[EE515] Security of Emerging Systems, 24 Fall

12:00 - 13:05

Finding Implementation Vulnerabilities in Cellular Basebands

CheolJun Park

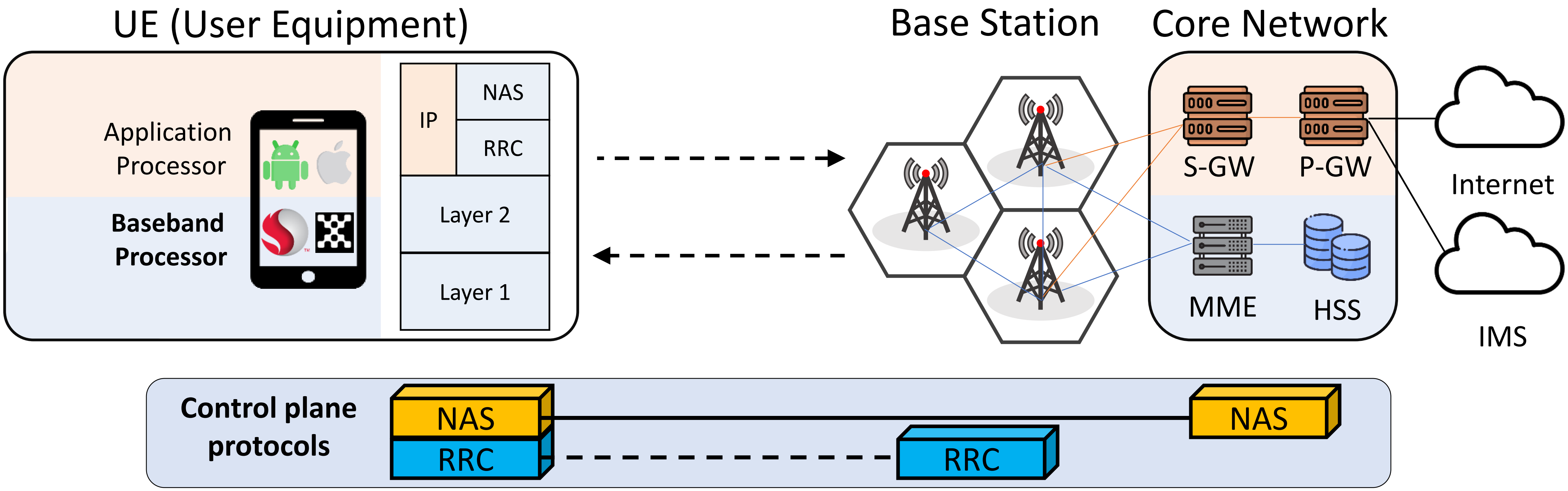
School of Computing, KHU
<cheoljunp@khu.ac.kr>

Nov 20, 2024

경희대학교
KYUNG HEE UNIVERSITY

Cellular network architecture

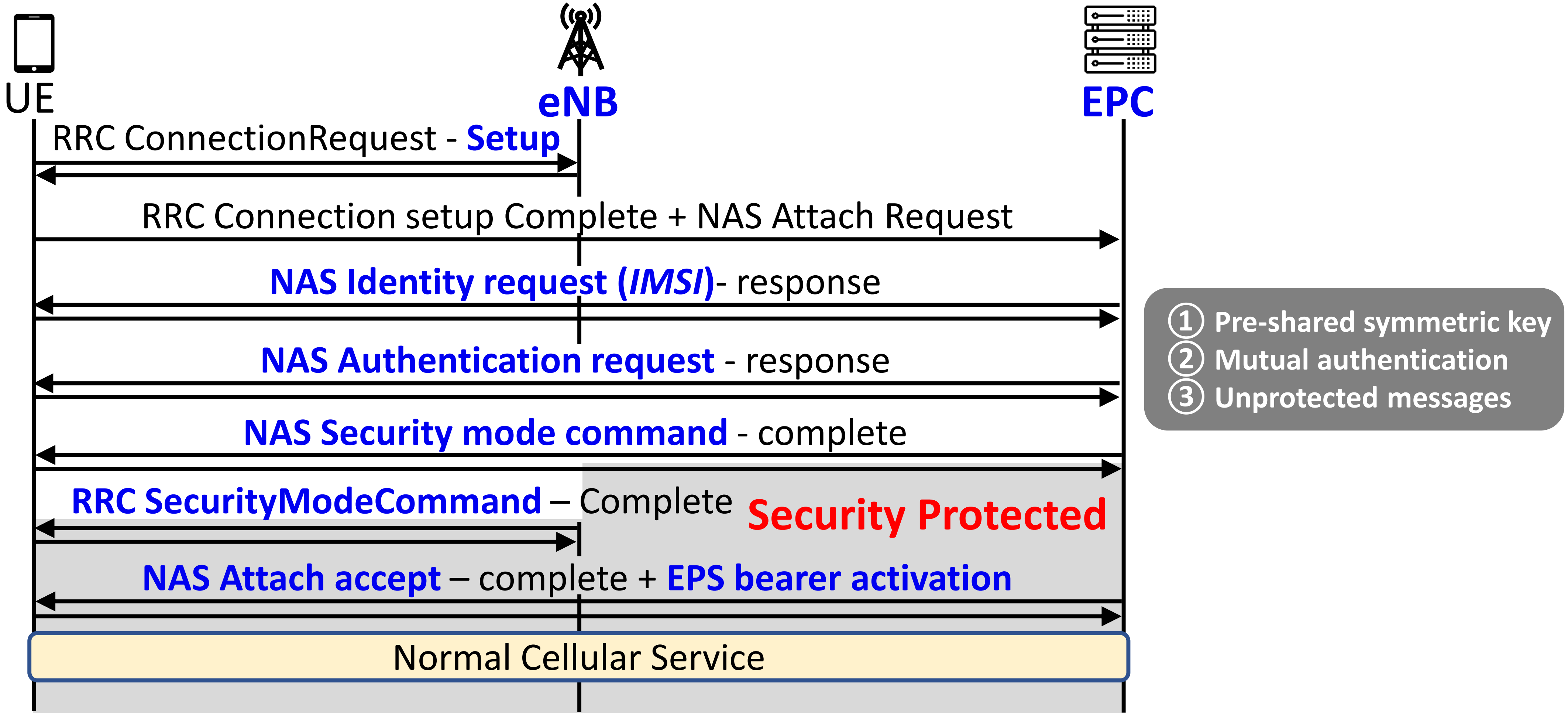
- Cellular service procedures are separated into **control plane** and **user plane**
  - Two main **control plane** protocols: **RRC, NAS**



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

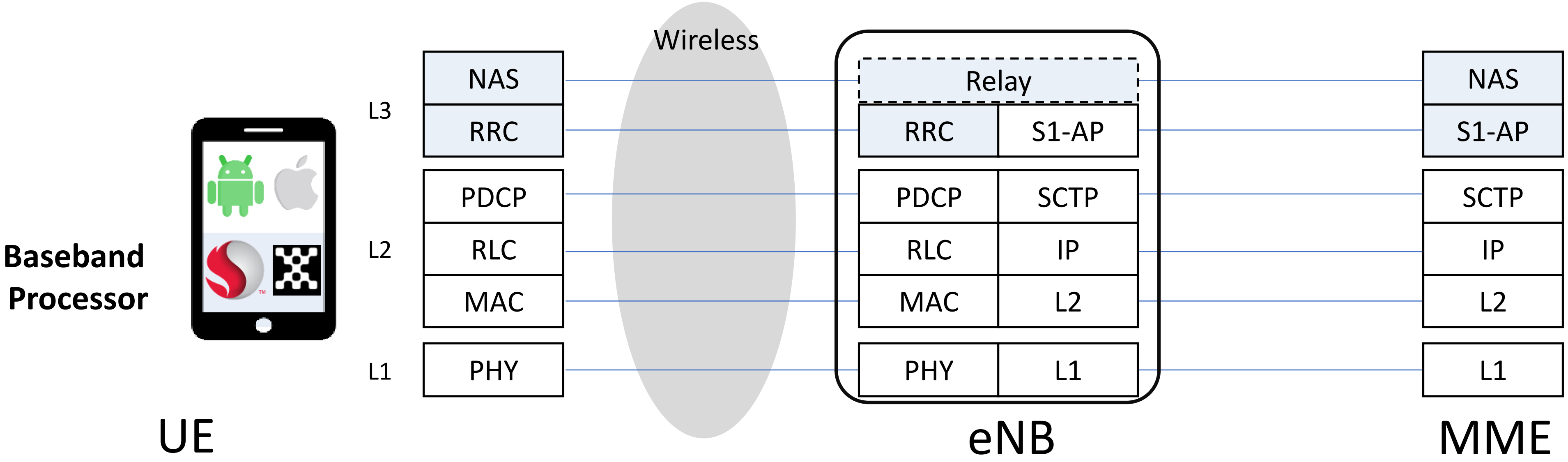
LTE attach procedure

❖ UE should send **security-sensitive** data **after security activation**



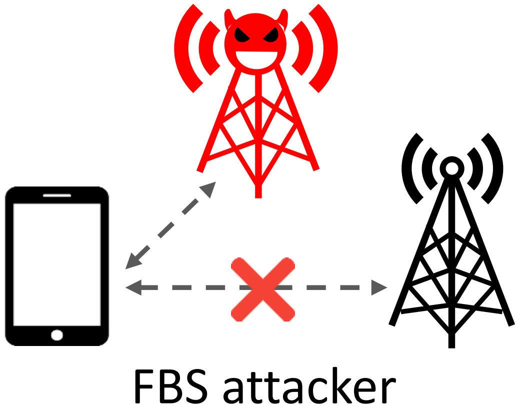
LTE protocol stack

- Each layer offers core control operations
  - RRC: Radio connection management, handover, ..
  - NAS: Authentication, key agreement, ..
  - PDCP: Encryption, integrity, replay protection
  - RLC: Acknowledgement, segmentation
  - MAC: Packet scheduling, …



Baseband (cellular modem) is a sweet attack target

1. Over-the-air interface



2. Zero-click remote attack surface
3. Unprotected certain procedures
4. Various security implications

Implications

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

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, …

Call hijacking through RCE on Galaxy series (Mobile Pwn2Own 2016)

E2E exploit on Huawei Smartphone (Black Hat USA 2018)


The Hacker News | News18 | BleepingComputer

Google Uncovers 18 Severe Security Vulnerabilities in Samsung Exynos Chips

Attentions on modem security issues (Google Project Zero 2023)

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

Security problems in baseband (UE)

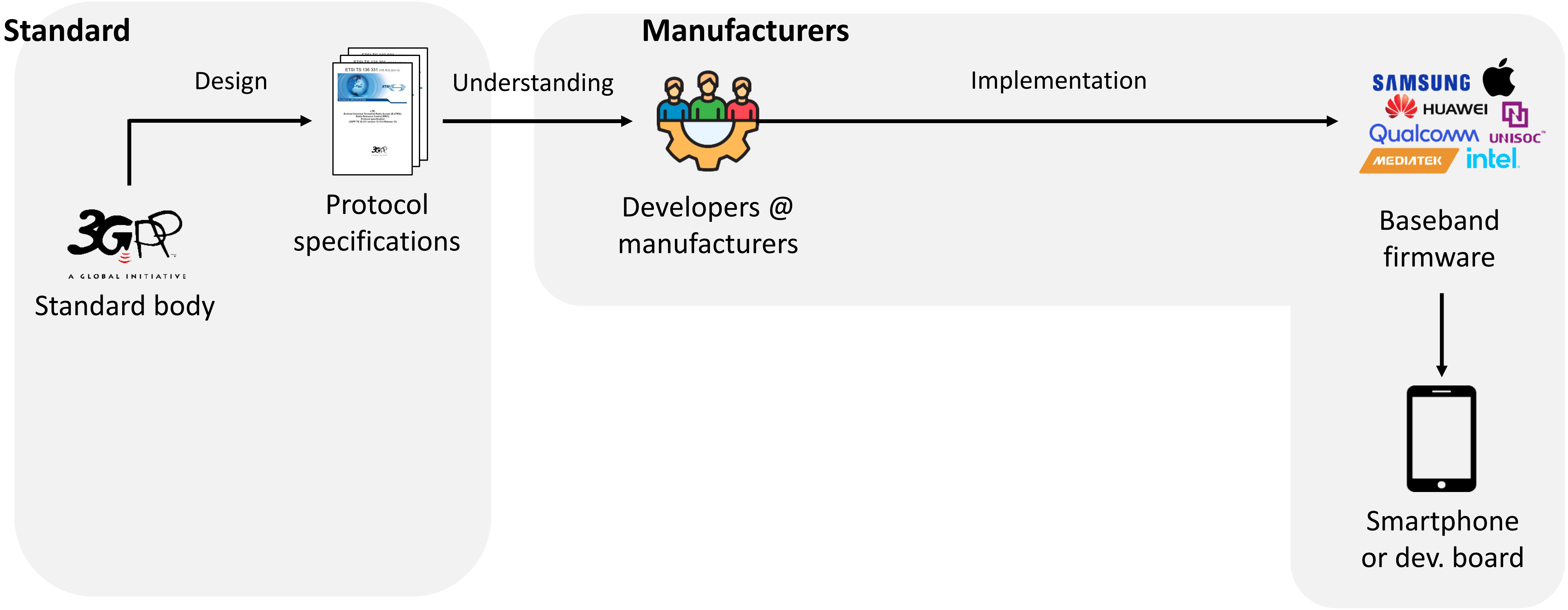
❖ Three types of LTE vulnerabilities

Baseband

Standard Body	Manufacturers	Network Operators
1. Design (standard) vul.	**2. Implementation vul.**	**3. Operational vul.**
- Insecure design by standard body - Logical bugs	- Mistakes by developers - Logical (non-standard-conformant) bugs, memory bugs	- Misconfigurations @ MNO - Under-specification, mistake ..

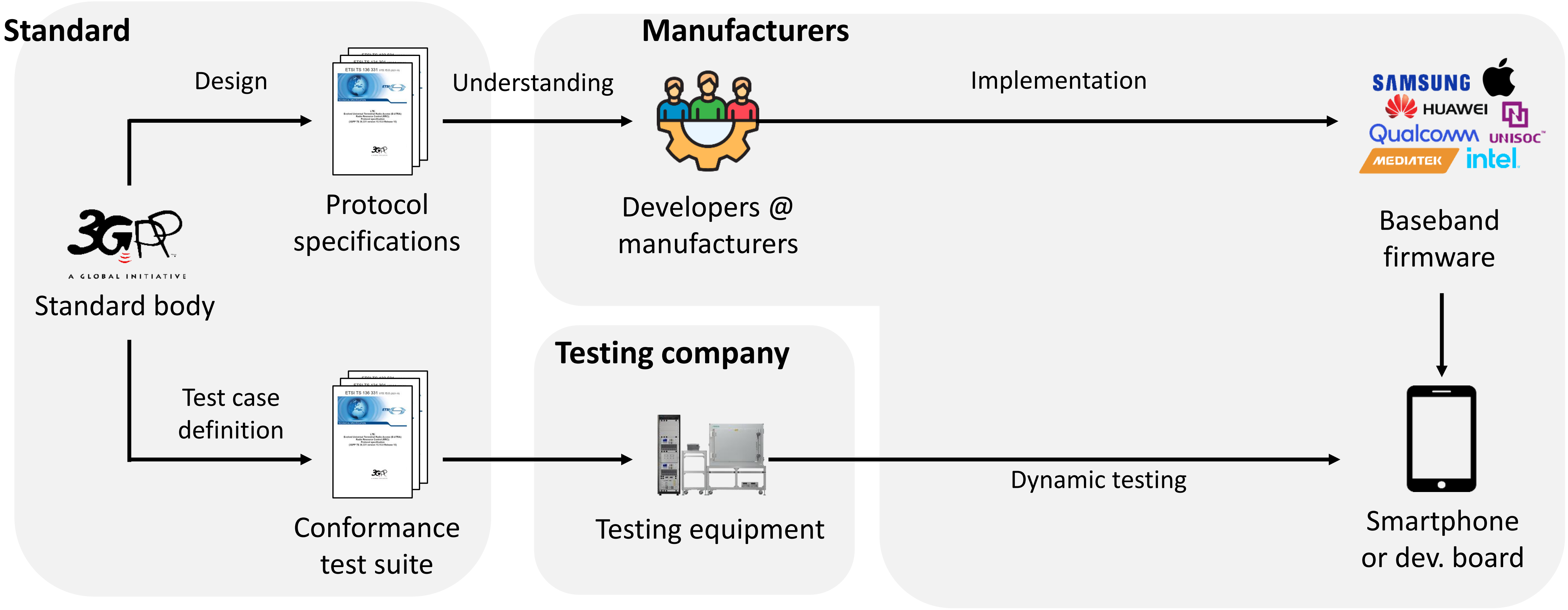
Security problems in baseband (UE)

