESniffer
Open-source	✓	✓	✓	-	-	✓
DL control channel	✓	✓	✓	✓	✓	✓
DL data channel	-	-	-	✓	✓	✓
UL data channel	-	-	-	-	✓	✓
Storing pcap	-	-	-	✓	-	✓

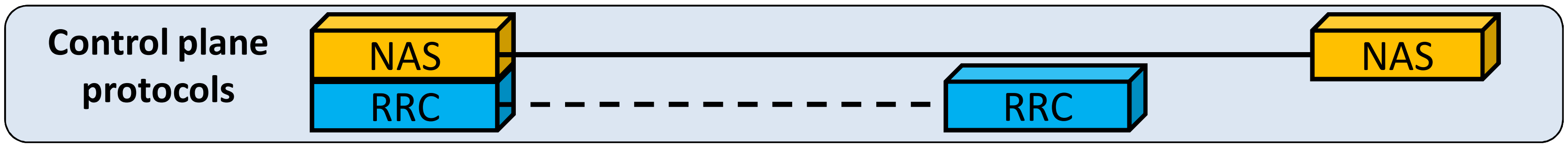
Finding Memory Bugs in the Cellular Baseband via Over-the-air Interface

CheolJun Park, Tuan Dinh Hoang
SysSec Lab, KAIST, Korea

Cellular network architecture

- Cellular service procedures are separated into **control plane** and **user plane**
  - Two main **control plane** protocols: **RRC, NAS**

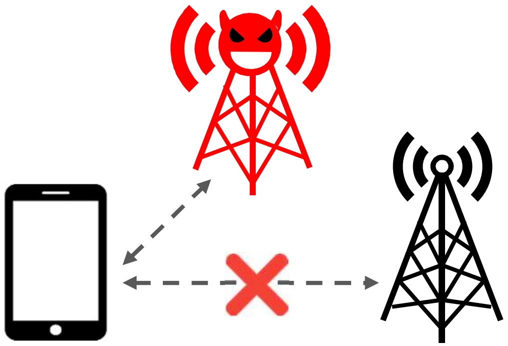




*UE: User Equipment, eNodeB: Base station, EPC: Evolved Packet Core, MME: Mobility Management Entity

Baseband is a sweet attack target

1. Over-the-air interface



2. Zero-click remote attack surface
3. Various security implications

Implications

Denial-of-Service, eavesdropping, location tracking, bidding-down cryptographic algorithms, data spoofing, potential RCE …

SysSec
System Security Lab

Memory bugs in cellular basebands

- Potential RCE
  - C/C++ codebase
  - Support 2G — 5G
  - Shared memory architecture, IPC
- Many offensive researchers/companies
  - TASZK security lab, Comsecuris, Tencent KEEN lab, Google Project Zero, …

E2E exploit on Huawei Smartphone (Black Hat USA 2018)

Attentions on modem security issues (Google Project Zero 2023)

0-click RCE on Tesla via a cellular modem (Pwn2Own Automotive 2024)

Previous works (public)

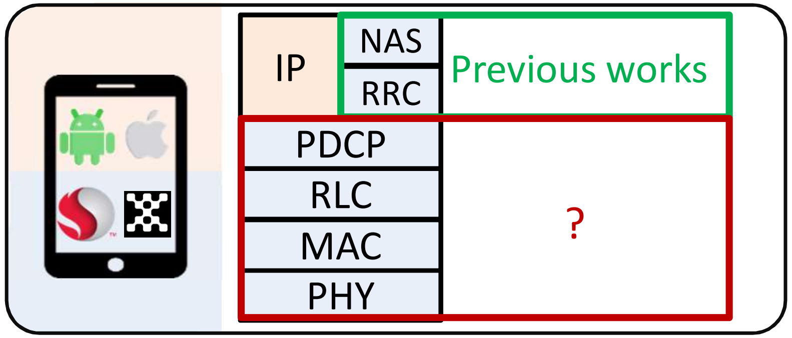
- Reverse-engineering efforts
  - (2010, Hack.lu) Ralf-Philipp Weinmann (Qualcomm, Intel)
  - (2016, Comsecuris) Nico Golde and Daniel Komaromy (Shannon)
  - (2018, OPCDE) Amat Cama (Shannon)
  - (2018, BlackHat) Marco Grassi, Muqing liu, Tianyi Xie (Huawei)
  - (2018, Comsecuris) Nico Golde (Intel)
  - (2020, Blog) Frederic Basse (Shannon)
  - (2020, OffensiveCon) Marco Grassi and Kira (MediaTek)
  - (2021, NDSS) Eunsoo Kim and Dongkwan Kim (Shannon)
  - (2023, OffensivCon) Amat cama (Intel)
  - (2023, OffensivCon) Daniel Komaromy (Shannon)
- Emulation-based approach
  - (2020, WiSec) Dominik Maier et al. (MediaTek)
  - (2022, NDSS) Grant Hernandez et al. (Shannon, MediaTek)
- Over-the-air fuzzing
  - (2011, USENIX Security) Collin Mulliner, Nico Golde (GSM feature phones)
  - (2021, WiMob) Srinath Potnuru and Prajwol Kumar Nakarmi (open-source baseband)
  - (2022, STISC) Hongxin Wang et al. (open-source baseband)
  - (2024) Matheus E. Garbelini et al. (Qualcomm, MediaTek)

