DoLTEst: In-depth Downlink Negative Testing Framework for LTE Devices

CheolJun Park*, Sangwook Bae*, BeomSeok Oh, Jiho Lee, Eunkyu Lee, Insu Yun, and Yongdae Kim
Korea Advanced Institute of Science and Technology (KAIST)
{fermioncj, hoops, beomseoko, jiholee, ekleez, insuyun, yongdaek}@kaist.ac.kr

Abstract

An implementation flaw in LTE control plane protocols at end-user devices directly leads to severe security threats. In order to uncover these flaws, conducting negative testing is a promising approach, whose test case only contains invalid or prohibited messages. Despite its importance, the cellular standard mostly focuses on positive test cases, producing many implementation vulnerabilities unchecked, as evidenced by many existing vulnerabilities. To fill this gap, we present DoLTEst, a negative testing framework, which can comprehensively test an end-user device. Enumerable test cases with a deterministic oracle produced from detailed specification analysis make it suitable to be used as a standard to find implementation vulnerabilities. We uncovered 26 implementation flaws from 43 devices from 5 different baseband manufacturers by using DoLTEst, demonstrating its effectiveness.

1 Introduction

Despite the passage of 13 years since the first release, long-term evolution (LTE) remains the dominant protocol over the newly implemented 5G network. Recent reports show nearly 6 billion LTE subscriptions worldwide, whereas 5G subscriptions only exceed 0.4 billion [20, 25]. Additionally, most 5G-capable devices still support the LTE protocol, owing to backward compatibility needs and slow 5G standalone (SA) mode deployment, compared to non-standalone (NSA) mode [26].

LTE security remains critical because it provides privacy-sensitive data plane services, such as voice calling, short message services (SMS), and internet access. For these data plane services to work efficiently and safely, the LTE control plane supports basic control operations, such as security control, mobility management, and authentication. Thus, any insecurities related to the control plane can affect the confidentiality, integrity, and availability in the data plane.

For LTE security, discovering implementation vulnerabilities is as important as eliminating standard (design) vulnerabilities. Even though researchers have uncovered substantial design flaws in the LTE control plane [17, 29, 30, 32, 36, 38, 40, 41, 49, 51–53], it does not necessarily lead to secure implementations. Implementation vulnerabilities have been continuously reported, including memory corruptions [22, 34, 42, 43] and non standard-compliant vulnerabilities [16, 27, 36, 43, 46, 48, 50]. Although the former mainly originates from development mistakes, the latter involves the nature of the LTE standard document (specification); it is written in a natural language rather than a formal language, resulting in several ambiguities. Exacerbating the issue, the specification only provides positive testing specifications (*i.e.*, conformance test suites [8, 9]) that mostly verify positive cases to see if valid messages are correctly handled. In other words, the conformance test mostly ignores "**negative testing**", *which examines if invalid or prohibited messages are appropriately handled*. Thus, this may leave many implementation vulnerabilities unchecked.

For this reason, previous studies have attempted several approaches to uncover implementation vulnerabilities. Earlier works focused on a few prohibited messages to check whether devices drop these messages via manual testing [45, 52]. Rupprecht *et al.* [48] proposed the first testing framework to determine whether prohibited algorithms can be selected in devices. LTEFuzz [36] presented the security testing system that examines the message authentication logic for various control plane messages in UEs and network equipment. Despite their successes in discovering implementation vulnerabilities, previous works still fail to comprehensively cover negative cases that are explicitly prohibited by specification. This is because they rely on only a small part of specification. We believe that manual, yet detailed efforts to understand the specification are unavoidable because the specification, written in informal and ambiguous forms, represents 13 years of LTE history, including endless discussions among 3GPP representatives.

In this paper, we propose DoLTEst, a negative testing framework for LTE, which can comprehensively test a user equipment (UE) (*i.e.*, devices) based on specification. Unlike other works, it supports an enumerable number of test cases

*These two authors equally contributed.

with a deterministic oracle that describe standard-compliant behaviors for each negative test case. Our approach inherently involves a significant amount of manual efforts to analyze the specification to find implementation flaws that are not compliant with it.

Despite our best efforts, it is nearly impossible to fully understand the specification due to its ambiguities and high complexity. To address this issue, DoLTEst uses the following approaches. First, we re-define UE states based on a security context, which changes UE's protocol flows. Second, with respect to this new definition, we generate *a guideline* by carefully analyzing the specification. The guideline is a manually-written rule that specifies message types and contents to generate negative test cases. To tame ambiguities in the specification, DoLTEst generates test cases over-approximately from the guideline. We then run negative testing with our initial test cases against various UEs (*i.e.*, 43 UEs). Our basic expectation is that every test case should be silently dropped because the guideline is designed to produce only negative cases. If any UE does not silently drop a certain case, we re-check the specification and even discuss with a 3GPP representative to confirm a standard-compliant behavior. As a result, we can refine these over-approximated test cases to obtain enumerable negative test cases with the deterministic oracle. Such a procedure for building test cases and oracle is labor-intensive; however, this is a one-time cost, and other implementations can easily adopt DoLTEst without re-spending such efforts that we already put. Finally, we hope that our test suite could be standardized to fill the gap of negative testing in conformance specification.

Using DoLTEst, we tested 43 cellular devices (from seven device manufacturers that use baseband processors from the top five major baseband manufacturers). We generated a total of 1,848 test messages considering the abstracted UE states. Via manual root cause analysis, we found 26 implementation flaws, of which 22 were not previously reported. These implementation flaws can directly lead to critical vulnerabilities such as eavesdropping, authentication bypass, or information leakage. We validated attack scenarios by exploiting the above vulnerabilities, including network identity & time zone spoofing, SMS injection, and traffic eavesdropping.