❖ Baseband development process



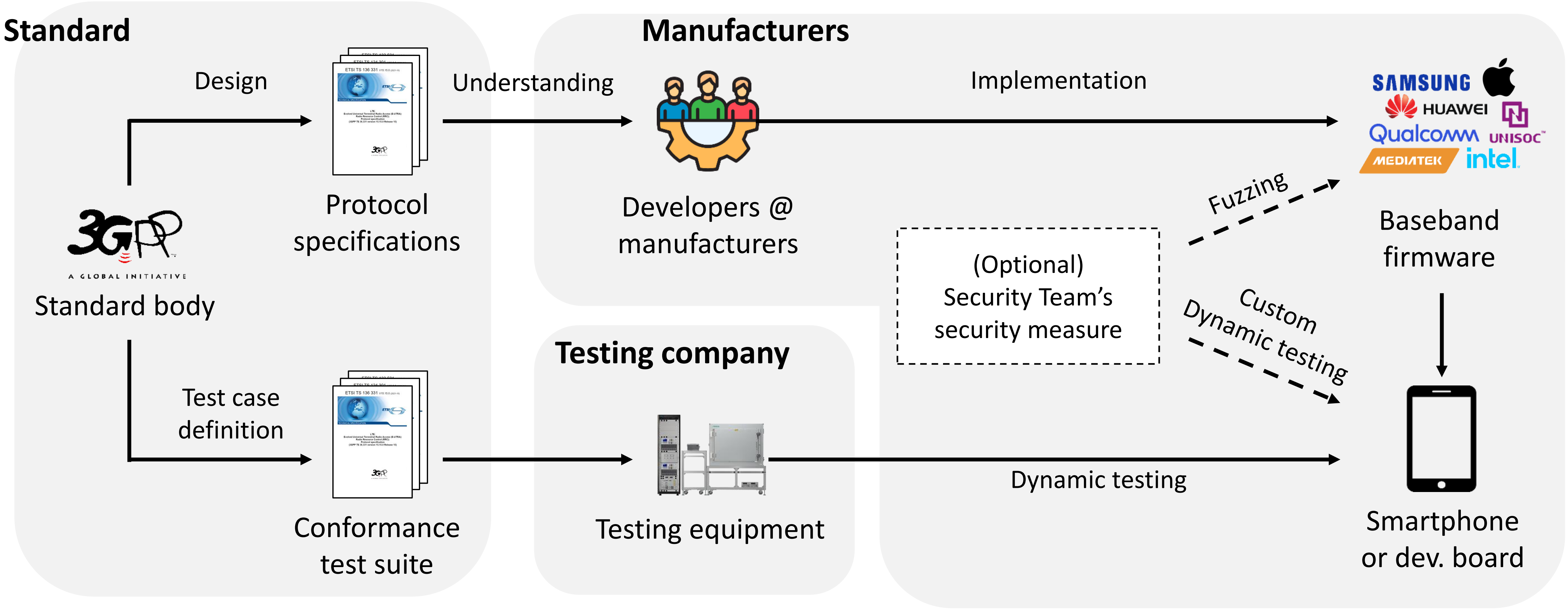
Security problems in baseband (UE)

❖ Baseband development process



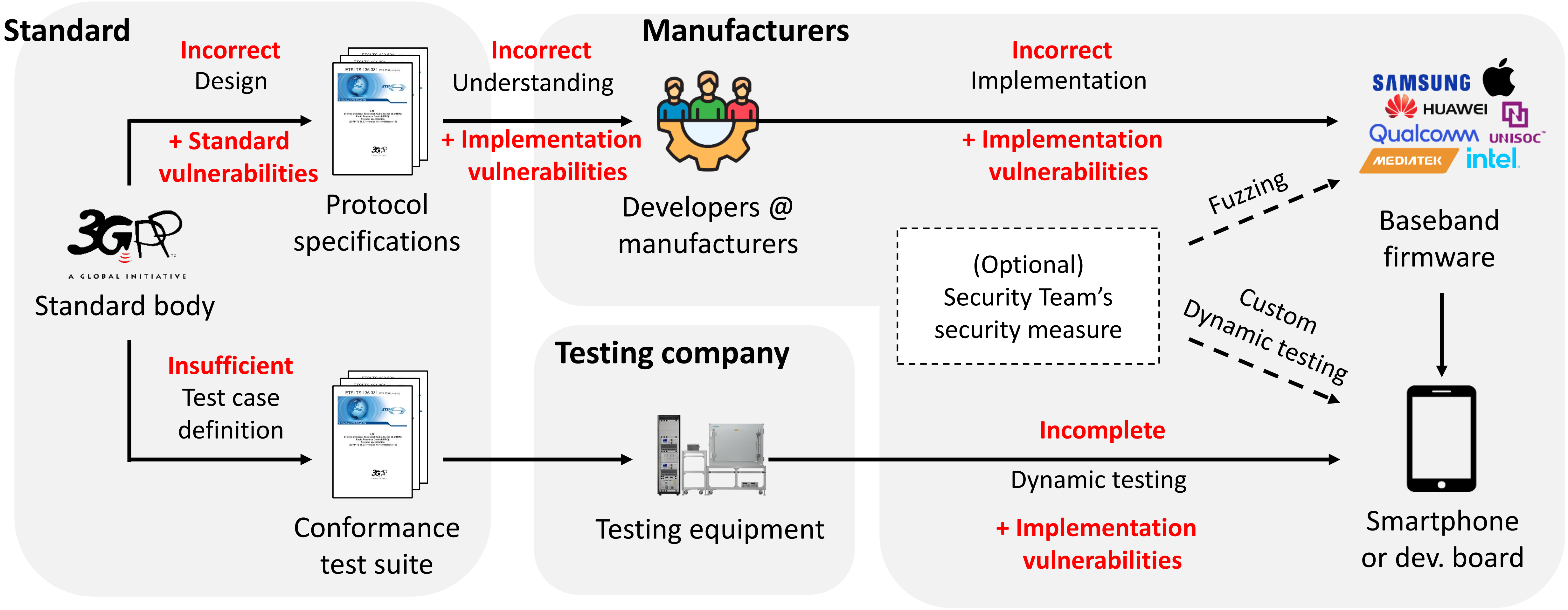
Security problems in baseband (UE)

❖ Baseband development process



Security problems in baseband (UE)

❖ Secure specification **does not necessarily lead** to secure implementations



Limitations of UE security testing (industry)

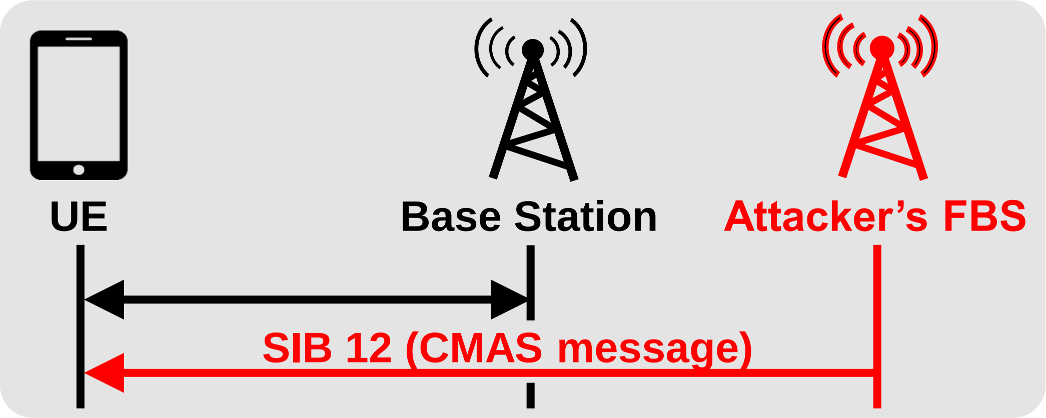
- UE conformance specification
  - Mostly positive test cases: Check if **valid messages** are correctly handled
  - Negative test cases? : Check if **invalid or prohibited messages** are appropriately handled
  - Among 993 test scenarios in conformance spec, **only 14 cases are negative**.[1]

- Internal solutions (of manufacturers' security team)
  - **Unknown**, and definitely insufficient
    - As evident by continuously reported bugs
  - **Not applicable** for every baseband
    - OEM firmware

[1] 36.523, v15.5.0

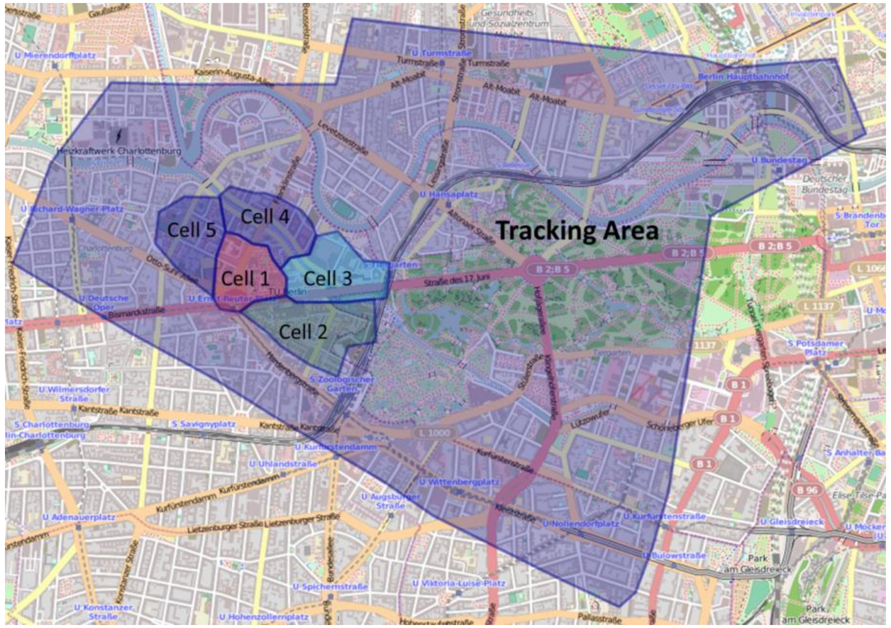
Attacks in LTE (Design Vul.)

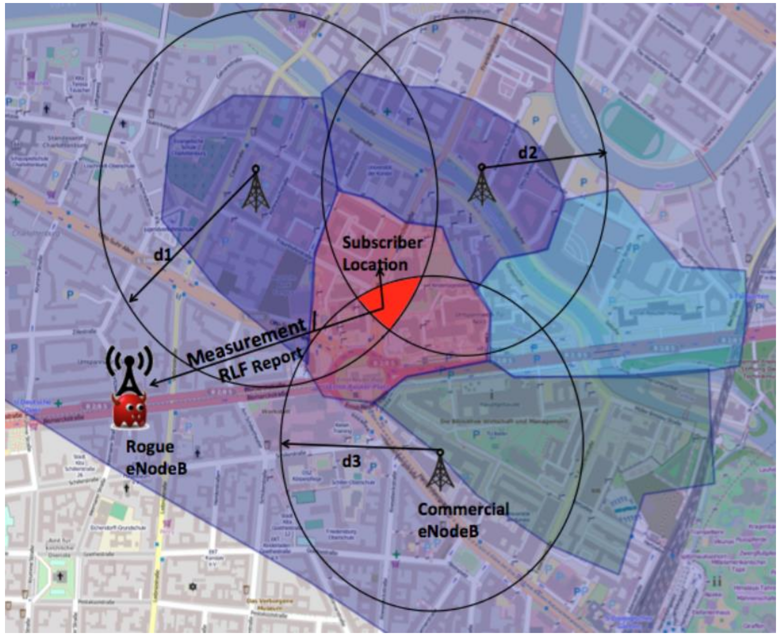
- Fake emergency alert injection
  - Can attack even UEs connected to the legitimate base station

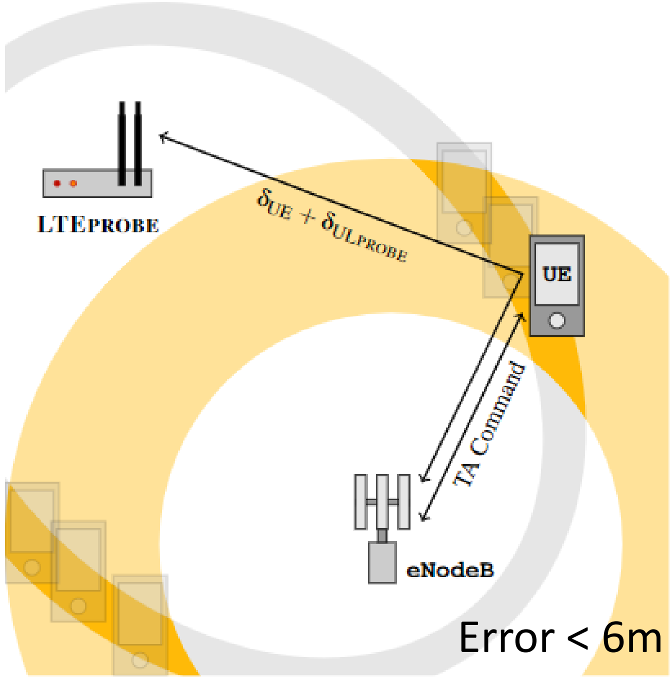


Attacks in LTE (Design/Implementation Vul.)

- Location tracking
  - Base station-level tracking (paging) [2]
  - Trilateration
  - Time of arrival [3]



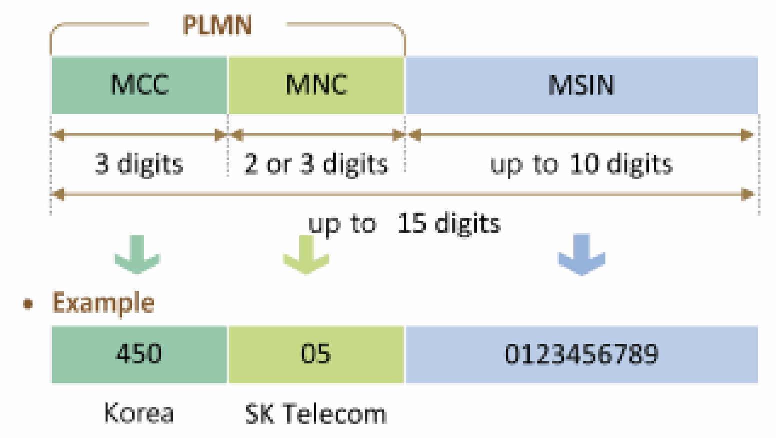




[2] Practical Attacks Against Privacy and Availability in 4G/LTE Mobile Communication Systems
[3] LTRACK: Stealthy Tracking of Mobile Phones in LTE

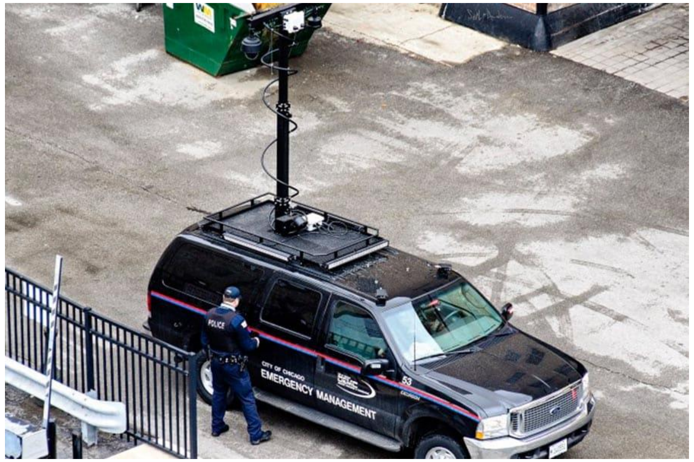
Attacks in LTE (Design Vul.)

- Identity tracking
  - IMSI-catcher
  - MSISDN (phone number) - IMSI mapping
  - RNTI-GUTI mapping
  - RNTI-IMSI mapping …



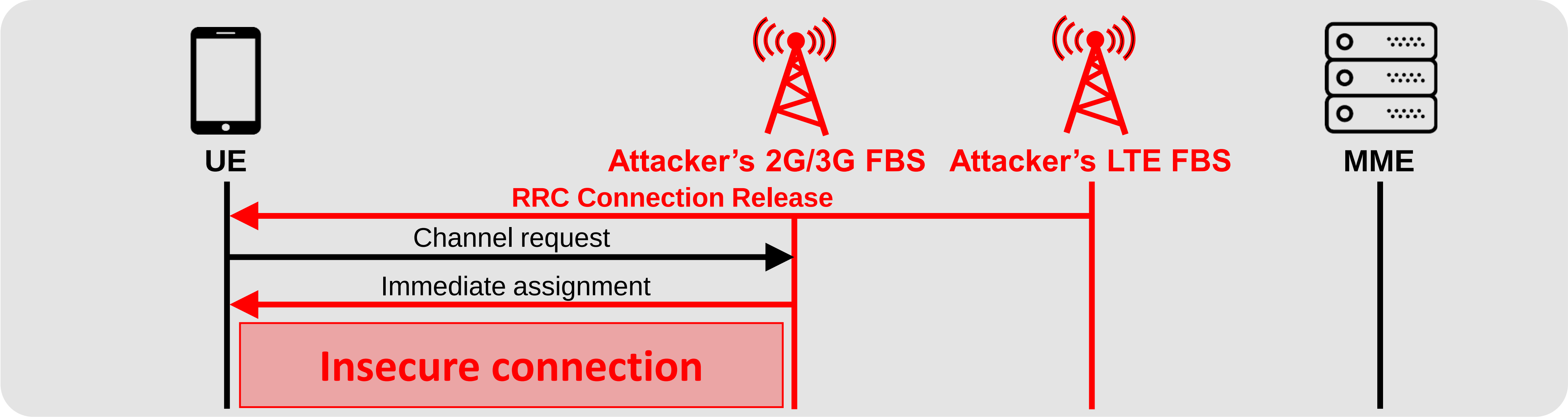
IMSI

IMSI: a USIM's unique ID
GUTI: a USIM's temporary ID
MSISDN: phone number
RNTI: a UE's ID @ radio layer



Attacks in LTE (Design Vul.)