Gaps in previous works

- Mostly targets **2G/3G**, and requires **manual efforts**
- {Emulation + AFL} suffered from **coverage**
- Recent works support **Shannon (Samsung) and MediaTek**
  - Qualcomm?
- Focused on Layer 3 protocols (i.e. NAS, RRC)
  - How about lower layers (PHY, MAC, RLC, PDCP)?

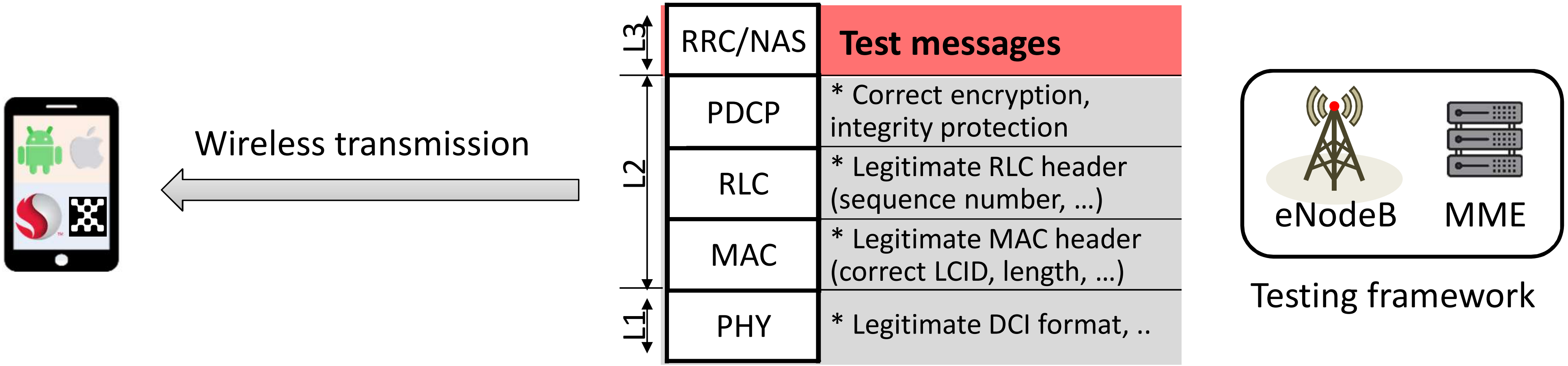
(2010) “Layer 1 not fruitful, Layer 2 messages to short, …”
(2012) “Below layer 3, there usually is little potential for exploitable memory corruptions”

UE (User Equipment)