We also discuss our experience analyzing ambiguities in specifications. First, we show a few cases, from which we were not able to define the standard-compliant behavior even from the specification. We report our discussion on three ambiguity issues with a representative from 3GPP. Second, we report a few example cases, where the specification does not clearly define the UE's behavior, leading to deviant behaviors among our test UEs. Later, we show how such behavior can be used to fingerprint baseband chipsets. Finally, considering the size of the specification, one may think that the use of NLP (Natural Language Process) could be useful to solve our problem. We discuss why this could be challenging, even with the recent progress in NLP technologies.

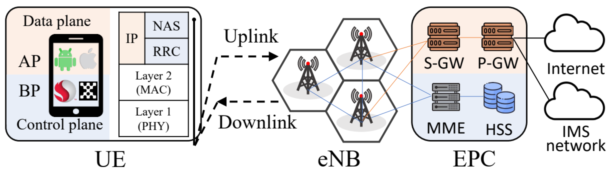


Figure 1: LTE network architecture

Contributions. DoLTEst is the first comprehensive negative testing framework for LTE downlink implementations to find non standard-compliant vulnerabilities in UEs. DoLTEst supports an enumerable number of test cases with a deterministic oracle that describes standard-compliant behaviors for each negative test case through the manual analysis on the specifications of NAS and RRC. Using a total of 1,848 test cases, we tested 43 cellular devices spanning 5 major baseband manufacturers to uncover 26 implementation flaws, including 22 previously unknown ones. Furthermore, we discuss our experience analyzing ambiguities in the specification in detail.

Scope. DoLTEst focuses on finding implementation vulnerabilities for downlink message processing in UEs. In particular, DoLTEst aims at finding non standard-compliant bugs for message authentication, without considering standard vulnerabilities or memory corruptions. DoLTEst focuses on control plane messages in LTE (*i.e.*, NAS and RRC). DoLTEst also adopts generic adversarial models of LTE network, which have no knowledge of cryptographic keys to bypass integrity checks in LTE.

Responsible Disclosure. We have responsibly disclosed all vulnerabilities that we found. We reported each to the corresponding (device and baseband) manufacturer since 2019 and are actively working with them for patching.

Availability. DoLTEst is publicly available on https://github.com/SysSec-KAIST/DoLTEst.

2 Background

2.1 LTE Network Architecture

The LTE network comprises three components: a UE, evolved Node B (eNB), and evolved packet core (EPC) (see Fig. 1).

UE refers to any device located at the edge of a cellular network that provides voice and data cellular services to end-users. It embeds an application processor (AP) that handles the operating system and application services, and a baseband processor (BP) that is responsible for mobile communications, including radio/digital signal processing, identity management, cryptographic protection, and network authentication.

eNB provides a wireless connection between the UE and the EPC as a base station. Furthermore, for secure and reliable cellular service, it manages radio resources and wireless connections while offering confidentiality and integrity protection to control plane messages and user plane data for each UE using a dedicated Radio Resource Control (RRC) protocol.

EPC comprises a mobility management entity (MME) for

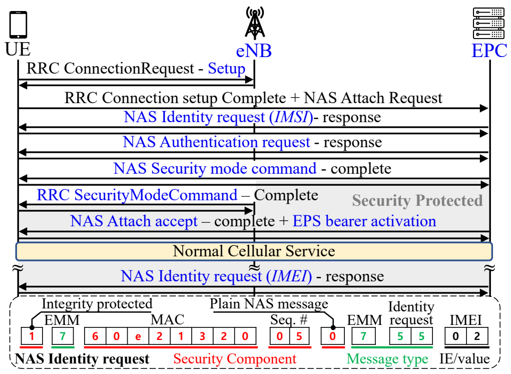

UE
eNB
EPC
RRC ConnectionRequest - Setup
RRC Connection setup Complete + NAS Attach Request
NAS Identity request (IMSI)- response
NAS Authentication request - response
NAS Security mode command - complete
RRC SecurityModeCommand – Complete
Security Protected
NAS Attach accept – complete + EPS bearer activation
Normal Cellular Service
NAS Identity request (IMEI) - response
Integrity protected
Plain NAS message
EMM
MAC
Seq. #
EMM
Identity request
IMEI
1 7 6 0 e 2 1 3 2 0 0 5 0 7 5 5 0 2
NAS Identity request
Security Component
Message type
IE/value

Figure 2: LTE control plane procedure and message structure

user authentication and key/session/identity management, gateways (GW) for managing IP data traffic, and a home subscriber server (HSS) for storing the authentication information of the mobile subscribers (*e.g.*, international mobile subscriber identity (IMSI) and international mobile station equipment identity (IMEI)). In particular, the MME communicates with the UE via the non-access-stratum (NAS) protocol.

2.2 Control Plane Operation

Attach procedure. The `ATTACH` procedure is the mandatory initial process for a UE to use cellular services when it is powered on or returns from airplane mode. The `ATTACH` procedure comprises several NAS and RRC procedures (Fig. 2). Initially, the UE establishes a radio connection with an eNB by exchanging `RRC Connection Request-Setup` messages. Secondly, the MME and UE establish the NAS security context by performing the EMM common procedures (*i.e.*, identification, authentication, and security mode control). Note that these procedures are initiated by the MME. In the identification procedure, the MME asks the UE to send its identifier by exchanging `Identity Request-Response` messages. In the authentication and key agreement (AKA) procedure, the UE and the MME mutually authenticate each other by exchanging `Authentication Request-Response`. The MME then negotiates the security algorithms used for encryption and integrity protection by exchanging `NAS Security Mode Command-Complete`. After establishing a security context, NAS messages are encrypted and integrity protected. Thereafter, the eNB and the UE establish the RRC security context by exchanging `RRC SecurityModeCommand-Complete`. Finally, the MME sends an `Attach Accept` message to inform that the `Attach Request` is accepted. This involves the Evolved Packet System (EPS) bearer activation, which is required for the data plane service. After the UE replies with `Attach complete` message, the `ATTACH` procedure completes.