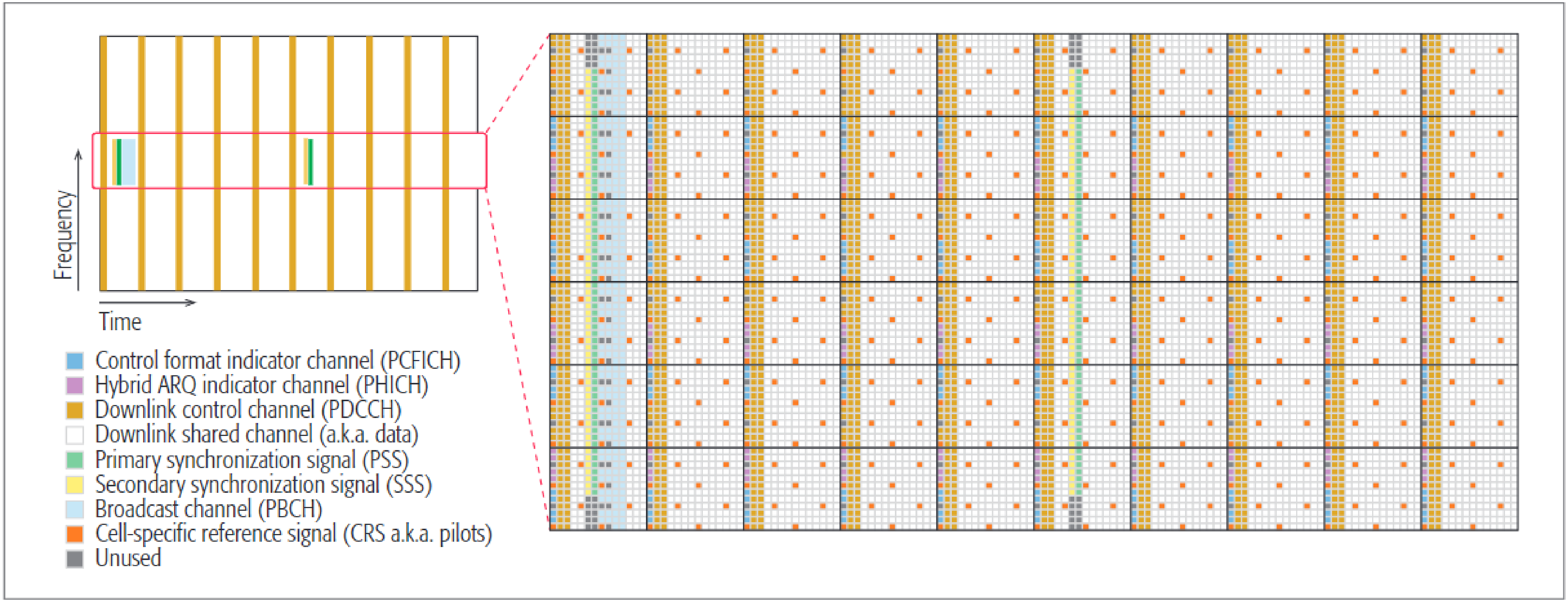
- Network downgrading
  - Downgrade to 2G or 3G
  - 2G (GSM)
    - Lack of mutual authentication
    - Use no (A5/0) or weak encryption algorithm (A5/1, A5/2)



Attacks in LTE (Design Vul.)

❖ Denial-of-service

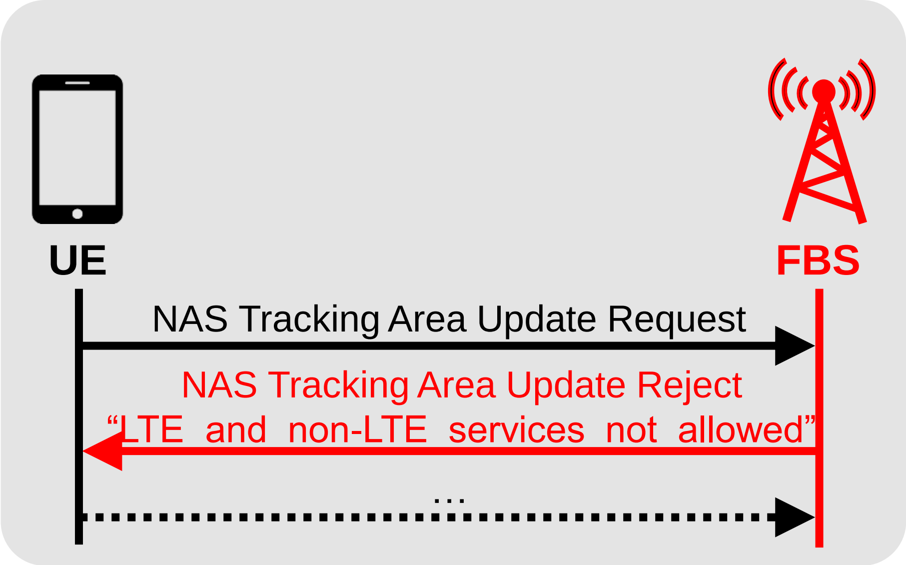
– Smart jamming (Protocol-aware selective jamming) [4]

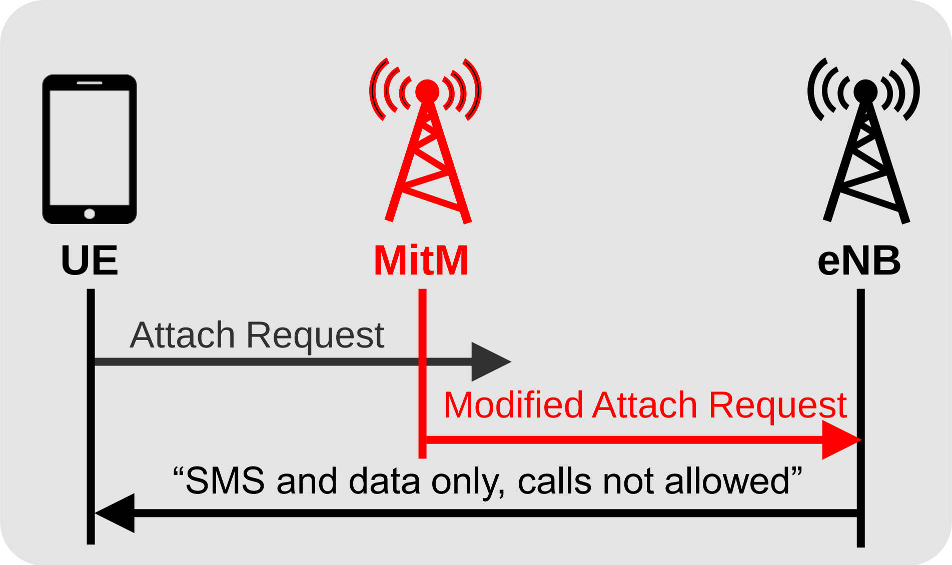


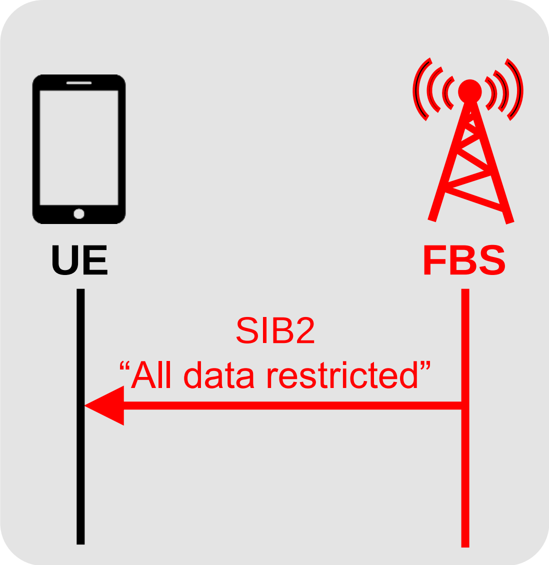
[4] Lichtman et al., LTE/LTE-A jamming, spoofing, and sniffing: threat assessment and mitigation

Attacks in LTE (Design/Implementation Vul.)

- Denial-of-service
  - Denying all or selected network services [2]
  - Selective DoS through access barring [5]
  - DoS several mins ~ several hours ~ until a UE is rebooted or USIM is re-inserted



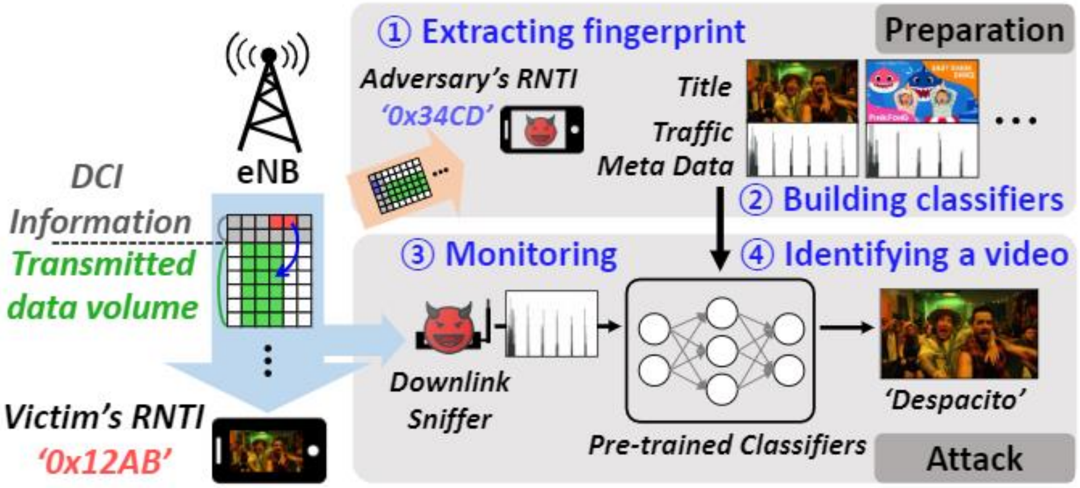


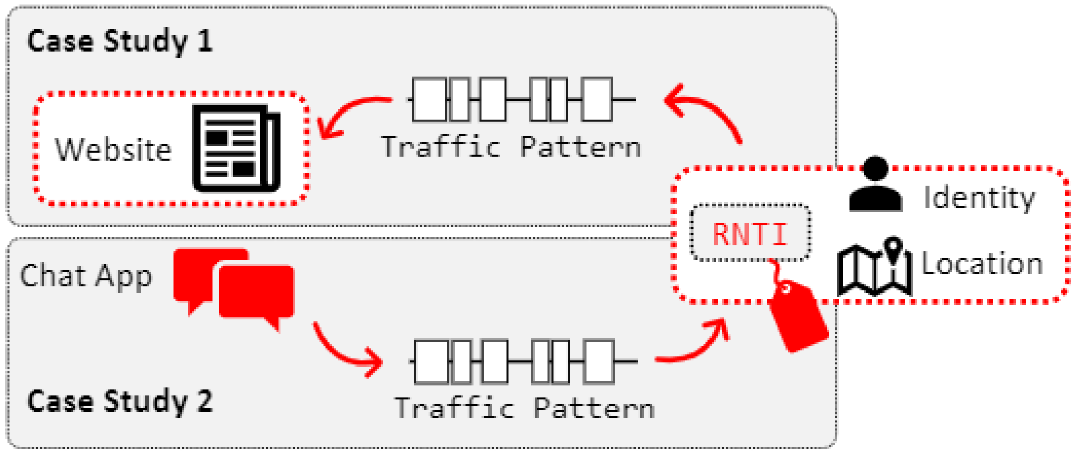


[2] Practical Attacks Against Privacy and Availability in 4G/LTE Mobile Communication Systems
[5] Hiding in Plain Signal: Physical Signal Overshadowing Attack on LTE

Attacks in LTE (Design Vul.)

- Service fingerprinting
  - Video fingerprinting [6]
  - Website fingerprinting [7]

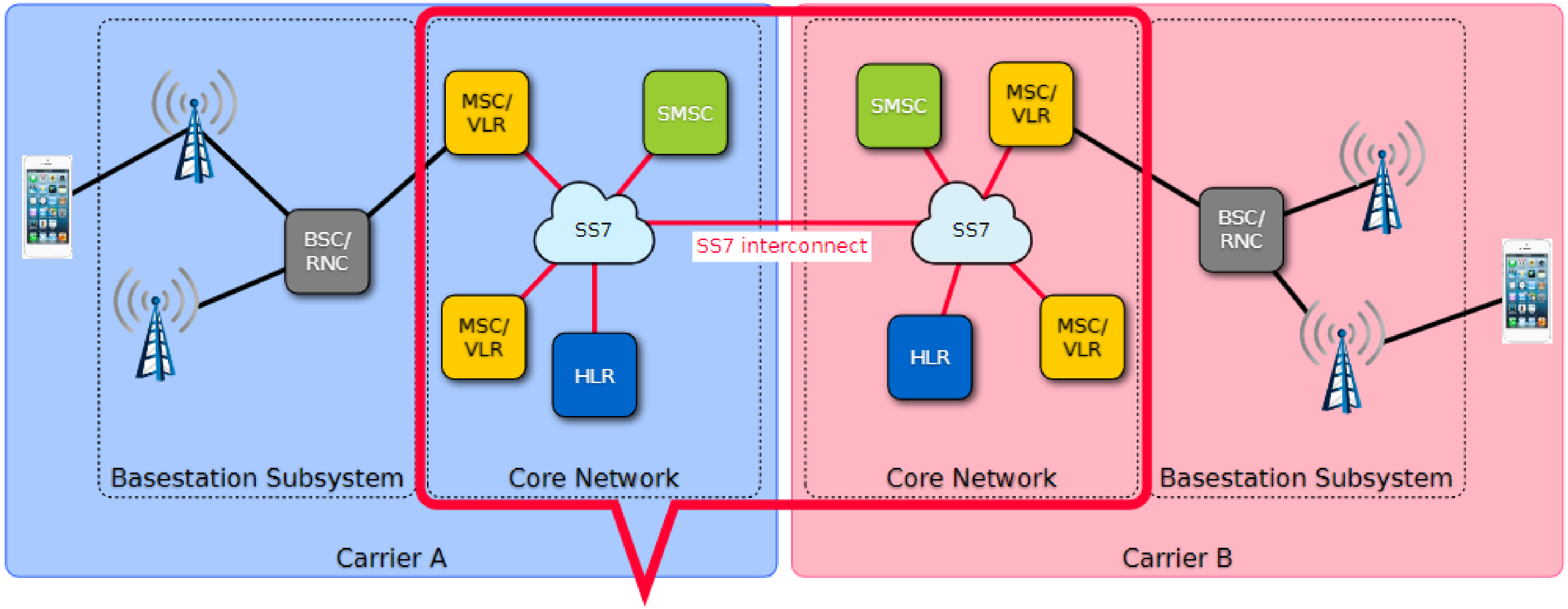




[6] Watching the watchers: Practical Video Identification Attack in LTE Networks
[7] Lost traffic encryption: fingerprinting LTE/4G traffic on layer two

Attacks in LTE (Design Vul.)

❖ SS7 attack
  – Location tracking
  – Denial-of-Service
  – Intercepting calls, SMS

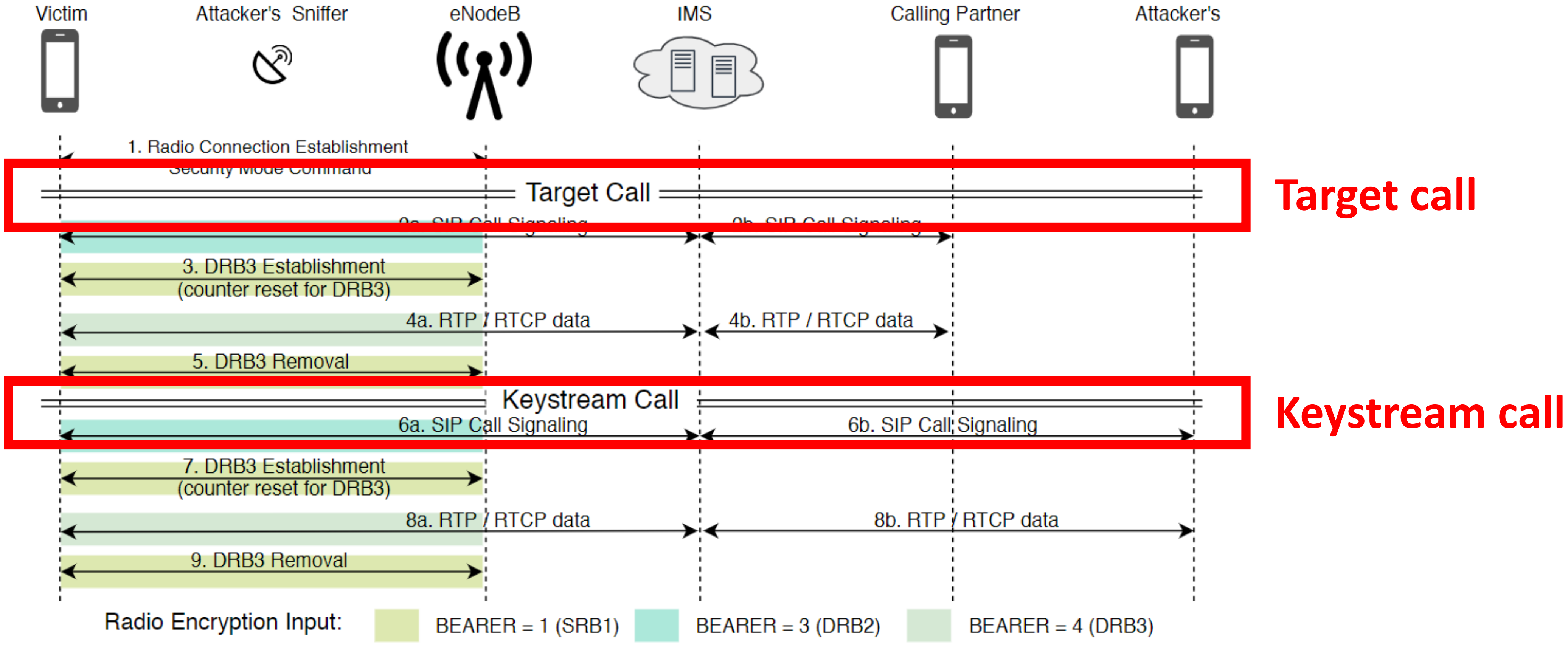


[31C3]

Attacks in LTE (Implementation Vul.)