Approach

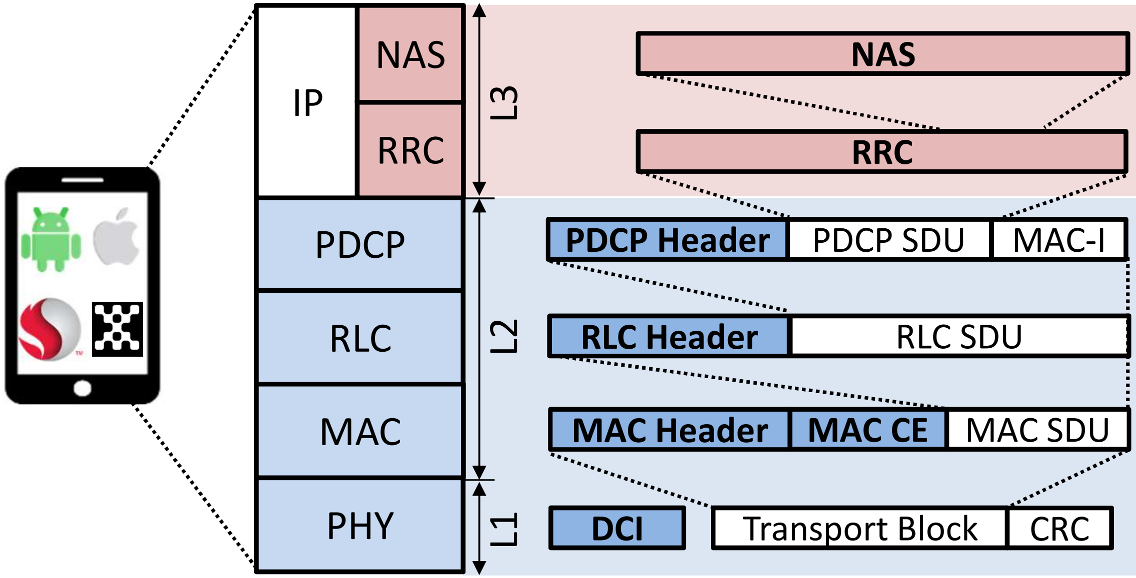
- Build testing framework using over-the-air interface
  - (☺) Applicability: can send test messages for L1~L3, regardless of the baseband vendor
  - (☺) By using legitimate messages, we can move UE's state
  - (☹) Due to its nature, hard to send a large number of test messages
  - (☹) Black-box



Example: targeting L3 messages

Goal of this work

- Finding memory bugs on COTS cellular basebands in both layer 3 and lower layers
  - Layer 3 (NAS, RRC) supports a lot of different message types / fields
    - E.g. RRC defines > 900 IEs (information elements) that contain > 4k fields
  - However, lower layers (PDCP, RLC, MAC, PHY) also carry various fields, header formats, and control information
    - More functionalities from 4G



For a given limited number of trials (slow OTA), how to generate test messages for testing enormous messages/fields in specification?

What lower-layer protocol features need to be tested and how?

SysSec
System Security Lab

Challenge 1: test case generation

- Specification defines a lot of messages and optional fields
  - Mutating commercial log is not effective
    - Many messages/fields are almost never used in the real world
  - Why don't we just use AFL?
    - Leveraging code coverage is hard

COTS baseband (ours)	No source code (proprietary)
Open-source baseband	Only supports a few essential messages
Baseband emulation	Limited code coverage (1% - 3.5%) as the state-of-the-art can't explore states

    - Most random packets (+mutations) are rejected in early stage
  - Meanwhile, the number of trial in OTA is limited

Approach 1: grammar-guided generation

- Leverage the protocol specification to obtain the grammar-coverage
  - Baseband implements decoder/handler for every protocol definition in the spec