Control plane message structure. Control plane messages comprise three parts: *message types*, *information elements (IEs)*, and *security components*. Each message type has its own functional purposes within the control plane procedure. For example, an `Identity Request` message type is used by the NAS identification procedure, and the RRC

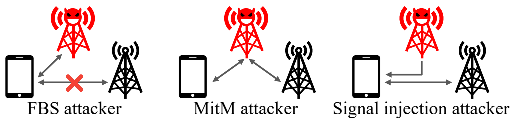


Figure 3: Threat models in LTE

`UEInformationRequest` message type is sent by eNB to request UE reports. Depending on the *message type*, a message can contain a single *IE* or multiple *IEs*. IEs carry certain values according to the defined length and the value type. When the message is constructed, it is encapsulated by *security components* that are used for integrity protection and encryption. These include a security header type, a message authentication code (MAC), and a sequence number. The security header type defines the level of protection in accordance with the following values: `0`–*no security protection*, `1`–*integrity protected*, and `2`–*Integrity protected and ciphered*.[1] Note that messages having security header type `0` are referred to as plain, unprotected, or unauthenticated messages, whereas messages having other security header types (`1-5`) are referred to as protected or authenticated if they have a valid MAC. The message in Fig. 2 shows the message type as `0x55` indicating that this is an `Identity Request`, and the *IE* value as `0x02` meaning that the requested identity type is IMEI. Because the security header type has a value of `1`, the message contains a MAC and sequence number to provide integrity protection.

2.3 Attack Models for UE in LTE

There are three representative active attack models in LTE network: Fake base station (FBS), Man-in-the-Middle (MitM), and signal injection Fig. 3. Because the adversaries in all three threat models have no valid cryptographic key of the victim, they can craft only plain messages or messages with a wrong MAC value. However, since the FBS attacker cannot help a UE to establish the security context with the network, it can send messages to the victim UE only before AKA procedure, whereas MitM and signal injection attacker can send messages even after AKA procedure. The operational logics of the attack models are detailed in App. A.

3 Problems of Prior Negative Testing

3.1 Lack of Negative Testing in Specification

Currently, *conformance specification* [8] is used to test UE implementations. This specification focuses on UE's correct behaviors; the UE handles valid control plane messages properly. This mainly entails *positive testing* to handle non-erroneous cases. Unfortunately, the specification rarely includes invalid or prohibited messages (*i.e.*, *negative testing*) although these messages are crucial for security. For example, if the UE blindly accepts messages without authentication, it can lead to serious attacks such as impersonation or information leakage. Therefore, it is important to test whether a UE successfully

[1]Security header type values `3`, `4`, `5`, and `12` are dedicated for certain message types [4].

drops such invalid messages related to security.

To demonstrate that the existing conformance specification has insufficient negative cases for security, we count the number of test scenarios that explicitly define the corresponding action to the message having prohibited IE/value or the message without integrity protection in the current conformance specification [8, 9]. We examined the specification of version 15.5.0. Among 993 test scenarios, we found that only 14 negative test cases; 3 and 11 cases are related to RRC and NAS, respectively. Even worse, the existing test scenarios only cover limited message types. For example, in RRC, the conformance testing covers only two message types for security. These scenarios check whether 1) the UE uses a correct security algorithm for integrity check and encryption (`SecurityModeCommand`) and 2) the UE discards the `UECapabilityEnquiry` message that has an invalid MAC. Moreover, these test scenarios fail to cover 1) unsafe behaviors explicitly prohibited by the specification, and 2) known implementation vulnerabilities reported by previous works [36, 45, 48, 52].

3.2 Limitations of Previous Works

Although there have been various approaches [36, 45, 48, 52] for negative testing to discover vulnerabilities in UE implementation, they all have the following limitations.

(1) Stateless testing. Existing works only consider limited states in LTE network. In particular, they only consider states where the FBS attacker has a chance to transmit the adversarial messages to the UE (*i.e.*, before establishing a security context). However, recent works have shown successful attack scenarios using new threat models, such as MitM [49, 53] and signal injection [59]. Such attackers are more powerful to transmit adversarial control plane messages even after the UE establishes its security context. Considering that the UE's operation is stateful, and the advanced threat model can send message at any state of UE, stateful negative testing is required. Unfortunately, existing techniques fail to reflect such powerful attackers in their negative testing.

(2) Limited coverage in negative messages. Existing works only focus on limited components in building negative messages. Remember that the control plane message consists of three parts: message type, IEs, and security components.Due to its large search space, the state-of-the-art tool, LTEFuzz, decides to mutate its IEs by randomly substituting values from commercial network logs. This approach is effective to avoid nonsense values that will be early-rejected. However, these real-world logs are network-dependent; they never contain prohibited messages or rarely include infrequently used messages by the network. For example, `Security Mode Command` messages in commercial networks never contain the security algorithms, EIA4–EIA7, whose behaviors are undefined and reserved for future extension. In addition, it would hardly contain the prohibited security algorithm, EIA0 (null integrity). Both cases are important to discover an integrity bypass vulnerability [48]. Unfortunately, LTEFuzz fails to explore messages that ordinary networks would not transmit.

4 Overview

4.1 Goals

The main goal of DoLTEst is to build a negative testing framework that 1) can test a UE comprehensively based on specification 2) using deterministic oracle that can interpret test results against 3) enumerable test cases. We hope that our test framework could be standardized to complement the lack of negative testing in conformance specifications.