❖ Keystream reuse @ voice call

– Call Me Maybe: Eavesdropping Encrypted LTE Calls With ReVoLTE [Security'20]

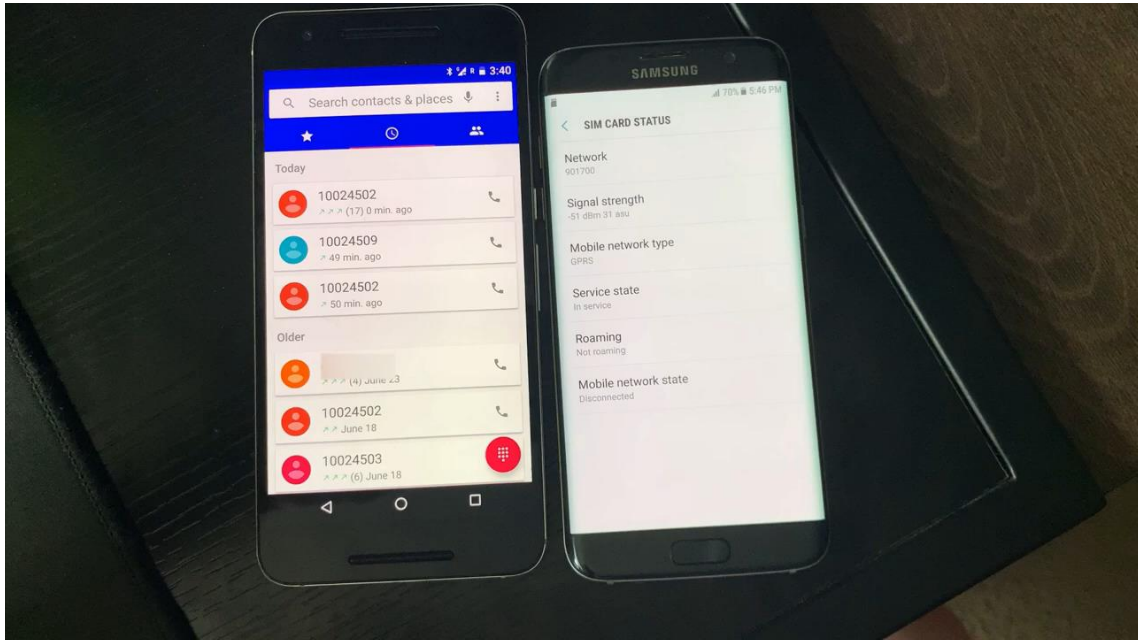


Attacks in LTE (Implementation Vul.)

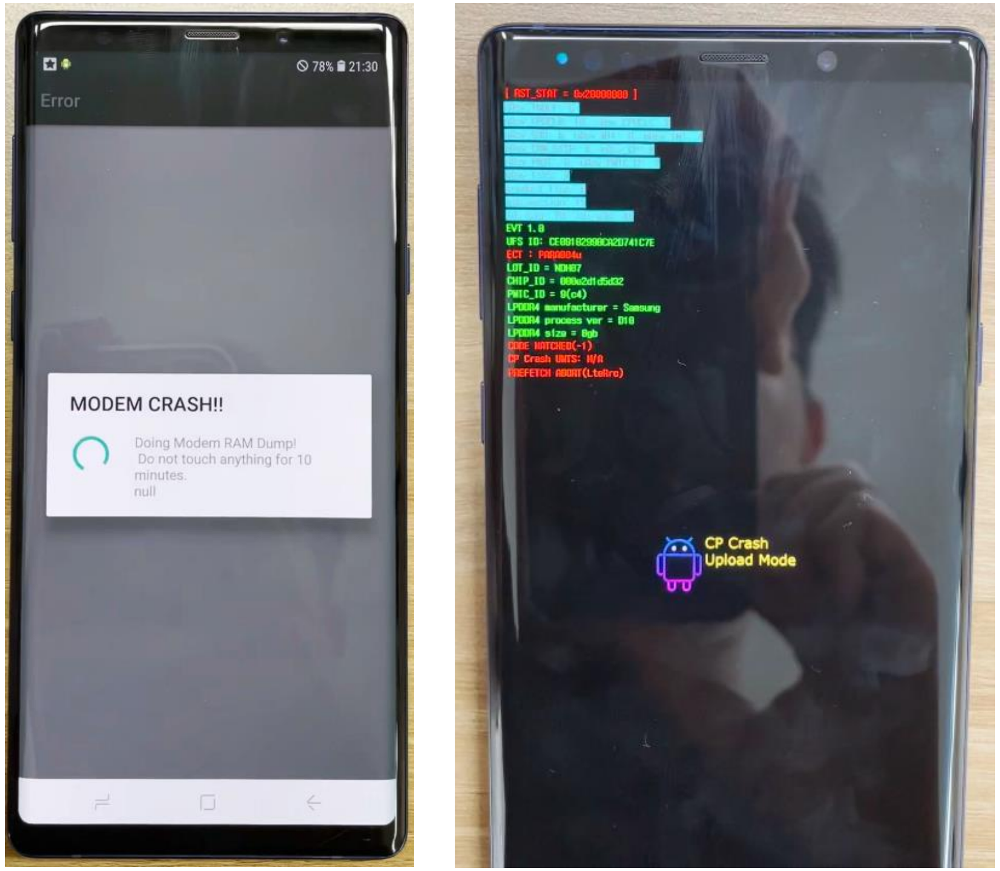
- Implementation vulnerabilities
  - Allowing the use of **null integrity protection**
  - Revealing **IMEI** (a device's unique identity)
  - **Authentication and key agreement (AKA) bypass**
  - **Accepting plaintext messages** even after sharing the security keys
  - **SMS** injection
  - **Network identity and time zone** spoofing
  - …

Attacks in LTE (Implementation Vul.)

- Implementation vulnerabilities
  - Memory corruption vulnerabilities
    - Reverse engineering
    - Fuzzing



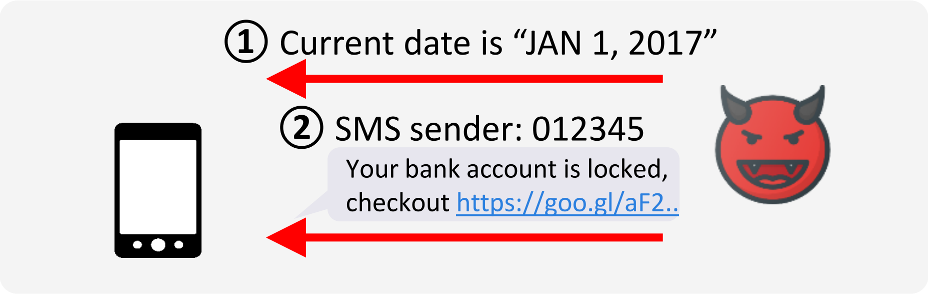
Emulating Samsung's Baseband for Security Testing



FirmWire: Transparent Dynamic Analysis for Cellular Baseband Firmware

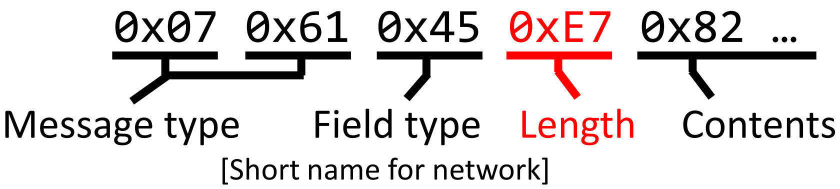
Implementation bugs

- Non-standard-conformant bug
  - Baseband accepts messages with invalid authentication
  - Example



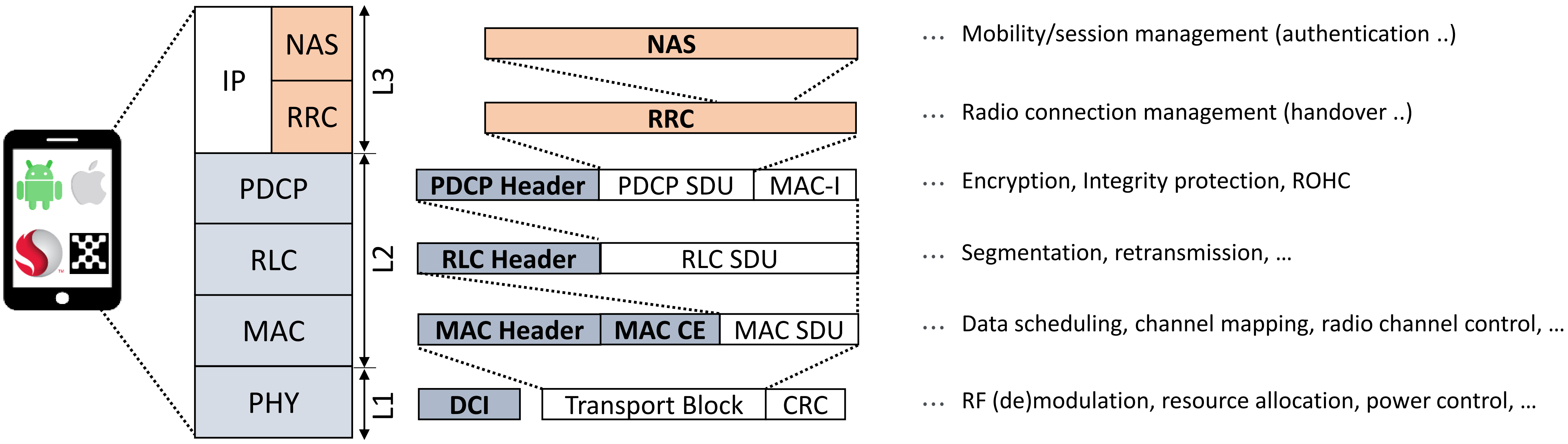
① EMM Information
② Downlink NAS Transport

- Memory bug
  - Baseband processor crashes
  - Example (CVE-2024-20039)



Protocol stack

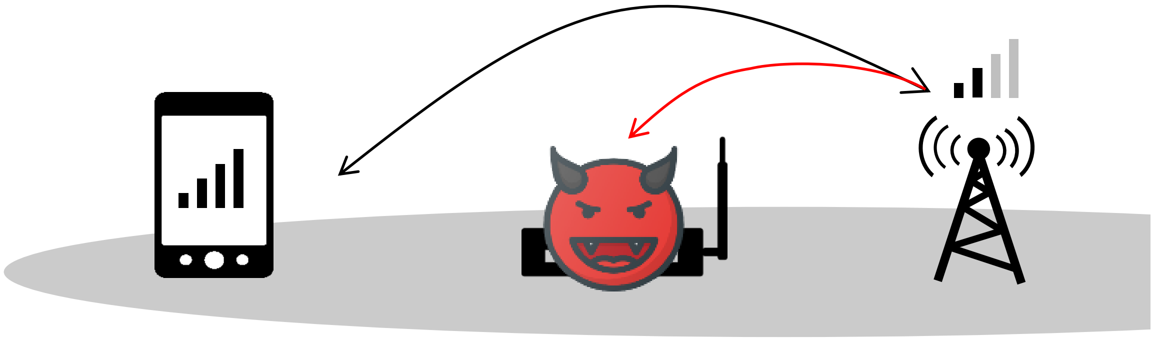
- Layer 3 (NAS, RRC) supports a lot of different message types / fields
  - E.g. RRC defines > 900 IEs (information elements) that contain > 4k fields
- However, lower layers (PDCP, RLC, MAC, PHY) also carry several fields
  - More functionalities from 4G



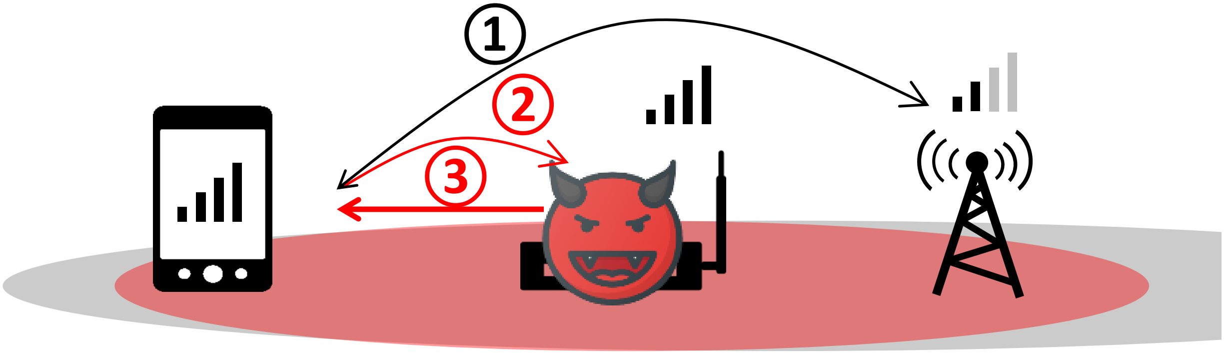
Attack models in LTE

❖ The four representative attackers in LTE

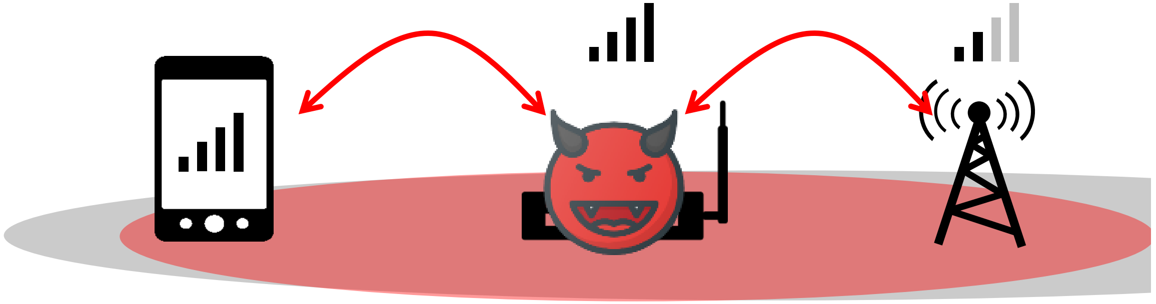
1. Passive (eavesdropping) attacker



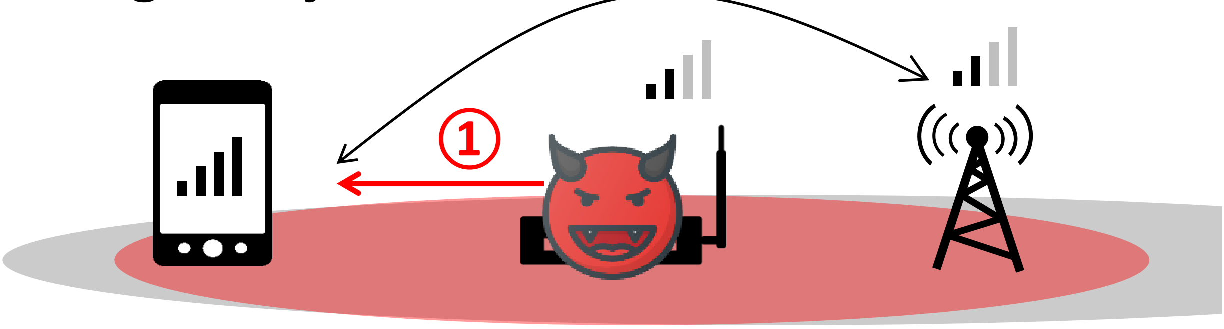
2. Fake base station (Stingray, IMSI-catcher)



3. Man-in-the-middle attacker



4. Signal injection attacker



Attack models in LTE

- Passive sniffer
  - Open-source: LTESniffer*, OWL, FALCON, ..
  - Commercial: AirScope, Wavejudge, ThinkRF, ..

- Fake base station
  - Commercial products: Stringray, chinese market, …
  - Open-source LTE stack

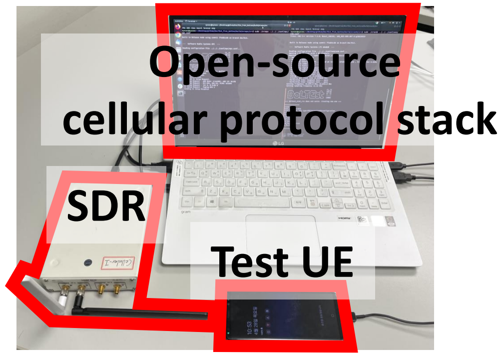
- Signal injection attacker
  - Open-source: SigOver**
  - Not open-sourced: Adapt-over (Mobicom’22), SigOver + alpha (37C3)

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

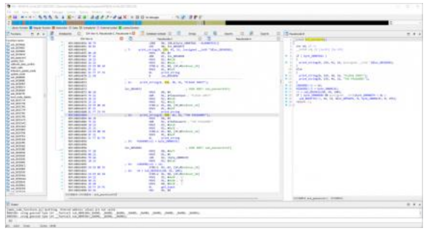
Methodologies: how to find implementation vulnerabilities in cellular devices?

How to find implementation vulnerabilities in cellular devices?

- Over-the-air testing
  - Security testing framework
    - [NDSS'15], [WOOT'16], LTEFuzz [S&P'19], DoLTEst [Security'22], BaseOTA [In-progress], Lower-layer fuzzing, 5GBaseChecker [Security'24] , …
  - NLP, formal analysis, FSM-based diff. analysis, …
    - Hermes [Security'24], Contester [Security'23], CREEK [Security'22], DIKEUE [CCS'21], 5GReasoner [CCS'19]..