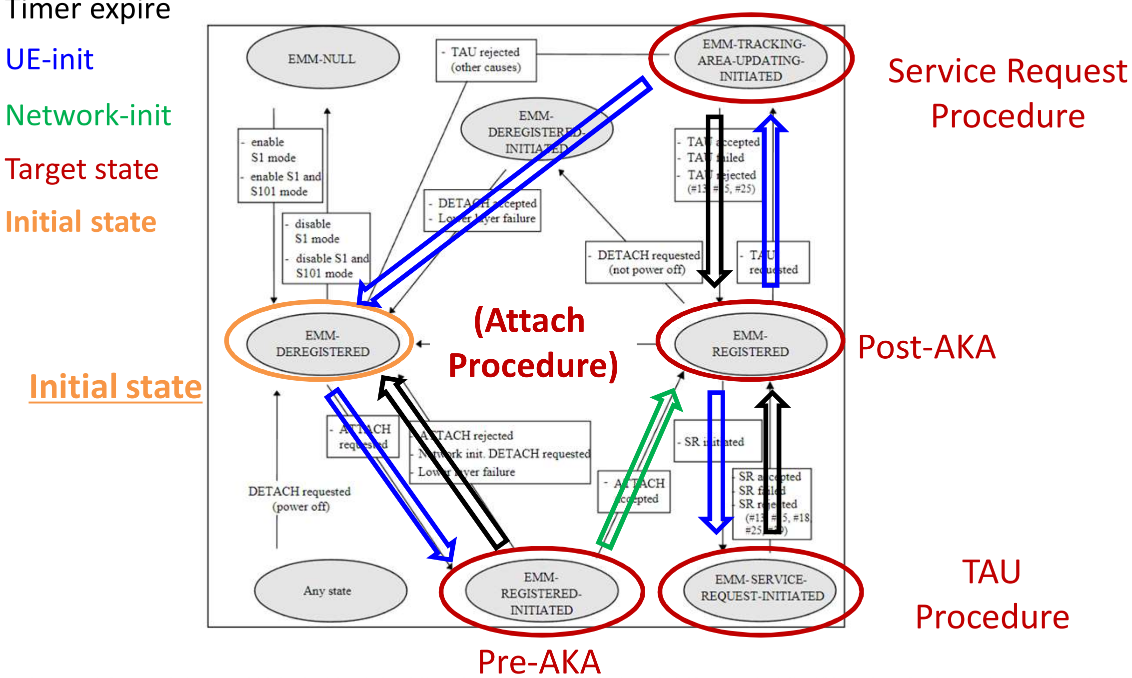
- First, generate legitimate packets that cover the message structures in the specification
  - Layer3 (NAS, RRC): specification defines huge number of messages/fields
  - Lower Layers (PHY, MAC, RLC, PDCP): Afawk, no one tested here
    - Their structures: defined as tables + natural language descriptions

- Then, grammar-aware mutation
  - Random mutations → early rejected + no coverage feedback
  - Many parts of the packets are not interested in terms of memory corruption

Challenge 2: stateful behavior of baseband

- The baseband is stateful and **initiates most state transitions**
  - It determines whether to connect or transition between states

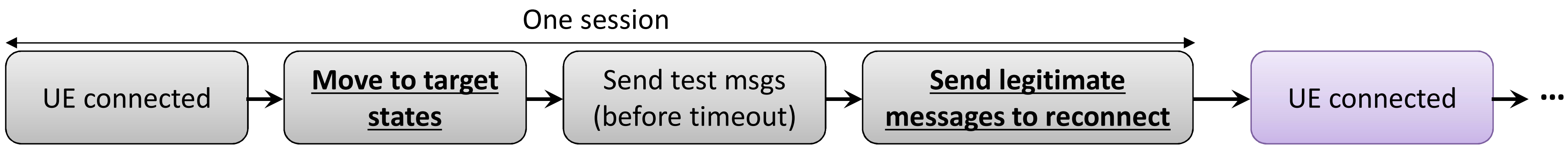
Timer expire
UE-init
Network-init
Target state
Initial state



State	Timer	Value	State transition
Pre-AKA	T3410	15s	**EMM-REGISTERED-INITIATED → EMM-DEREGISTERED**
Post-AKA	-	-	**-**
Service Request	T3417	5s	**EMM-SERVICE-REQUEST-INITIATED → EMM-REGISTERED**
TAU Request	T3430	15s	**EMM-TRACKING-AREA-UPDATING-INITIATED → EMM-REGISTERED**

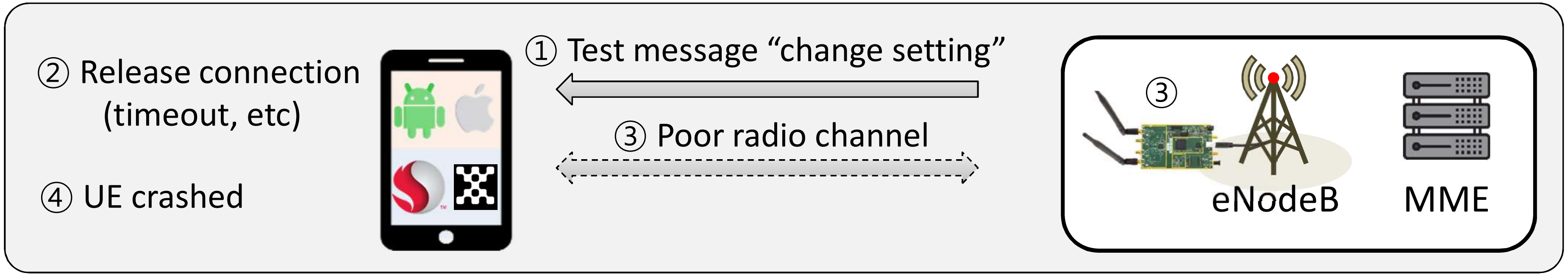
Approach 2

- Find network-side state transition logic through specification analysis
  - Requirement
    - i) Network-side mechanism that ii) instantly trigger UE-side state transition
  - Several implementation and experimentation efforts
    - Open-source didn't support Detach, TAU and SR handling logic
    - Exynos had two implementation flaws (wrong state transition)
  - Batch testing