Comprehensive testing. DoLTEst should have comprehensive test cases, to thoroughly examine the prohibited or invalid situations that the UE can encounter. To this end, it needs to consider both states and test messages simultaneously because each message's behavior depends on the current state of the UE. Moreover, unlike other works, DoLTEst should consider every part of message components: message type, IE, and security components. It enables DoLTEst to reflect the most powerful adversary from the Dolev-Yao model [19]; the adversary can overhear, intercept, and synthesize any message and is only limited that she has no valid cryptographic key of the victim. This includes all existing attacks in a cellular network such as FBS, MitM, and signal injection.

Deterministic oracle. DoLTEst should support deterministic oracle for each test case to device which behavior is standard-compliant. This is essential for a standardizable test suite to understand the correct behavior for the test case without any exception.

Enumerable test cases. Finally, we want DoLTEst to generate only an enumerable number of test cases. This property allows developers to efficiently test their implementations and to easily understand the root causes of discovered bugs. Thus, DoLTEst is desired to choose the significant ones among nearly infinitely many candidates for negative test cases.

4.2 Challenges in Negative Testing

To achieve the previously mentioned goals, we need to address the following challenges.

C1: Security-irrelevant state definition in specification. To fully evaluate a UE's security, negative testing should consider its various states; however, existing definitions of states are improper for security testing. In particular, the existing specification explicitly defines `REGISTERED/DEREGISTERED` and `IDLE/CONNECTED` states for NAS and RRC, respectively. Unfortunately, these definitions only describe UE connectivity without considering their security contexts. Moreover, specifications also describe implicit states to differentiate UE's behavior based on security contexts. For instance, a UE should not reply to an `Identity Request` that asks for IMEI before the security activation, implying implicit states. However, because of this implicitness, manufacturers implement these states arbitrarily as long as they can pass conformance test specifications. In a nutshell, existing definitions of both explicit and implicit states make negative testing hard to 1) enumerate test cases for stateful security testing, and 2) test

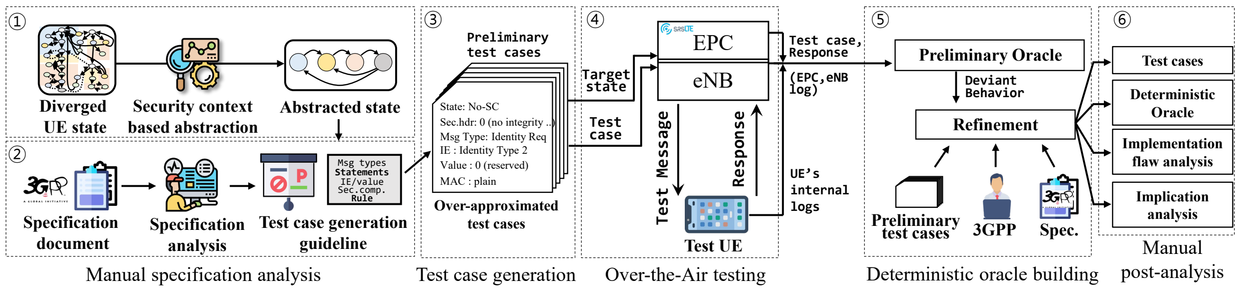


Figure 4: Overview of DoLTEst

multiple UEs regardless of their underlying implementations.

C2: Non-enumerable negative cases. It is infeasible to test all possible negative cases because there can be an enormous number of different LTE messages. In particular, the RRC and NAS have 118 message types, and each message has corresponding optional *IEs* with their *values*, whose numbers are more than a thousand. Therefore, considering all components — message types, IEs and values — they comprise a huge number of combinations. Note that the number of candidates becomes even larger if we perform stateful testing.

Meanwhile, each trial for negative testing in UEs is expensive. Due to the nature of over-the-air testing, it is hard to send a large number of test messages. Also, when UE fails the attach procedure multiple times within a certain period, it disconnects from the test network and suspends reconnection for a few seconds. Note that such a case frequently happens in the negative testing because it can interfere the normal attach procedure due to their abnormal cases. Thus, we need to carefully select only essential cases for negative testing.

C3: Ambiguities in complicate specification. It is difficult to determine the UE's correct behavior when receiving each test case, even if we can enumerate all negative test cases. This is mainly because the specification has many ambiguous statements and descriptions of the operations are spread over multiple documents, each of which are several hundred pages long. For each test case, the oracle needs to determine implementation correctness based on the corresponding behavior. However, we found that specification ambiguously describes whether the test case is actually prohibited or invalid (§6.2). Additionally, the statements regarding a single control procedure are defined at different places over multiple documents. This makes developers misunderstand the specification even with their best efforts. Therefore, building oracle based solely on the specification is challenging, considering the ambiguities and the size of the specifications.

4.3 Our Approach

A1: Defining security abstracted states. To address **C1**, we abstract the implicit states relevant to security based on the specification. In particular, we focus on parts in specification that define the actions of the UE depending on its authenticated states. Our key intuition comes from the fact that the specification defines the acceptance of control plane messages depending on the existence of a security context. Thus, the four abstracted states are categorized by the existence of security context in RRC and NAS. Note that these states are implementation-independent and are fully controllable at the network, meaning that we can use this definition for negative testing regardless of the underlying UE implementations.

A2: Specification-driven test message generation. To reduce the huge search space in negative testing, DoLTEst generates test cases based on the specification, which includes every essential point for negative cases. However, it is nearly impossible to completely understand the specification due to its size and complexity. To address this, we first extract guidelines from the specification by carefully analyzing it, and DoLTEst over-approximately generates test cases based on the guidelines.

A3: Building deterministic oracle. We first assume that negative test messages from DoLTEst should be silently dropped. Then, we validate this by evaluating each message to an extensive set of devices (see Tab. 6). If we discover deviant behaviors from our assumption, we double-check the specification to refine the initial oracle (§5.4). It is worth noting that this refinement is possible since DoLTEst only has an enumerable set of test cases.