- Static analysis
  - Manual analysis @ many hacking conferences, companies, researchers
  - Automatic approaches @ academia
    - BaseSpec [NDSS'21], BaseComp [Security'23]



- Emulation
  - QEMU & AFL++ @ Exynos, MediaTek
    - BaseSAFE [WiSec'20], FirmWire [NDSS'22], SIMurai[Security'24]

CheolJun Park*, Sangwook Bae*, BeomSeok Oh, Jiho Lee, Eunkyu Lee,
Insu Yun, and Yongdae Kim

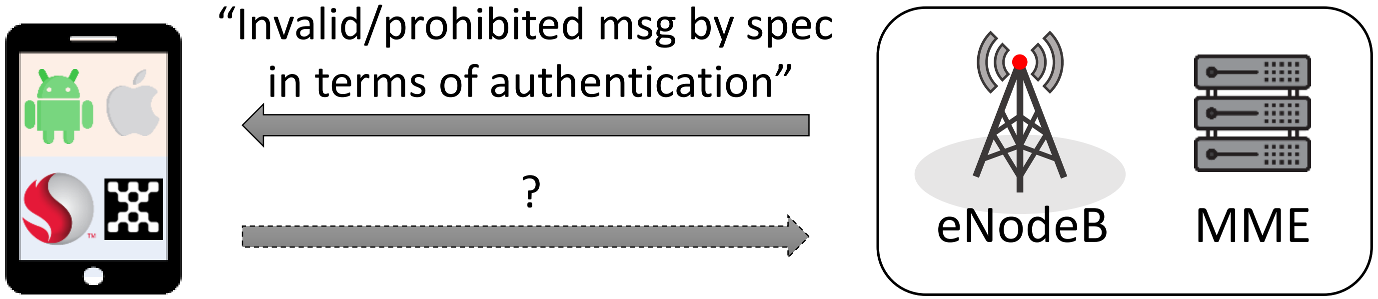
USENIX Security 2022

Developing framework for finding non-standard-conformant bugs (DoLTEst)

Goal

- Finding non-standard-conformant bugs for message authentication in baseband

- Motivation
  - Among **993** test scenarios in conformance specification[1], only **14** cases are negative*
    (check if **invalid or prohibited messages** are appropriately handled)
  - Previous work: Stateless testing, limited coverage in negative messages



Challenges

1. Security-irrelevant state definition in specification
   - Existing definitions states are not proper for security testing

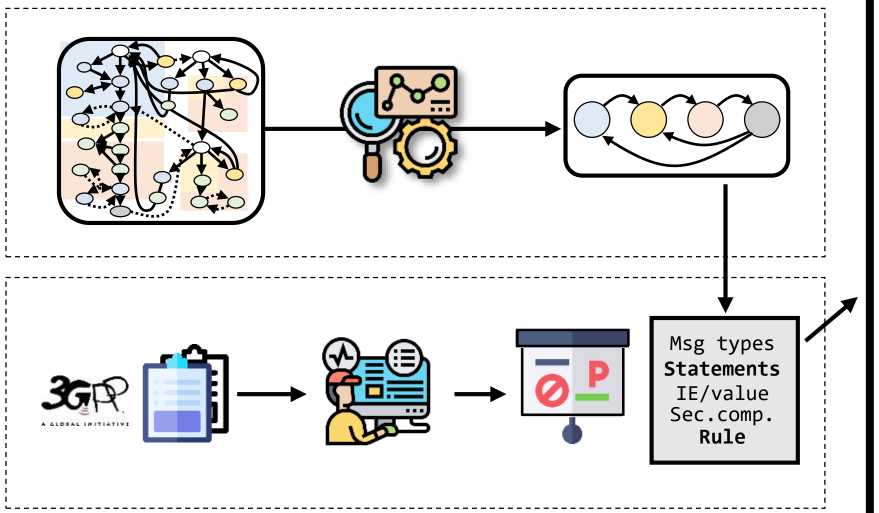
2. Enumerating negative (violating) cases
   - Specification defines >100 message types, and 1,000> optional fields
   - Each trial for negative testing in UEs is expensive

3. Ambiguities in complicate specification
   - Specification is hard to understand
   - Determining the UE's correct behavior when receiving each test case is difficult

Overview of approach (DoLTEst)

1. Manual spec. analysis

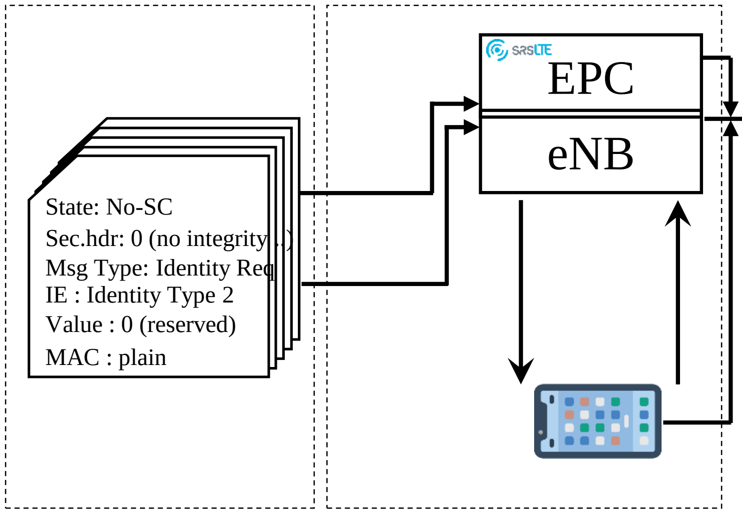
① Define new security-abstracted states



② Construct *guidelines*

2. Test case generation & OTA testing

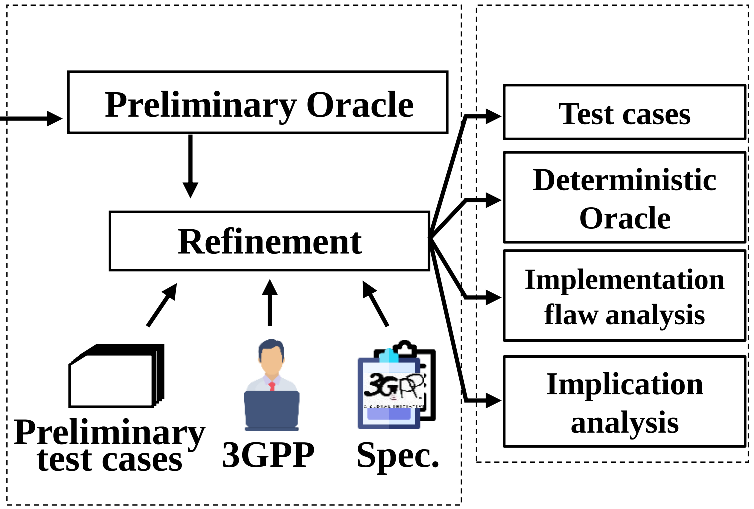
③ Generate test cases



④ Open-source LTE stack based over-the-air device testing

3. Manual post-analysis

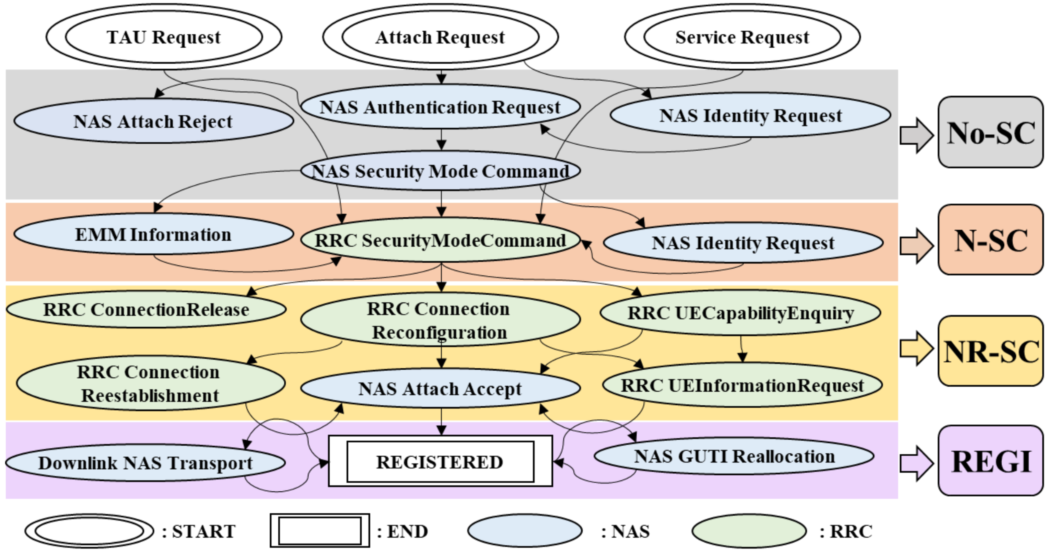
⑤ Deviant behavior analysis

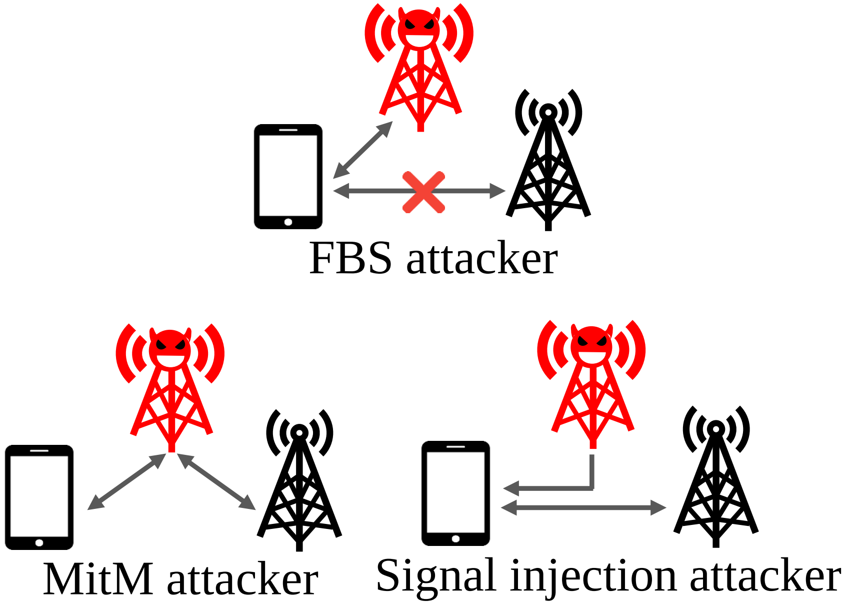


⑥ Flaw& implication analysis, oracle refinement

Security abstracted states

- Re-define the existing implicit UE states as **new security abstracted states**
- Advantages
  - Reflecting **advanced LTE attacks**
  - **Reduce** total number of test cases





Test case generation

- ❖ Goal: Generating test messages that are **invalid or prohibited by specification**
  - – We found every **statement** related with message authentication[1,2]
  - – Addressing ambiguities in the spec: over-approximation

Protocol	Guideline					MAC	Reference	# of test cases for each state	Page #
	No.	State	Security Header Type	Message Type	IE				
RRC	1	*	N/A	RRCConnectionReconfiguration	drb-ToAddModList: {...}	*	A.6, 5.3.1.1 in [7]	2	68p
	2	*	N/A	RRCConnectionReconfiguration	srb-ToAddModList: {SRB2}	*	A.6, 5.3.1.1 in [7]	2	39p
	3	*	N/A	RRCConnectionReconfiguration	measConfig: {...}	*	A.6, 5.5.5.1 in [7]	2	68p
	4	*	N/A	RRCConnectionReconfiguration	mobilityControlInfo: {...} securityConfigHO: {...}	*	A.6, 5.6.5.1 in [7]	2	918p, 72p
	5	*	N/A	RRCConnectionRelease	...	*	A.6 in [7]	2	918p
	6	*	N/A	SecurityModeCommand	integrityProtection: {EIA1, EIA2, EIA3}[c]	*	A.6, 5.3.1.2 in [7]	10	70p
	7	*	N/A	UECapabilityEnquiry	...	*	A.6, 5.6.3.2 in [7]	2	230p
	8	*	N/A	counterCheck	...	*	A.6 in [7]	2	918p
	9	*	N/A	UEInformationRequest	...	*	A.6, 5.6.5.2 in [7]	2	919p
	10	*	N/A	DLInformationTransfer	...	*	A.6 in [7]	2	918p
NAS	11	*	*	Identity Request	Identity Type2: {IMSI}[c]	*	4.4.4.2 in [4]	124	50p, 51p
	12	*	*	Security Mode Command	integrityProtAlgorithm: {EIA1, EIA2, EIA3}[c]	*	4.4.4.1, 4.4.4.2 in [4]	155	50p
	13	*	*	GUTI Reallocation Command	...	*	4.4.4.2 in [4]	31	50p, 51p
	14	*	*	EMM Information	...	*	4.4.4.2 in [4]	31	50p, 51p
	15	*	*	Downlink NAS Transport	...	*	4.4.4.2 in [4]	31	50p, 51p
	16	*	*	Attach Reject	EMM cause:{#25}	*	4.4.4.2, 5.5.1.2.5 in [4]	31	50p, 51p, 129p
	17	*	*	Attach Accept	...	*	4.4.4.2 in [4]	31	50p, 51p

[1]: TS. 24.301, [2]: TS. 36.331

Example

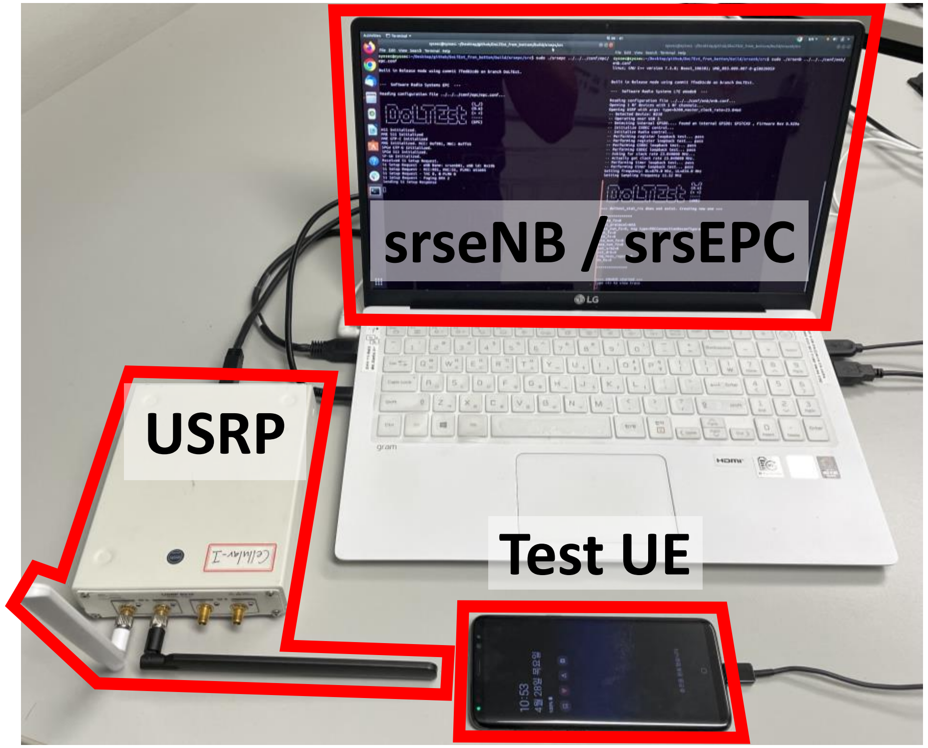
Specification	**Except the messages** … **below**, no NAS signalling messages shall be processed by the UE… unless the network has **established secure exchange of NAS** messages… … - **Identity Request** ((if requested identification parameter is **IMSI**)

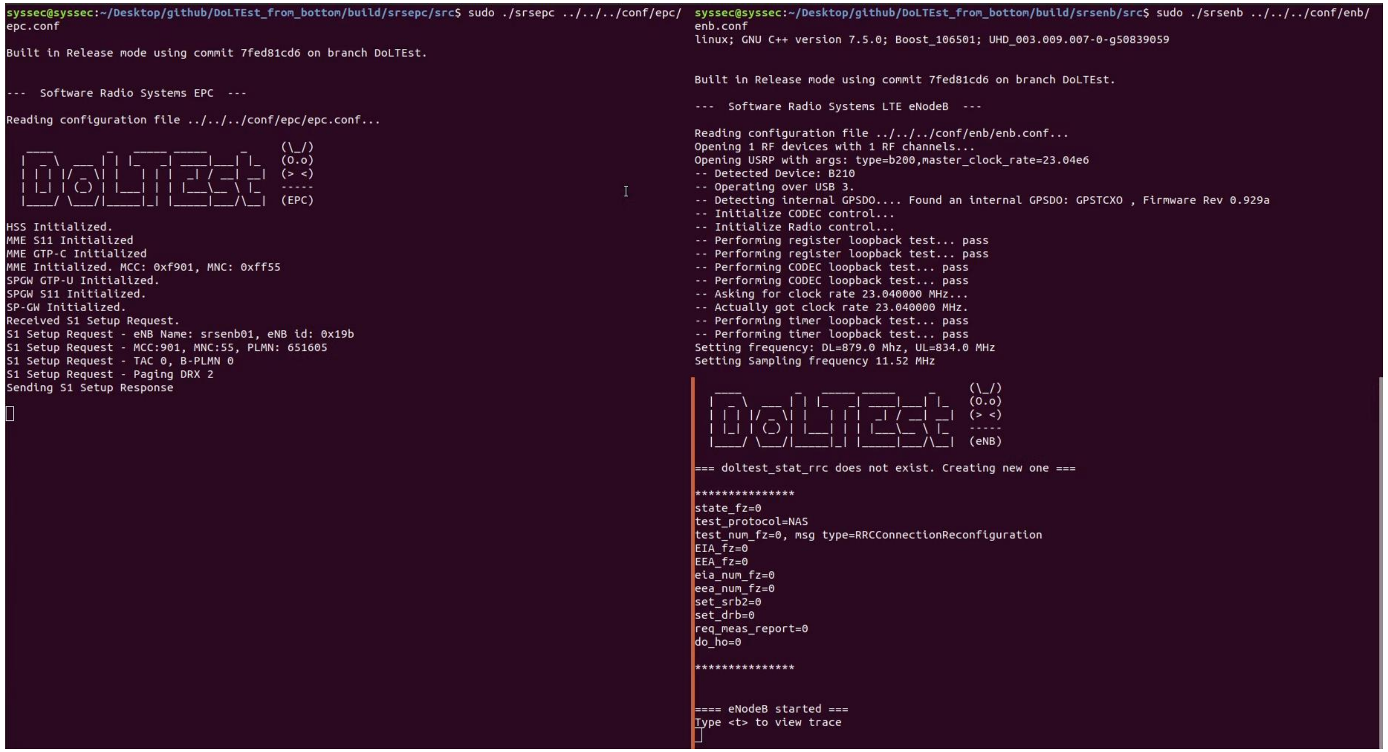
	State	Security Header Type	Message Type	IE	Value	MAC
Guideline	*	*	**Identity Request**	Identity Type 2	**not IMSI**	*