Challenge 3: fragile radio connection

❖ UE hangs or disconnects due to various reasons
1. Our test message may alter the radio configuration to an incorrect settings
2. UE may release the connection by itself
3. Connection maybe dropped out
   - Poor radio channel at that moment
   - Hardware (SDR) failure
4. UE crashed



SysSec
System Security Lab

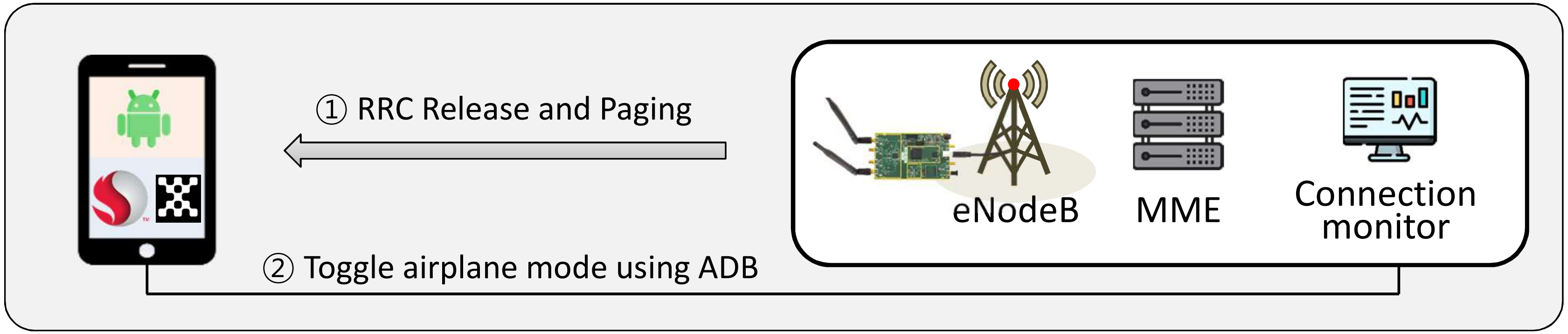
Approach 3

- When UE is disconnected or do not respond
  – Reconnect UE using two methods

Step 1. Use cellular protocol messages to make UE to connect again

- However, UE may ignore any further messages

Step 2. When UE does not reconnect after **Step 1**, use ADB to toggle airplane mode



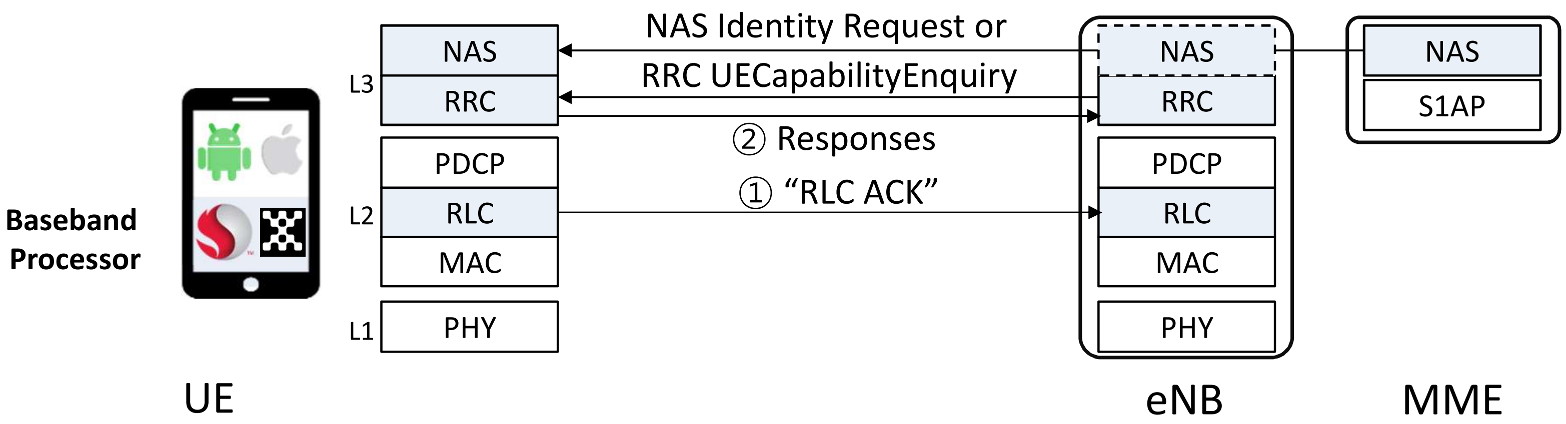
Challenge 4: oracle for detecting crashes