5 Design

Fig. 4 illustrates the workflow of DoLTEst. Remember that DoLTEst aims at constructing negative test cases with deterministic oracle. To this end, DoLTEst performs the following six steps. ① First, we redefine the existing implicit UE states as security abstracted states for negative testing (§5.1). ② With this new state definition, we construct rules for test case generation, called *guidelines*, by carefully analyzing the specification (§5.2.2). ③ Then, DoLTEst generates test cases according to the guidelines. To address ambiguities in the specification, DoLTEst enumerates test cases in an over-approximated manner (§5.2.3). ④ DoLTEst conducts preliminary over-the-air testing using the over-approximated test cases. For testing, DoLTEst uses its test EPC/eNB for transmitting a test message to a UE at a specific state. Through the testing, DoLTEst collects the outputs (*e.g.*, responses and a UE's internal logs) for each test UE. Using these outputs, DoLTEst checks whether the test UE shows deviant behaviors (§5.3). ⑤ DoLTEst refines its preliminary test cases and oracle. If a UE shows a deviant behavior in the preliminary testing, we re-check the specification and even discuss with a

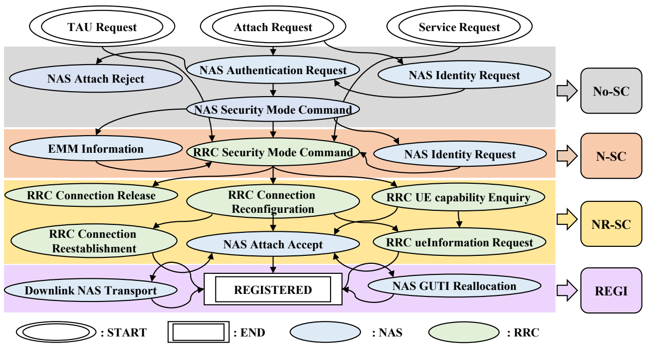


Figure 5: An abbreviated state machine and security abstracted states for RRC and NAS protocol

representative from 3GPP to understand a standard-compliant behavior. ⑥ If this behavior is not buggy, we modify our oracle or eliminate non-negative test cases from our over-approximated set (§5.4). Otherwise, we manually analyze this implementation flaw (§6) and its implications (§7). After this refinement process, DoLTEst can finally obtain negative test cases with deterministic oracle.

5.1 Security Abstracted States

We redefine states in the LTE network for negative testing based on the presence of NAS and RRC security contexts: **No-SC**, **N-SC**, **NR-SC**, and **REGI**. As shown in Fig. 5, one can define a UE's state machine using control plane messages from the network (left side). However, this representation can incur a state explosion if we assume that a UE can receive an arbitrary message in any state. Note that this is a valid assumption in adversarial settings. Moreover, this state machine is independent of the existence of security contexts. Thus, DoLTEst newly defines a UE's state based on security contexts (right side in Fig. 5). This security abstracted state allows DoLTEst to explore different authentication logic, which is changed by the existence of security contexts. In the following, we discuss each state in our new definition.

- **No-SC**: In this state, only an initial radio connection is established between the UE and eNB without any security context. Therefore, all control plane messages in this state remain unprotected.
- **N-SC**: After completing the AKA procedure, a UE enters this state, and the NAS security context is established. In this state, all the NAS messages exchanged between them are protected. However, RRC messages are still unprotected because the RRC security context is not yet established.
- **NR-SC**: The UE enters this state after RRC security context is established alongside the NAS security context. Thus, all RRC and NAS messages are protected in this state. Notably, an FBS attacker cannot attack a UE in this state, as she cannot help the UE to establish the security context with the network.
- **REGI**: The UE finally enters this state when it completes the entire registration process with the LTE network. This state is the same with `REGISTERED` state defined in the specification. A UE in this state is successfully established as a data bearer (EPS bearer); hence, it can use cellular services such as calling, and data services.

Table 1: Abstracted states

	No-SC	**N-SC**	**NR-SC**	**REGI**
NAS security context	✗	✓	✓	✓
RRC security context	✗	✗	✓	✓
ATTACH accomplishment	✗	✗	✗	✓
Reflected Threat model	F, M, I	F, M, I	M, I	M, I

∗ F: FBS, M: MitM, I: Signal Injection

Compared to the existing explicit or implicit state definition, this new definition entails the following properties:

Security-oriented. These abstracted states reflect the nature of LTE security, where the existence of different security contexts allows different authentication logic for control plane messages. To comprehensively evaluate such logic, our first three states (*i.e.*, **No-SC**, **N-SC**, and **NR-SC**) reflect the implicitly-defined states of NAS/RRC with various security contexts. We further consider the state, **REGI**, to reflect the explicit state, REGISTERED, in the specification. Although it has the same security contexts with **NR-SC**, it is reasonable to consider **REGI** specially because it has unique behaviors for certain messages defined by the specification. Tab. 1 shows how each threat model is associated with the abstracted states.

Implementation agnostic. This new definition for abstracted states helps us to accomplish implementation-agnostic testing. As these states reflect fundamental authentication procedures for LTE, we can explicitly control transitions between states by following standardized LTE authentication steps. For instance, we can change a UE's state from **N-SC** to **NR-SC** by sending an `RRC SecurityModeCommand` message from our testing network. Therefore, DoLTEst enables stateful testing for multiple devices regardless of implementation (§5.3).

5.2 Specification-driven Message Generation

DoLTEst generates test messages that are invalid or prohibited by specification in a *selective yet approximated* manner. Basically, our approach is best-effort; as shown in Fig. 4, we first carefully analyze specifications to build a rule for test case generation, called *a guideline*.