	State	Security Header Type	Message Type	IE	Value	MAC
Over-approximation	No-SC	0 (no integrity protected)	Identity Request	Identity Type 2	0 (reserved)	plain
	…	…	…	…	…	
	No-SC	1 (no integrity protected)	Identity Request	Identity Type 2	2 (IMEI)	random
	…	…	…	…	…	
	No-SC	3 (integrity protected with…)	Identity Request	Identity Type 2	3 (IMEISV)	random
	N-SC	3 (integrity protected with…)	Identity Request	Identity Type 2	3 (IMEISV)	plain

Implementation

- Edited srsLTE (9,234 LoC) to send total 1,848 test messages
  - State control + test message generation





Results

❖ Tested **43** cellular devices from **five** major baseband manufacturers

#	Name	Phone vendor	Baseband vendor	Chipset model	Firmware version	Last update (YYMM)	Implementation flaw
1	iPhone 6	Apple	Qualcomm	MDM9625	7.21.00 / 7.80.04	1810/2101	S1,S3,I1 / S2,S3,I1
2	iPhone 8	Apple	Intel	XMM 7480	4.02.01	2103	I3
3	iPhone XS	Apple	Intel	XMM 7560	1.03.08	1902	I3
4	iPhone 12 Pro	Apple	Qualcomm	Snapdragon X55	1.62.11	2104	-
5	Y9	Huawei	HiSilicon	Kirin 659	21C60B269S003C000	1806	S3,I3
6	P10 Lite	Huawei	HiSilicon	Kirin 658	21C60B268S000C000	1711	I3
7	P10	Huawei	HiSilicon	Kirin 960	21C30B323S003C000	1805	I3
8	Mate 10 Pro	Huawei	HiSilicon	Kirin 970	21C10B551S000C000	1801	I3
9	P20 pro	Huawei	HiSilicon	Kirin 970	21C20B369S007C000	1904	I3
10	Mate 20 pro	Huawei	HiSilicon	Kirin 980	21C10B687S000C000	1812	I3
11	X401	LG	Mediatek	MT6750	MOLY.LR11.W1552.MD.TC01.LVSF.SP.V1.P22	1802	S2,M1
12	X6	LG	Mediatek	Helio P22 MT6762	MOLY.LR12A.R3.TC01.PIE.SP.V1.P10.T12	1907	S2
13	K50	LG	Mediatek	Helio P22 MT6762	MOLY.LR12A.R3.TC01.PIE.SP.V1.P26	2012	S2
14	G6	LG	Qualcomm	MSM8996 Snapdragon 821	MPSS.TH.2.0.1.c3.1-00024-M8996FAAAANAZM-1.142344.1.143233.1	1804	S1,S2,S3
15	V35 ThinQ	LG	Qualcomm	SDM845 Snapdragon 845	MPSS.AT.4.0.c2.9-00057-SDM845_GEN_PACK-1	1901	S1,S2
16	G7 ThinQ	LG	Qualcomm	SDM845 Snapdragon 845	MPSS.AT.4.0.c2.9-00088-SDM845_GEN_PACK-1.299473	2008	S2
17	G8 ThinQ	LG	Qualcomm	SM8150 Snapdragon 855	MPSS.HE.1.0.c4-00104-SM8150_GEN_PACK-1	2101	S2
18	V50	LG	Qualcomm	SM8150 Snapdragon 855	MPSS.HE.1.5.c4-00270.1-SM8150_GENFUSION_PACK-1.215515.14	1909	S2
19	Oppo find X	OPPO	Qualcomm	SDM845 Snapgragon 845	Q_V1_P14,Q_V1_P14	1808	S1
20	Galaxy S4	Samsung	Qualcomm	MSM8974 Snapdragon 800	E330KKKUCNG5	1609	S1,S2,S3,M1,M2,I1,I2,I3
21	Galaxy S5	Samsung	Qualcomm	MSM8974AC Snapdragon 801	G900VVRU1ANI2	1411	S1,S3,M1,M2,I2
22	Galaxy S5 A	Samsung	Qualcomm	APQ8084 Snapdragon 805	G906LKLU1CPK2	1612	S1,S2,S3,M2,I1,I2,I3
23	Galaxy Note5	Samsung	Samsung	Exynos 7 (7420)	N920SKSU2DQH2	1708	S2,M1,I2
24	Galaxy S6	Samsung	Samsung	Exynos 7 (7420)	G920SKSU3EQC9	1704	S2,M1,I3
25	Galaxy Note FE	Samsung	Samsung	Exynos 8 (8890)	N935JJJU4CTJ1	2102	S2,M1
26	Galaxy Note8	Samsung	Samsung	Exynos 9 (8895)	N950NKOU4CRH2	1810	S2,M1
27	Galaxy S8	Samsung	Qualcomm	MSM8998 Snapdragon 835	G950U1UES5CSB2	1902	S1,S2,S3
28	Galaxy Note9	Samsung	Samsung	Exynos 9 (9810)	N960NKOU3DSLA	1912	S2,M1
29	Galaxy S10	Samsung	Samsung	Exynos 9 (9820)	G977NKOU2BTA2 / G977NKOU4DK1	2001/2011	S2,M1,I1,I2 / S2,M1,I1
30	Galaxy S10	Samsung	Qualcomm	SM8150 Snapdragon 855	G977UVRS3YSJK	1911	-
31	Galaxy A31	Samsung	Mediatek	Helio P65 MT6768	A315NKOU1BUA1	2102	S2
32	Galaxy S20	Samsung	Qualcomm	SM8250 Snapdragon 865	G981NKSU1CTKD	2011	-
33	Galaxy A71	Samsung	Samsung	Exynos 9 (980)	A716SKSU1ATF4 / A716SKSU3BTL2	2006/2012	S2,M1,I1,I2 / S2,M1,I1
34	Galaxy Note20	Samsung	Qualcomm	SM8250 Snapdragon 865	N986NKSU1CUC9	2103	-
35	Redmi 5	Xiaomi	Qualcomm	SDM450 Snapdragon 450	MPSS.TA.2.3.c1-00522-8953_GEN_PACK-1_V042	1712	S1,S3
36	Redmi note 4x	Xiaomi	Qualcomm	MSM8953 Snapdragon 625	953_GEN_PACK-1.122638.1.123338.1	1712	S1,S3
37	Mi Max 3	Xiaomi	Qualcomm	SDM636 Snapdragon 636	AT32-00672-0812_2359_46aa9a7	1807	S1
38	Mi 5S	Xiaomi	Qualcomm	MSM8996 Snapdragon 821	TH20c1.9-0612_1733_9fe7ce8	1805	S1,S3
39	Mi Mix 2	Xiaomi	Qualcomm	MSM8998 Snapdragon 835	AT20-0608_2116_6c4a86b	1805	S1,S3
40	Black Shark	Xiaomi	Qualcomm	SDM845 Snapdragon 845	00888-SDM845_GEN_PACK-1.163713.1	1811	S1
41	POCOphone F1	Xiaomi	Qualcomm	SDM845 Snapdragon 845	AT4.0.c2.6-144-1008_1436_e3055ba	1809	S1
42	ZTE Blade V8 Pro	ZTE	Qualcomm	MSM8953 Snapdragon 625	-8953_GEN_PACK-1.79091.1.79899.1	1612	S1,S3
43	ZTE Axon 7	ZTE	Qualcomm	MSM8996 Snapdragon 820	TH.2.0.c1.9-00104-M8996FAAAANAZM	1712	S1,S3

Results

- Tested **43** cellular devices from **five** major baseband manufacturers
  - Qualcomm, Exynos, MediaTek, HiSilicon, and Intel
- Discovered **26** implementation flaws, of which **22** were new

Type of flaw for handling: S*- Security header type, M*- Message type, I*- IE/value

<table>
<tr><th rowspan="2">Protocol</th><th rowspan="2">Message</th><th colspan="6">State</th><th rowspan="2">Implication</th><th rowspan="2">Studied?</th></tr>
<tr><th>No-SC</th><th>N-SC</th><th>NR-SC</th><th>REGI</th><th colspan="2">All</th></tr>
<tr><td rowspan="7">RRC</td><td>RRCConnectionReconfiguration</td><td colspan="2">I1(2)†, I1</td><td colspan="2">M2</td><td colspan="2">-</td><td>AKA bypass (I1), Location leak (I1,M2)</td><td>[36], [52]</td></tr>
<tr><td>RRCConnectionRelease</td><td colspan="2">-</td><td colspan="2">M2</td><td colspan="2">-</td><td>Redirection attack (M2)</td><td>[41]</td></tr>
<tr><td>SecurityModeCommand</td><td colspan="2">I2†,I3</td><td colspan="2">-</td><td colspan="2">-</td><td>Eavesdropping (I2,I3)</td><td>[48]</td></tr>
<tr><td>UECapabilityEnquiry</td><td colspan="2">-</td><td colspan="2">M2</td><td colspan="2">-</td><td>Information leak (M2)</td><td>[53]</td></tr>
<tr><td>CounterCheck</td><td colspan="2">M1</td><td colspan="2">M2</td><td colspan="2">-</td><td>Information leak (M2)</td><td>-</td></tr>
<tr><td>UEInformationRequest</td><td colspan="2">M1†</td><td colspan="2">M2</td><td colspan="2">-</td><td>Location leak (M1,M2)</td><td>[52]</td></tr>
<tr><td>DLInformationTransfer</td><td colspan="2">-</td><td colspan="2">M2</td><td colspan="2">-</td><td>-</td><td>-</td></tr>
<tr><td rowspan="7">NAS</td><td>Identity Request</td><td>I2,I3</td><td colspan="3">-</td><td>S1,S2(2)</td><td rowspan="7">S3</td><td>Information leakage (S1,S2,I2,I3)</td><td>[43]</td></tr>
<tr><td>Security Mode Command</td><td>I3</td><td colspan="3">-</td><td>-</td><td>Eavesdropping (I3)</td><td>[48]</td></tr>
<tr><td>GUTI Reallocation Command</td><td colspan="4">-</td><td>S1</td><td>Identity spoofing (S1), Denial-of-Service (S1)</td><td>[36]</td></tr>
<tr><td>EMM Information</td><td>-</td><td colspan="3">S1</td><td>-</td><td>NITZ spoofing (S1)</td><td>[45]</td></tr>
<tr><td>Downlink NAS Transport</td><td colspan="3">-</td><td>S1</td><td>-</td><td>SMS phishing (S1)</td><td>[43]</td></tr>
<tr><td>Attach Reject</td><td>S2,I2</td><td colspan="3">-</td><td>S1</td><td>Denial-of-Service (S1,S2,I2)</td><td>[52]</td></tr>
<tr><td>Attach Accept</td><td colspan="4">-</td><td>-</td><td>-</td><td>-</td></tr>
</table>

Studied?: Attacks using the message type was previously studied, †: Previously reported

Findings

- Manufacturer-dependent flaws
  - Five NAS (UE ↔ core network) integrity bypass, every **Qualcomm BP**
  - Two RRC (UE ↔ base station) integrity bypass, every **Exynos BP**

- Device-specific flaws
  - Disabling RRC integrity protection (null integrity algorithm), Galaxy S10 (**Exynos**)
  - Exposing measurement report, Galaxy S10 (**Exynos**)
  - AKA (Authentication and Key Agreement) bypass, iPhone 6s (**Qualcomm**)
  - …

- Others
  - Exposing identity @ every **MediaTek/Exynos** BP and some **Qualcomm** BP

CVE-2019-2289, CVE-2021-30826, SVE-2021-20291 (CVE-2021-25516)

Findings

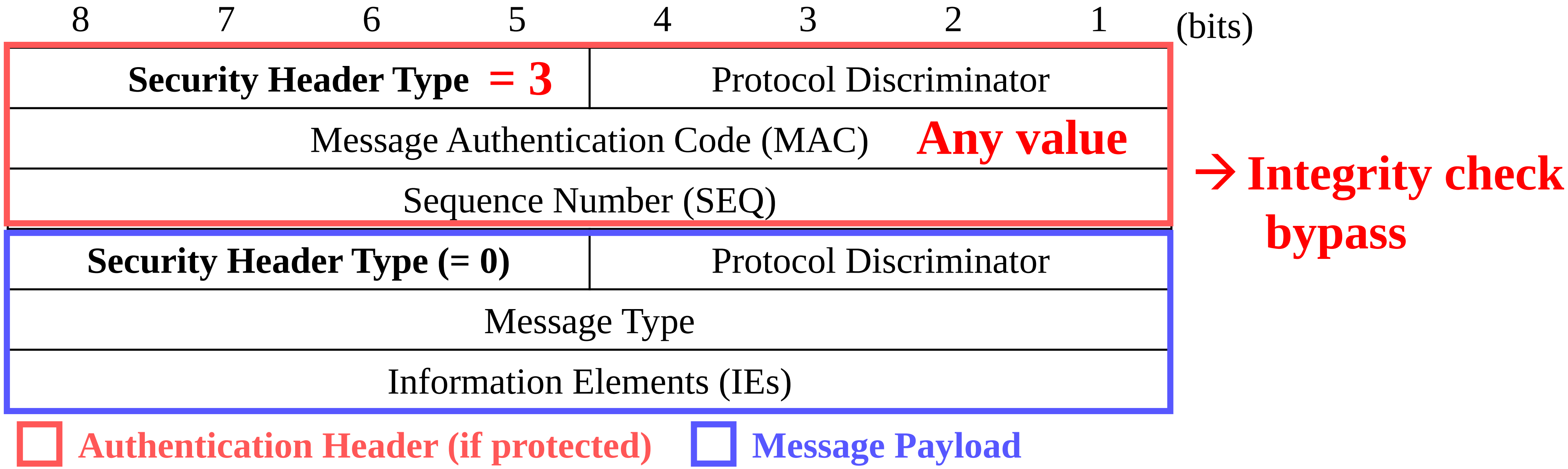
- Example: CVE-2019-2289, Qualcomm, critical (CVSS score: 9.8)

8	7	6	5	4	3	2	1 (bits)
Security Header Type				Protocol Discriminator			
Message Authentication Code (MAC)							
Sequence Number (SEQ)							
Security Header Type (= 0)				Protocol Discriminator			
Message Type							
Information Elements (IEs)							

Authentication Header (if protected) **Message Payload**

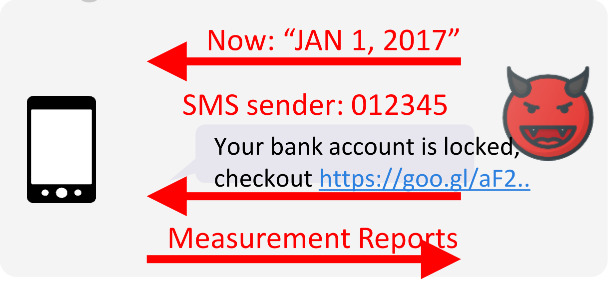
Findings

❖ Example: CVE-2019-2289, Qualcomm, critical (CVSS score: 9.8)



Attacks

- NAS integrity bypass
  - Network identity and time zone spoofing,
  - Device identity capturing (IMSI, IMEI)
  - SMS injection
- RRC integrity bypass
  - Location leakage
- RRC security misconfiguration
  - Eavesdropping and manipulating data traffic
- Deviant behaviors for handling non-standard-conformant messages
  - Device fingerprinting



Baseband	Device	Message				
		#1	#2	#3	#4	#5
Intel	Apple iPhone XS	·	·	·	A_5	·
Qualcomm	Xiaomi Mi Mix 2	·	A_2	A_4	A_5	A_3
Exynos	Samsung Galaxy S10	A_1	·	A_4	A_5	·
MediaTek	LG K50	·	·	A_4	A_6	·
HiSilicon	Huawei Mate 20 Pro	·	A_3	·	A_5	·

What else?

- **Old bug reappearing**
  - Allowing null integrity algorithm is an old (early-LTE) bug
  - However, it suddenly re-appeared on brand-new device, Galaxy S10 (Exynos)

- **New bug after firmware patch**
  - After patching to the latest firmware, new bug appeared
  - Galaxy S8 (Qualcomm), iPhone 6s (Qualcomm)

Summary of non-standard-conformant bugs analysis

- **Only a few negative test cases** in the conformance specification

- **DoLTEst**: a negative testing framework for finding non-standard-conformant bugs in UE
  - Tested 43 devices and found 26 implementation flaws
  - Brand-new device, firmware patch can bring a new logical bugs
  - Open-sourced (https://github.com/SysSec-KAIST/DoLTEst)

- **The conformance test specification** 3GPP should **include much more negative test cases**

Best/good questions

❖ Best

- With the rapid evolution and commercialization of NLP, I believe that rewriting specifications in a computer-readable format (similar to code) could greatly enhance processing with current technology. Do you think this direction of development is realistic? (YoungHyo Kang)
- What do you think is the most effective way to check whether the LTE implementation correctly complies with the standard? Is fuzzing, like in this paper, the most effective approach? (Changgun Kang)
- How can the telecommunications industry ensure continuous integration of comprehensive negative testing frameworks like DoLTEst into future mobile communication standards (e.g., 5G or 6G)? (Donghyo Bang)

❖ Good

- What are the practical limitations of using over-the-air testing for large-scale deployments of DOLTEST?
- Were there any cases where vulnerabilities persisted despite previous patches, and what lessons can be drawn from such occurrences?
- As we enter the age of 5G NR (and imminent 6G!) will similar attacks be possible? To put it in another way, do you think that the LTE standards will be expanded to mitigate such attacks, or do you think we will just move to NR which (hopefully!) has a more robust standard than LTE?
- Were there significant differences in the types of vulnerabilities found across different baseband manufacturers?
- 5G? (most of the student's question)
- The NSA ANT catalog, revealed in 2013, is a classified product catalog by the NSA that shows many pieces of equipment for attacking the GSM network, such as eavesdropping or hacking phones to perform malicious operations. With advancements in security research for cellular networks, do you think government agencies are still able to conduct these kinds of activities?
- What do you think is the most effective way to check whether the LTE implementation correctly complies with the standard? Is fuzzing, like in this paper, the most effective approach?