- Limited oracles for detecting crashes
  - Previous works used i) memory sanitizer (emulation) or ii) crash log at the terminal (open-source basebands)
- Prior methods to confirm crash after replay
  - Checking the signal bar or connectivity, manufacturer's debug mode, …

Target	Impact	Work	Validation w/ 1-day	False positives	Automation?
Visual feedback	Signal bar disappear	NDSS'22	☺	☹	☹
Cellular connection	Loose connectivity	Security'11, 23	☺	☹	☹
ADB log	"CP Crash" log	NDSS'22	☹	☺	☺
Bluetooth connection	Bluetooth dead	Security'11	☹	☹	☺
Manufacturer's debug mode	Kernel panic	WiSec'20, Security'23	☺	☺	☹

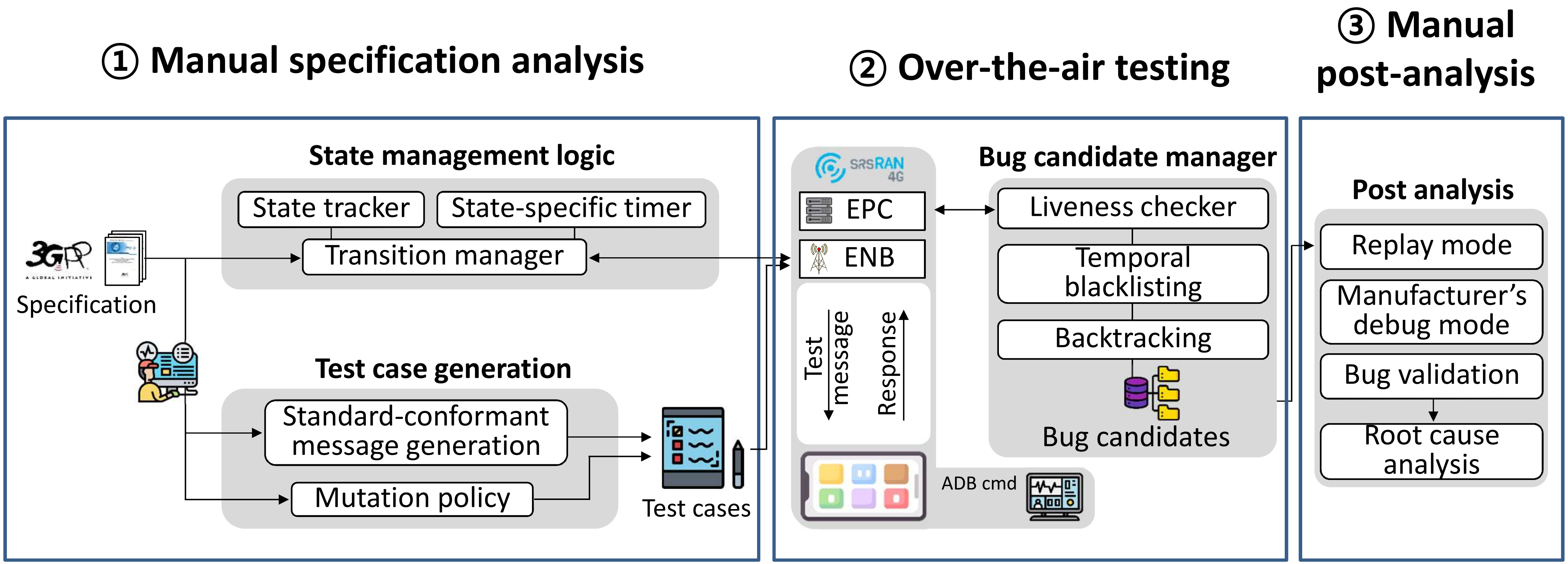
Approach 4

- Passive and active liveness detection based on cellular protocol
  - P: Layer2 RLC ACK
  - A: Layer3 RRC / NAS message that
    - i) Does not change the state of the UE and ii) UE always respond (in all states)