5.2.1 Manual Specification Analysis

We manually read the entire RRC and NAS specification to create guidelines. In particular, we focused on statements related with message authentication, because DoLTEst aims at discovering security issues in LTE (please see our GitHub repository for the representative statements used to make guidelines [55]). However, these statements are not solely sufficient; we also need to refer to other parts of the specification to create guidelines (e.g., need to understand corresponding IEs for messages).

To review our analysis, we leverage two auxiliary methods. First, we rechecked the paragraphs around the security-critical words — 'shall not', 'security activation', and 'integrity protection' —. These words are essential to describe message authentication, which DoLTEST focuses on. Second, we compared different versions of the specification. This is based on our intuitions that 1) revision could happen due to security patches, and 2) new features are likely to have flaws during implementing them.

Despite our best efforts, it is nearly impossible to completely understand the specification. Therefore, the guideline allows to specify ambiguities, and DoLTEST generates test cases by over-approximately interpreting such ambiguities. This over-approximation often leads to include exceptions in its test cases (i.e., non-negative test cases); therefore, DOLTEST re-validates these cases while constructing the oracle (§5.4). Unfortunately, DoLTEST still cannot guarantee any completeness, which will be discussed in §8.

5.2.2 Guideline generation

We carefully analyze the specification and make *guidelines* for negative test case generation. The guidelines are handcrafted rules for test case generation that include every component for test cases, which consist of *message type*, *IE and its values*, *security components (security header type and MAC)*, and *states*. To address ambiguities in the specification, the guideline allows to specify a wildcard for the component that specifies arbitrary value. This procedure to make guidelines is easier than understanding the specification completely.

Message type selection. We first need to choose message types for guidelines. As described in §1, we aim at evaluating security authentication in UEs; therefore, we focus on the messages that the UEs should not accept without valid security protection (*i.e.*, integrity protection). In other words, we exclude messages that are not protected by design (*i.e.*, plain). As a result, we consider 14 message types for testing both NAS and RRC, which can be found in Tab. 3.

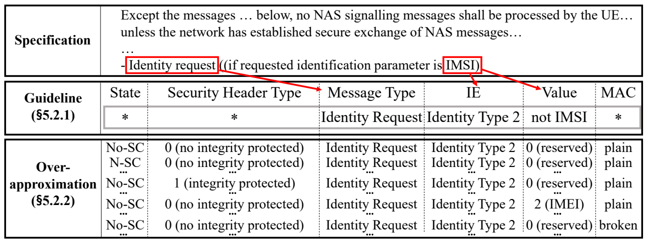
Construction. For each message type, we generate guidelines for negative test case generation. To this end, we first carefully analyze the specification and identify statements that explicitly prohibit certain messages. We then generate guidelines manually by transforming these statements. We have three rules for this transformation.

- If specification does not explicitly mention a certain component, the guideline adds all possible values to its candidates (*i.e.*, wildcard).
- If specification defines invalid or prohibited values for a component, the guideline adds them to its candidates.
- If specification defines allowed values for a component, the guideline adds its complement set to its candidates.[2]

As a result, we generate 17 guidelines from the specification (Tab. 5).

Example. Fig. 6 shows how we generate a guideline from specification using `Identity request` as an example. As shown in the topmost part of Fig. 6, the specification only allows

[2]For example, as shown in Tab. 5, if EIA1, EIA2, EIA3 are allowed, we add test cases for EIA0 and EIA 4 to 7.



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 (§5.2.1)	*	*	Identity Request	Identity Type 2	not IMSI	*
Over-approximation (§5.2.2)	No-SC	0 (no integrity protected)	Identity Request	Identity Type 2	0 (reserved)	plain
	N-SC ...	0 (no integrity protected) ...	Identity Request ...	Identity Type 2 ...	0 (reserved) ...	plain
	No-SC ...	1 (integrity protected) ...	Identity Request ...	Identity Type 2 ...	0 (reserved) ...	plain
	No-SC ...	0 (no integrity protected) ...	Identity Request ...	Identity Type 2 ...	2 (IMEI) ...	plain
	No-SC ...	0 (no integrity protected) ...	Identity Request ...	Identity Type 2 ...	0 (reserved) ...	broken

Figure 6: The guideline for the `Identity Request` test messages. Note that the '*' indicates the wildcard.

specific messages to be processed by the UE without integrity check. One of such messages is `Identity request`; it has an additional restriction that the message's requested identification parameter (*i.e.*, `Identity Type 2`) should be IMSI, whose value is 1. According to the rules described above, we formulate a guideline for `Identity request`. This guideline has a complement set for `Identity Type2` to exclude the specified value (IMSI) from its candidates. Moreover, we set arbitrary values (*i.e.*, wildcards) for remaining components: state, security header type, and MAC. This reflects the above specification; it says that `Identity request` without IMSI should be always protected. In other words, a UE should not accept such a message even with any combination of the remaining components. Therefore, DoLTEST considers their all possible combinations to comprehensively evaluate a UE's authentication logic.

As another example, the fourth guideline in Tab. 5 is derived from the statement "... *the E-UTRAN may configure the UE to perform measurement reporting, but the UE only sends the corresponding measurement reports after successful security activation.*" This configuration is handled by an `RRCConnectionReconfiguration` message with the `measConfig` IE. Therefore, the guideline is used for validating whether the UE checks integrity correctly with this message. To understand this message, one should also look procedure description on measurement configuration (clause 5.5) and message structure (clause 6) in [7].

5.2.3 Over-approximated test case generation

After finishing the guideline generation, DoLTEST enumerates every test message according to the guidelines, as shown in Fig. 6. In particular, for making a test case, DoLTEST chooses one of the candidate values in each component. For information elements and security header components, DoLTEST uses possible values defined in [4] and [3]. For MAC, DoLTEST uses zero (*i.e.*, no integrity) and random (*i.e.*, broken integrity) MAC for their candidates. This step is fundamentally over-approximation because there may be some exceptions in our enumerations, which are not truly negative messages. DoLTEST can discover these exceptions by observing the behaviors of multiple implementations, which are discussed in §5.4. Finally, we generate 1,848 test cases in total from 17 guidelines, which are shown in Tab. 5.