OTABase: Finding Memory Bugs in the
LTE Cellular Baseband via Over-the-Air Interface

CheolJun Park, Marc Egli, Tuan Dinh Hoang, Mathias Payer,
Insu Yun, and Yongdae Kim

In-progress

Developing framework for finding memory bugs (BaseOTA)

경희대학교
KYUNG HEE UNIVERSITY

Goal

- Finding memory bugs in baseband protocol implementation using **<u>OTA interface</u>**

- Motivation
  - Previous works have the following limitations
    - A lot of reverse engineering, applicability, stateless
  - No one focused on OTA approach
    - “We don’t recommend OTA live fuzzing at all!” (Recon’16)
    - ..
    - “Finding security bugs in basebands is prohibitively difficult.. OTA testing is slow” (NDSS’22)

Challenges

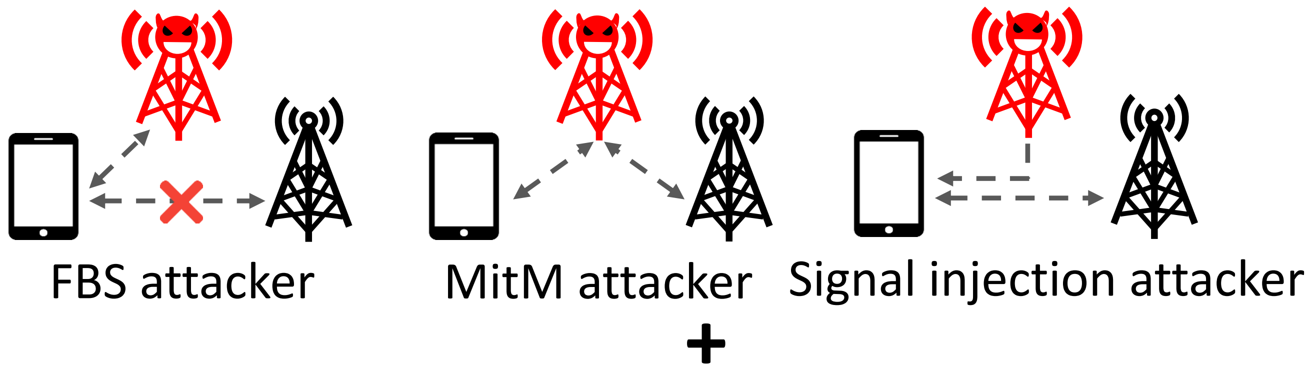
- Test case generation without coverage feedback
  - Specification defines large number of messages and their fields

- Fragile radio connection
  - UE determines whether to connect or transition between states
  - Slow and unstable radio connection

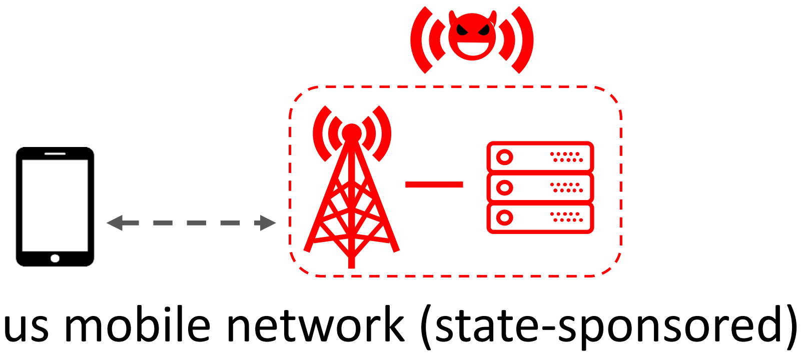
- Limited oracles for detecting crashes
  - We don't have a memory access

Attack model

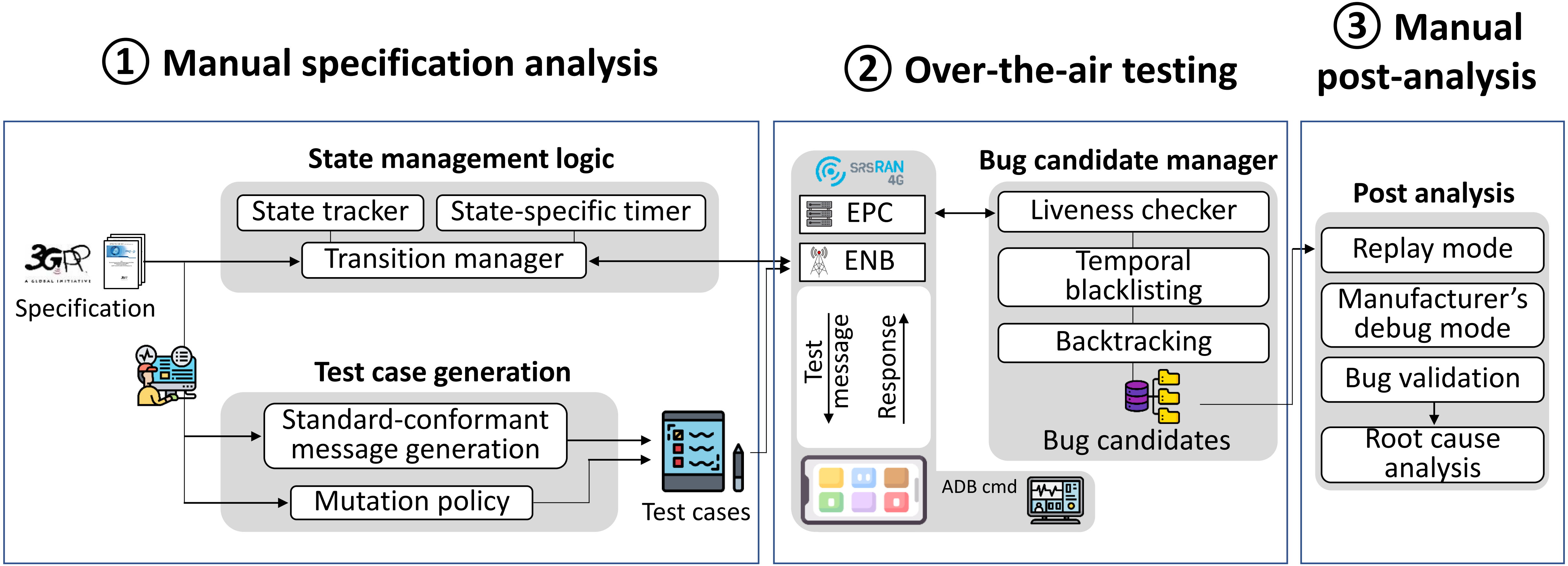
- Memory bugs
  - Attacks: DoS, potential RCE
  - Beyond the implications of authentication bypass



New attack model:

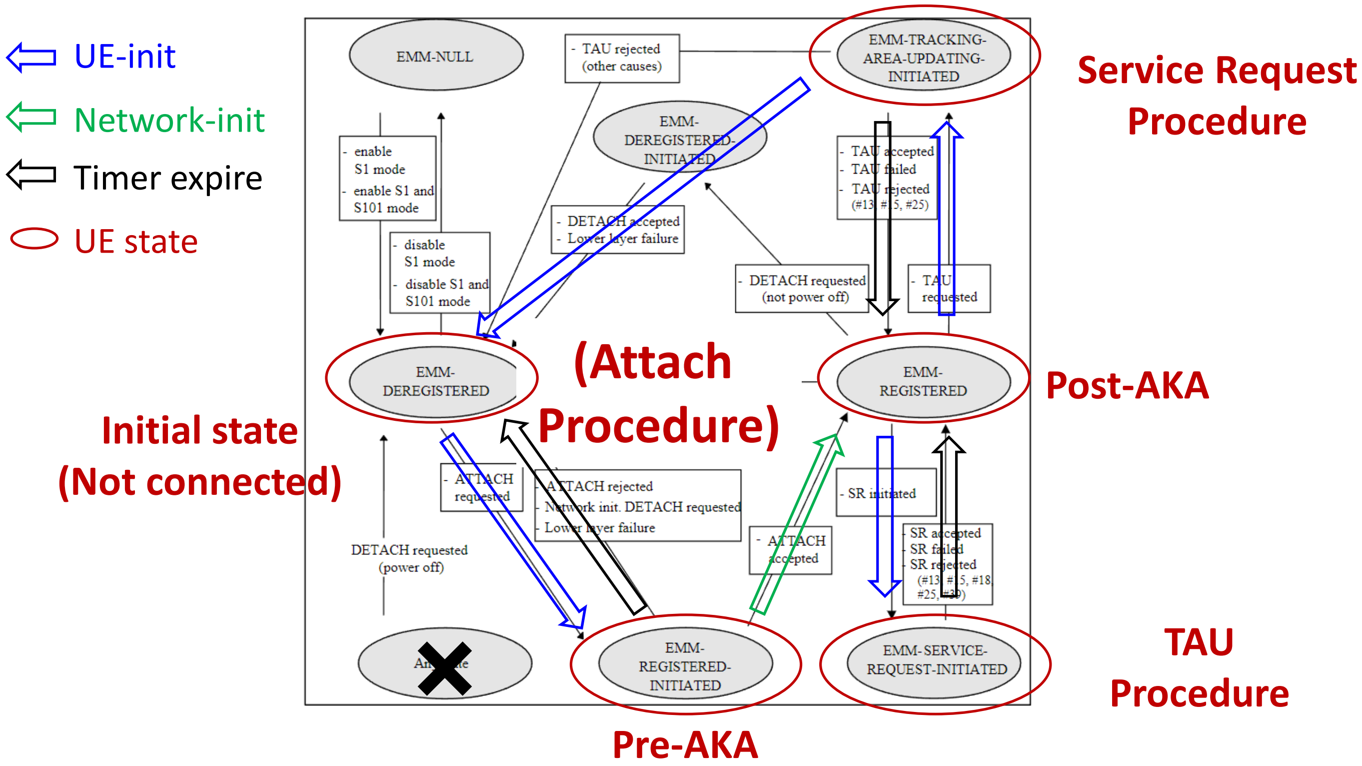


Overview of approach (BaseOTA)



Challenge 1

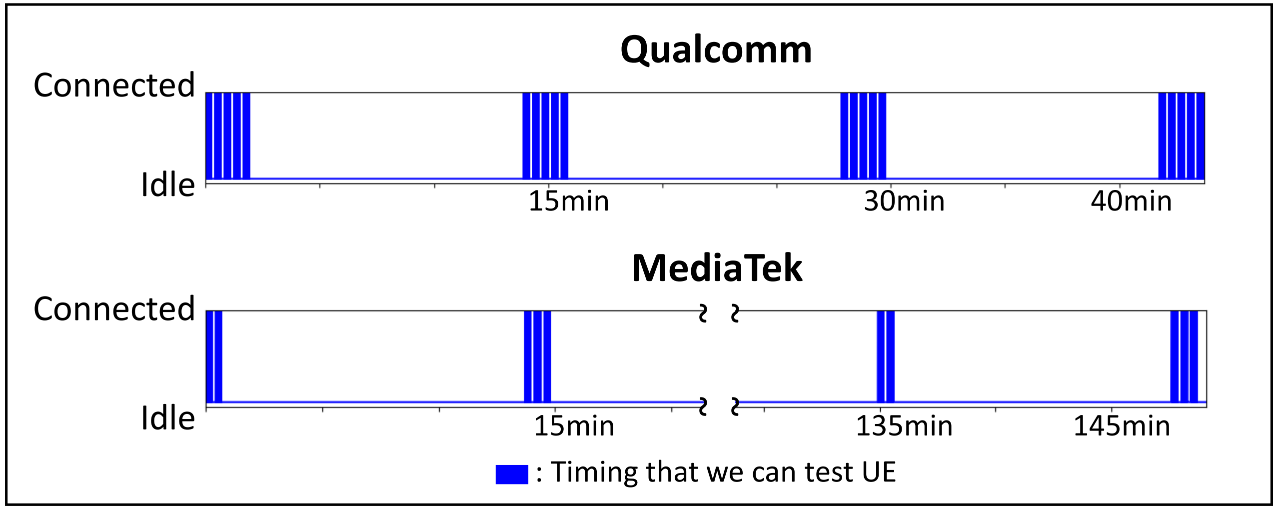
- The baseband is stateful and **initiates most of state transitions**
  - However, network should trigger transitions for testing
  - From initial state, transition takes ~ 0.3 - 1 second (slow for testing), and stays for 5 or 15 seconds (prepare for state expiration)



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

Challenge 1

- Also, 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)

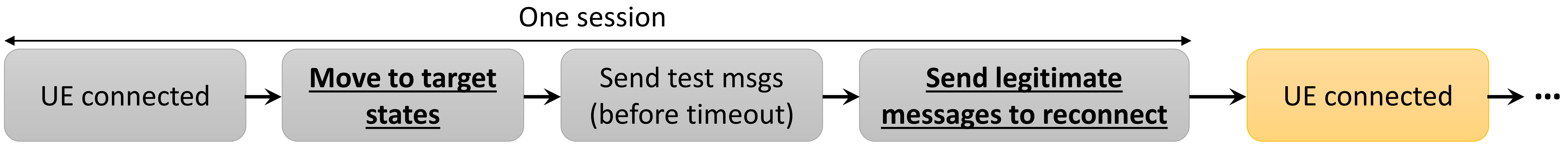


UE's connection status in a normal testing scenario

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

Approach 1

- 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 2

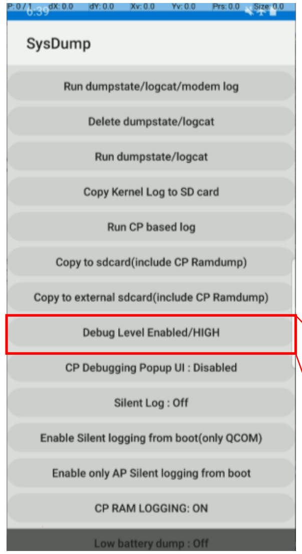
- Limited oracles for detecting crashes
  - Previous works used i) memory sanitizer (emulation) or ii) crash log at terminal

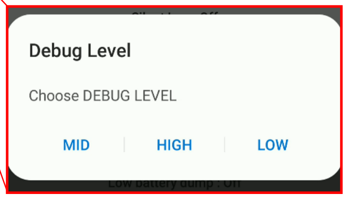
- Existing methods to confirm crash after replay
  - Not reliable or scalable

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	☺	☺	☹

Challenge 2

- Limited oracle for detecting crashes
  - Manufacturer's debug mode (troubleshooting features) is reliable, but





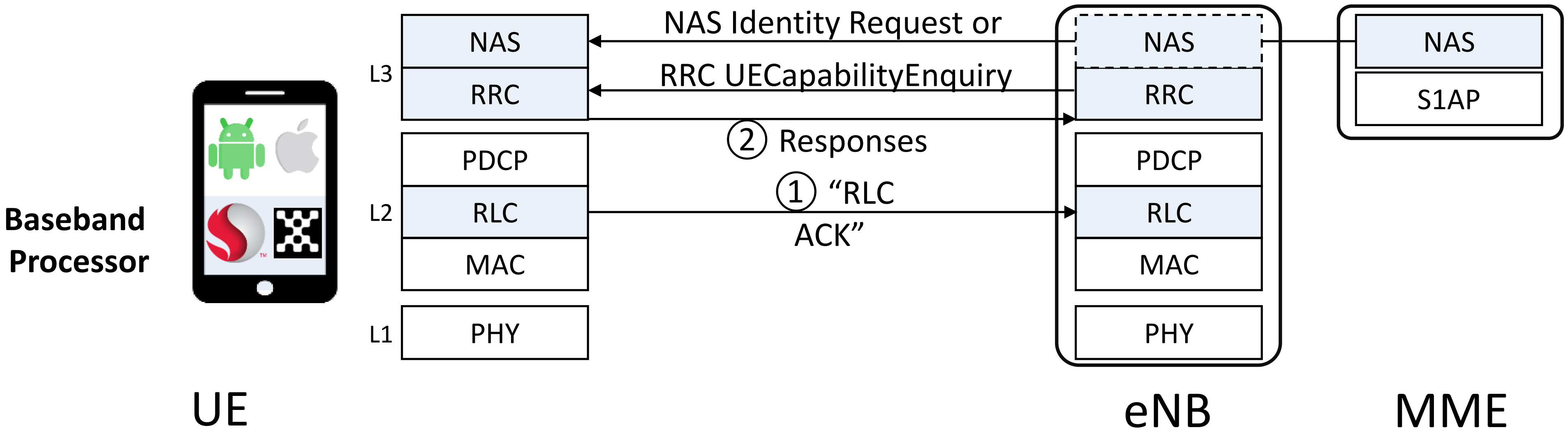
w/ debug mode

w/o debug mode

	Pros	Cons
w/ debug mode	No false positive (Kernel panic)	**Time delay, require manual reboot**
w/o debug mode	Testing automation (modem driver reboots the modem)	False positives