- For lower layers: Monitor ADB radio logcat output
  - Separate thread for ADB to eliminate performance issues
  - Detect string: "Modem Reset", "RADIO_OFF_OR_UNAVAILABLE"

System overview



SysSec System Security Lab

Result

- Tested devices from 3 major baseband vendors (Qualcomm, Exynos, and MediaTek)
  - Layer 3 (NAS, RRC): **6** cellular devices
  - Lower layers (PHY, MAC, RLC, PDCP): **8** cellular devices

- Discovered implementation flaws
  - Layer 3: **7** 0-day and **3** 1-day bugs from MediaTek and Exynos basebands
  - Lower layers: **9** 0-day bugs from Qualcomm, MediaTek, and Exynos basebands

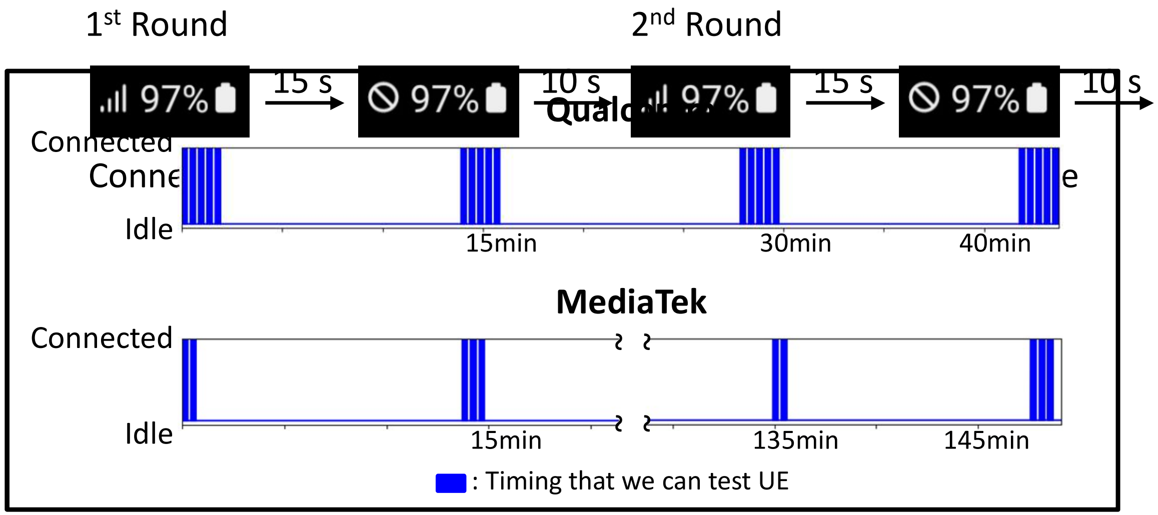
Thank you!

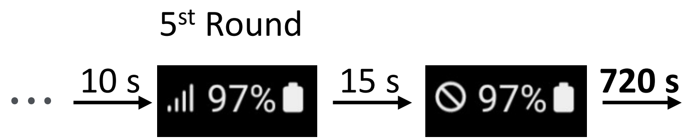
Questions?

- You can reach me:
  - Tuan D. Hoang: tuan.hoangdinh@kaist.ac.kr (@hdtuanss)
- KAIST SysSec Lab (Prof. Yongdae Kim)
  - https://www.syssec.kr/

Challenge 2: stateful behavior of baseband

- Example: testing baseband at "pre-AKA" state
  - When the timer expires 5 times, UE does not reconnect for a long time
  - E.g. Qualcomm: 15 sec × 5 = 75 sec (connected time) + 760 sec (idle time) → 91.02% idle time
  - Worst case: 99.07% idle time (MediaTek)





	Period	Down time	Ratio
Exynos	~1160s	~830s	~87.07%
Qualcomm	835s	760s	91.02%
MediaTek	8065s	7990s	99.07%
HiSilicon	~835s	~760s	~ 91.02%

UE's connection status in a normal testing scenario