5.3 Over-the-Air Testing

DoLTEst conducts the generated test cases using its over-the-air testing framework.

Testing environment. As shown in Fig. 4, over-the-air testing is composed of the test EPC/eNB and the test UE. To implement the test EPC/eNB, we use a modified version of srsLTE [24] (release 19_09) in combination with an USRP B210 as software-defined radio. The test UE is equipped with a programmable SIM card (ISIM-SJA2 or SIM-SJS1-4FF). It is connected to the testing framework, which uses the diagnostic message (DM) monitoring tool [28, 58] to obtain changes in baseband and Android Debug Bridge (ADB) [11] to obtain system-wide changes. During the test case execution, the radio interface and test UE are encapsulated into a Faraday case [54] to avoid any interference to the commercial network and users.

Testing procedure. For each test case, which consists of a test message and an abstract state, DoLTEst first moves the testing UE's state to the target state, sends the test message, and collects responses for finding bugs. To change the UE's state, DoLTEst follows standard authentication procedures in LTE; the EPC/eNB sends 1) `Security Mode Command` in NAS and RRC for **N-SC** and **NR-SC**, respectively, and 2) `Attach Accept` message for **REGI**. Then, the EPC/eNB sends the test message to UE, and DoLTEst collects the UE's responses (if any), alongside with internal log messages retrieved via DM and ADB. The responses and logs are then used to build the deterministic oracle (§5.4).

Reducing state transition overhead. For efficiency in over-the-air testing, DoLTEst can reduce repetitive state transition overhead. Remember that state transition happens only by special messages such as `Security Mode Command` or `Attach Accept`. In other words, we can assure a UE's state remains the same if it accepts any message that is not one of these special ones. Using this property, DoLTEst makes a UE to execute multiple messages consecutively without going back to the initial state, **No-SC**. This makes DoLTEst exempt from a time-consuming procedure that reboots the UE by invoking airplane mode at every test case execution.

However, it does not mean that DoLTEst can send an unlimited number of test messages at once. The specification defines that every message should be processed in a specific time slot; otherwise, a UE silently releases the connection and increases its attempt counter. If this attempt counter exceeds its threshold, the UE will stay in a disconnected state for 12 minutes (*i.e.*, T3402 [4]), resulting the delay in testing. Thus, DoLTEst limits the number of consecutive test cases. Empirically, the current prototype of DoLTEst only sends *six* messages in a session to avoid this issue.

Testing time. DoLTEst requires several hours to test all 1,848 test messages due to its over-the-air testing. We observed that one session took about 10 seconds, and the overall testing took 8 hours on average; this testing time varies over the tested devices. There are several reasons for such large time consumption. First, DoLTEst needs to wait for seconds to determine whether the UE does not respond to a test message. One of the most common reactions against negative messages is silently rejecting the messages. To distinguish this from a slowdown in delivering responses, we use a timeout — 2 seconds in our prototype — which delays the overall testing. Second, the UE occasionally stops reconnecting to the testing framework after disconnection. This situation lasts for a few minutes due to the radio environment or internal logic of the UE. To escape from it, we sometimes manually invoked the airplane mode to reset. This idle time occupies over 70% of the overall testing.

5.4 Deterministic Oracle Building

DoLTEst builds a deterministic oracle to overcome the ambiguities in understanding specification. In particular, DoLTEst first makes initial assumptions and refine them for building the oracle.

Initial assumptions. DoLTEst initially assumes that a UE should silently drop all our test messages. This is because we construct our test messages that are explicitly prohibited or invalid based on the specification.

Refinement. DoLTEst refines a preliminary oracle based on deviant behaviors in implementations. Below, we show our refinement procedure based on the type of deviant behaviors.

- **If a UE returns a normal response,** it means either 1) the UE has an implementation flaw or 2) the message is exceptional in the specification. To confirm this, we investigate every part of the specification that defines how the UE should behave if it receives the test message. Then, we check whether the current response is standard-compliant or not. If the specification allows this test case as an exception, we eliminate this case from the over-approximated set because it is actually a positive case. Otherwise, we consider it as an implementation flaw (§6.1) and further analyze its implication (§7). Finally, we might not be able to figure out the standard-compliant behavior even after checking the specification, due to its ambiguity. In such a case, we discuss with a representative from 3GPP (§6.2) to understand the standard-compliant behavior.
- **If a UE sends a reject or an error message,** we further check whether the specification mandates to use a specific reject or an error message. If so, we modify the oracle to regard the correct behavior as sending that message, and consider other behaviors as non standard-compliant. If a standard-compliant behavior for the test case is undefined, we refine our oracle to ignore any reject response. Note that this unspecified behavior could be used for device fingerprinting, which will be discussed in §6.3.
- **If a UE has no response,** we further check the UE's internal logs for particular message types. These message types have no response inherently according to the specification. Such message types are `Attach reject`, `EMM information`,

and `RRC ConnectionRelease`. Since these message types can change the UE's internal state, we use information from DM or ADB, which is collected from DoLTEst's testing framework to determine whether the messages are actually rejected (§5.3). If the UE changes its state, similar to the first case, we investigate the specification related to our test case and check the standard-compliant behavior. In App. B, we described the detailed logic for handling these messages. Except for these three message types, other message types have explicit corresponding response messages when they process the transmitted test messages. Thus, for those message types, DoLTEst can determine the UE's rejection of the messages without monitoring UE's state.