Approach 2

- ❖ 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)



Approach 2

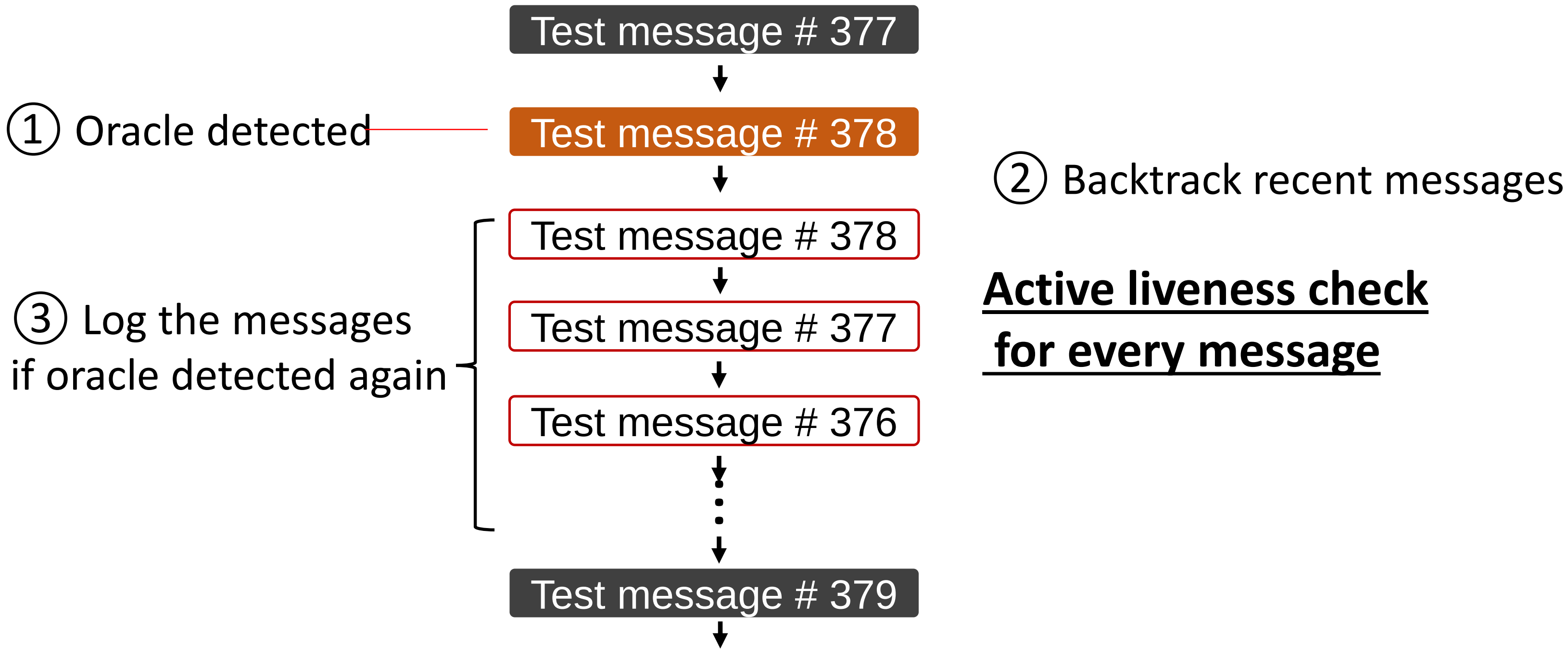
- Tradeoff in the active liveness detection
  - Accuracy
    - Active > Passive: For a few bugs, layer2 died slightly later (few packets later)
  - Speed
    - Active < Passive: Packet transmission degrades testing speed (e.g. if we always send, 50%)

Liveness check method	Accuracy	Speed degrade
Active (layer 3)	More accurate	20 ms ~ 100 ms
Passive (layer 2)	Less accurate	0

- Thus, we send active probing packet for every N test message in normal testing

Approach 2

- Backtracking logic
  - Active liveness check for every message
  - Replay previous N messages and save a bug candidate



Approach 2

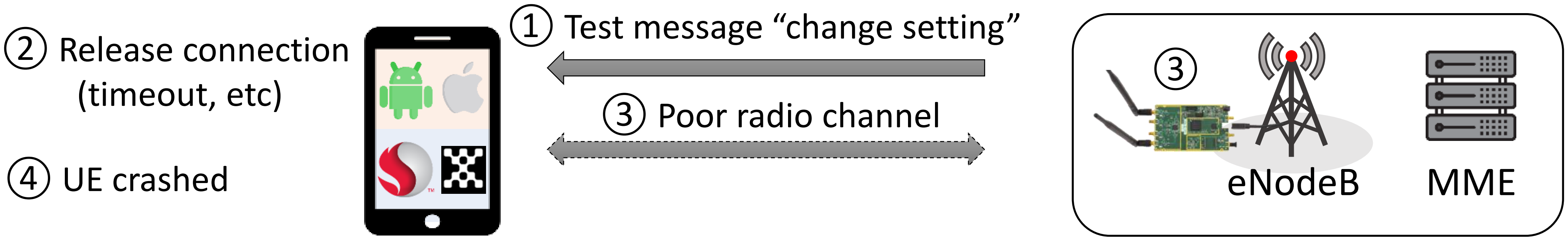
❖ Summary of the proposed oracle

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	☺	☺	☹
Liveness check using cellular protocol	No response	Proposed work	☺	☹	☺

Challenge 3

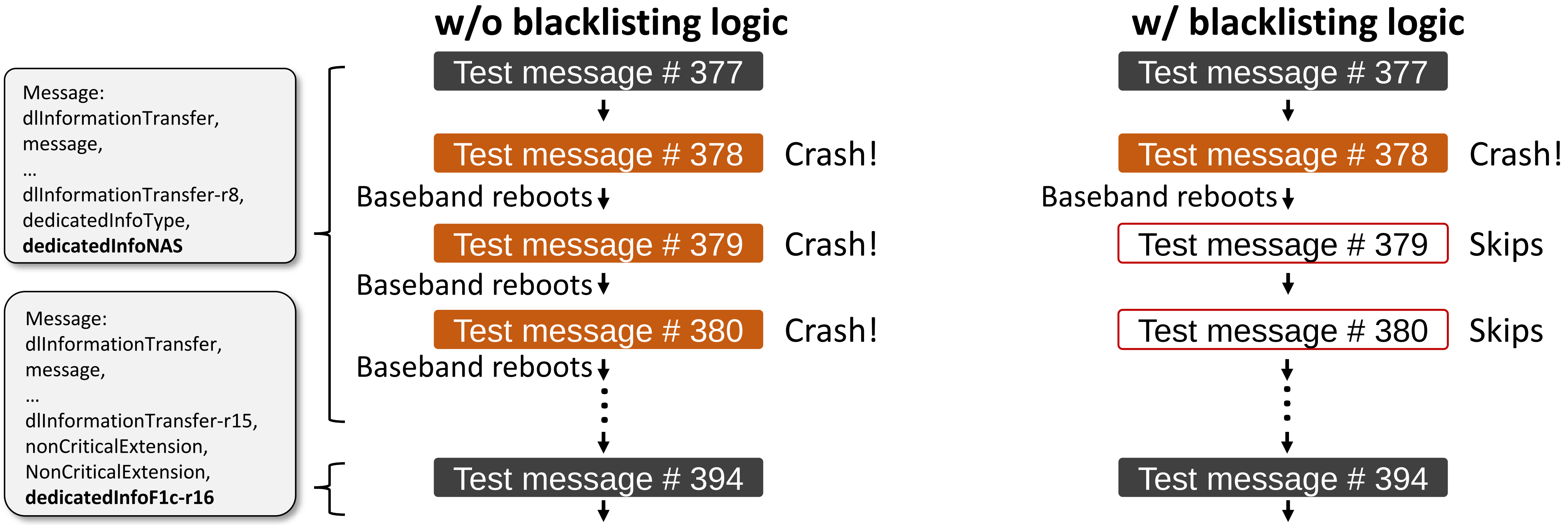
❖ UE hangs or disconnects due to various reasons

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
   - USRP (Hardware) failure
4. UE crashed



Approach 3

- When UE is crashed
  - Temporally blacklist target field for testing
  - Mutations for targeting the same {message + field} will keep crashing the UE
    - Degrades testing speed a lot since it crashes a baseband



Approach 3

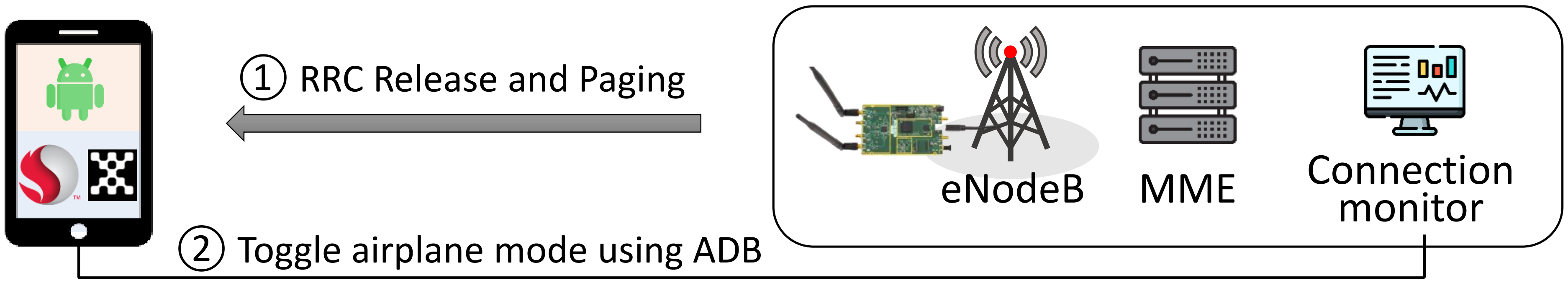
❖ When UE is disconnected or do not respond

– Reconnection 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 A, use ADB to toggle airplane mode



Challenge 4

- 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
  - 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

  - Meanwhile, the number of trial in OTA is limited

Approach 4

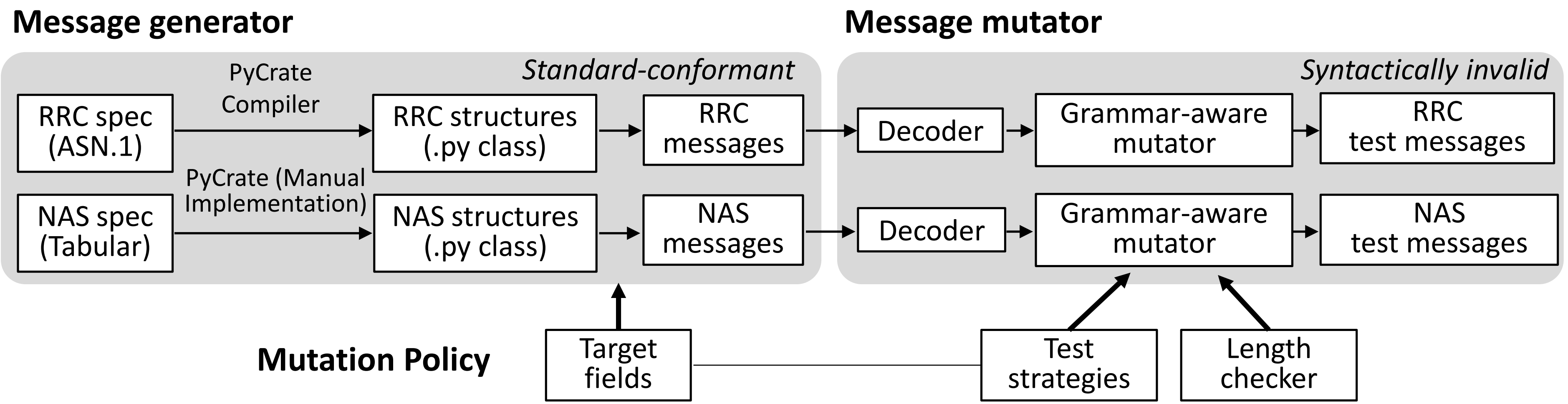
❖ Grammar-guided test case generation based on RRC/NAS specification

1. Analyze the maximum length of the message
   - Fundamental constraints by design (layer 2 – 8188 bytes)
   - RRC - 2042 bytes and NAS - 2037 bytes

2. Empirically find the maximum reliable testing speed
   - We can transmit a lot, but can't ensure if they are all processed
   - RRC – ~ 50 msg/s, NAS – Varies a lot (max 50 msg/s)
   - Previous:
     - SMS-of-death (Security'11) : 1 msg per 1s
     - Berserker (WiMob'21) : 1 msg per 20 ~ 125 s
     - DoLTEst (Security'22) : 1 msg per 2s

Approach 4

- Grammar-guided test case generation based on RRC/NAS specification
  3. Target security-sensitive IEs (information elements) and fields
     - Length and those interested in terms of memory corruption
     - Target: 709 / 4066 RRC fields, 52 out of 62 NAS IE

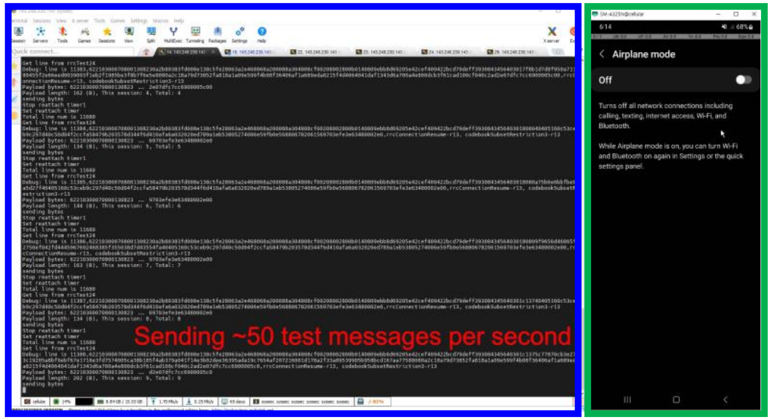


RRC: release v17.4, NAS: release v16.6 *IE (information element): group of information (≈ field

Evaluation

❖ Implemented on top of srsLTE (C++ 5,116 LoC) and pycrate (python, 6,091 LoC)

eNB/EPC



Test UE (mirrored)

Message name	Attach Accept	Detach Request	Detach Accept	TAU Accept	TAU Reject	Service Reject	Service Accept	GUTI Realloc. Command	Auth. Request	Auth. Reject	Identity Request	Security Mode Command	EMM Status	EMM Information	DL NAS Transport	CS Service Notification	DL Generic NAS Transport
Number	12686	399	149	12657	940	1476	1258	2914	748	149	250	2437	200	2275	600	1597	1207

# of test messages per NAS message type

Message name	csfbParameters Response CDMA2000	dlInformation Transfer	Handover FromEUTRA Preparation Request	Mobility From EUTRA Command	rrcConnection Reconfiguration	Security Mode Command	ueCapability Enquiry	Counter Check	ueInformation Request-r9	Logged Measurement Configuration -r10	rnReconfiguration-r10	rrcConnection Resume-r13	dlDedicated Message Segment-r16
Number	138	750	138	726	108183	69	360	99	81	396	1797	40668	150

# of test messages per RRC message type

Results

- Tested **6** cellular devices from **3** major baseband manufacturers (new, old)
  - Qualcomm, Exynos, and MediaTek
- Discovered **7** 0-day and **3** 1-day implementation flaws

Baseband vendor	Device model	Chipset model	# of NAS bug (0-day/1-day)	# of RRC bug (0-day/1-day)
Qualcomm	Galaxy Zflip 4	SM8475 Snapdragon X65 5G	0/0	0/0
	Galaxy S8	MSM8998 Snapdragon 835	0/0	0/0
Exynos	Galaxy S21	Exynos 10 (2100)	3/0	0/0
	Galaxy Note8	Exynos 9 (8995)	3/1	0/2
MediaTek	Galaxy A32	Helio G80 MT6769V/CU	3/0	1/0
	Galaxy A31	Helio P65 MT6768	3/0	1/0

Finding

1. NAS Detach Accept, Authentication Reject with **more bytes than defined** (any state)
   - CVE-2023-37366 (Google Android Security Team, Exynos)

2. **Incorrect checking the length** of certain IE in 4 types of NAS message (post-aka)
   - 3 MediaTek, 1 Exynos
   - CVE-2024-20039 (MediaTek, CVSS score: 8.8, RCE)

3. **Missing contents** in RRC DLInformation message (any state)
   - CVE-2023-32890 (MediaTek)
   - NULL point dereference

4. 1-day bugs from old devices

Summary of memory bugs analysis

- Proposed methods to circumvent over-the-air challenges based on the specification
  - We found 0-day bugs that previous emulation works could not find

- Lessons learned
  - Finding memory bugs were quite painful as many things were unknown
  - Contrary to common beliefs, **dynamic over-the-air approach** can effectively find memory bugs in baseband protocol implementation

Future works

- Within cellular technology
  - Testing uplink (base station and core network)
    - Challenges: require test network access
  - Applicability to 5G SA
    - 5G RRC and NAS are a similar to 4G
    - Challenges: hardware and open-source support
  - Testing lower layer implementation (L1 and L2)
  - **Defense system using design vulnerabilities**

- Testing other protocol
  - Wireless, black box and specification-based system
  - Wifi, Bluetooth, LoRaWAN, ..

Conclusion

- **Specification-based over-the-air dynamic approach for effective discovery of protocol implementation bugs in cellular baseband**

- Finding non-standard-conformant bugs
  - Using DoLTEst, we found that a lot of basebands fail to handle non-standard-conformant messages in terms of message authentication

- Finding memory bugs
  - Using BaseOTA, we found memory bugs in protocol implementation in over-the-air

- We should keep doing specification-driven baseband testing!

Thank you for listening!

Q/A