During the refinement procedure, we successfully handle messages that return reject responses, including `Security Mode Failure`, `EMM status`, or `RRCConnectionReestablishmentRequest`. Moreover, we also eliminate positive cases from our over-approximation related to `Identity Request`. Lastly, we refined three ambiguities in the specification (§6.2) and two types of unspecified rejecting behaviors (§6.3).

6 Negative Testing Results

We applied DoLTEst on 43 cellular devices from five major baseband manufacturers: Qualcomm, Exynos, MediaTek, HiSilicon, and Intel. We include details for these devices in Appendix (Tab. 6). Using the guidelines in §5.2, we generated 1,848 test messages and conducted the test at security abstracted states in §5.1. In summary, we uncovered 26 implementation flaws using our negative test framework (§6.1). We also uncovered inconsistent error handling among devices (§6.3). Note that such inconsistencies are legitimate because they are undefined in the specification; however, they can also be used for device fingerprinting (§7.2).

Statistics. Among 1,848 test messages, 229 of them have helped DoLTEst to discover flaws. In addition, 83 test messages exhibited inconsistent error handling behaviors as described in §6.3. For the remaining 1,536 messages, all UEs handled them without any problems.

6.1 Implementation Flaws

Using DoLTEst's negative test suite, we could discover 26 implementation flaws. We categorized implementation flaws into nine types based on the problematic behaviors as shown in Tab. 2. The numbers in column 'D' represents the our findings on each flaw type.

Tab. 3 shows uncovered implementation flaws on each message in each security abstracted state. Due to the space limit, we leave our full testing results in the Appendix (Tab. 6). Existing work has shown that some UEs accept a few unauthenticated RRC/NAS messages. While those bugs are patched in newer basebands, we were able to find new variants thanks to our extended coverage. We found a total of 22 cases, among which 4 cases were previously discovered in old basebands. In addition, owing to the stateful testing that considers the various threat models, DoLTEst uncovered four flaw types (M2, S1, S2, and S3) in the state where the UE has security context (*i.e.*, post-AKA).

Table 2: Description of implementation flaw types.

	S: Security header type mishandling	**D**	**L**	**B**	**A**
S1	Accept invalid security header types for certain message types	5	0	0	0
S2	Accept invalid security header type for certain UE states	3	2	0	0
S3	Mishandle reserved security header type	1	0	0	0
	M: Message type mishandling	**D**	**L**	**B**	**A**
M1	Accept prohibited message types before security activation	2	2	0	0
M2	Accept unprotected messages with certain message types after security activation	6	0	0	20
	I: IE/value mishandling	**D**	**L**	**B**	**A**
I1	Accept prohibited IEs	3	1	0	0
I2	Accept prohibited values	3	0	0	0
I3	Mishandle reserved values	3	0	2	0
I4	Mishandle reserved IEs	0	0	47	0

D: DoLTEst, **L**: LTEFuzz, **B**: BaseSpec, **A**: Atomic

To show the effectiveness of our approach, we compared DoLTEst with the recent works — LTEFuzz, BaseSpec, and Atomic — on uncovering baseband implementation flaws. We directly compare with the results reported in these papers, excluding design flaws that are out-of-scope of our work. This is because LTEFuzz [36] and Atomic [15] are not publicly available, and BaseSpec only supports a subset of our tested devices. Plus, due to the anonymity, part of the analysis code for BaseSpec is not publicly available, which makes reproduction non-trivial. The numbers in right four columns indicate uncovered implementation flaws for each flaw type. In particular, the state-of-the-art negative testing tool, LTEFuzz [36] can only cover three types (S2, M1, I1) finding five implementation flaws. This is because, unlike DoLTEst, LTEFuzz is limited to exploring UE states and message components (*e.g.*, negative IE/values and security header types). Other works are specialized in finding specific types of flaws. In particular, Atomic [15] found 20 flaws of the M2 flaw type, which are originated from mishandling three message types. Also, BaseSpec [34] targeted the implementation flaws in message parsing and found numerous flaws of the I3 and I4 flaw types. It uncovered 47 flaws of the I4 flaw type, which are originated from the mishandling of NAS messages having malformed structure. This type of flaw is out of DoLTEst's testing scope. The generated test messages in DoLTEst follow the defined structure by specifications. Compared to other works, DoLTEst could uncover the various types of flaws owing to its over-approximated test case generation and stateful testing.

The exploitability of each implementation flaw heavily depends on its tested state, which is directly mapped to the threat model. In particular, while the implementation flaws in **No-SC** could be exploited by FBS, the implementation flaws in other states only can be exploited by MitM or Signal Injection. Note that each threat model has its own challenges for conducting the attack(s). The FBS attacker needs to operate with higher signal strength to lure the victim. Including the FBS requirement, the MitM attacker requires to maintain radio configuration is additionally. Signal injection attacker

Table 3: Implementation flaws on messages. The numbers in parentheses indicate the number of flaws that have the same message type and states but different details, which results in different implementation flaws. (See §6.1 for the detailed reasons)

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

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

needs to be synchronized to the legitimate eNB, and its signal strength needs to be higher than 3 dB for the capture effect.

In the following, we discuss implementation flaws that DoLTEst can discover in detail.

6.1.1 Mishandling security header type

S1: Accept security header types invalid for certain message types. The 3GPP specification defines the value of the available security header types for each message. For example, the two security header types — *integrity protected with new EPS security context* (`3`) and *security header for the Service Request message* (`12`) — are set to be used only for `Security Mode Command` and `Service Request`, respectively. Therefore, the other message types should not use these dedicated values according to the specification (clause 9.3.1 in [4]).

The S1 flaws are uncovered by checking if devices accept NAS messages having such invalid security header types. We found that every device using Qualcomm baseband accepts any NAS message (except two message types) without checking the integrity if the security header type value is set to `3`.[3] This implies that the adversary can launch all known attacks on NAS (Tab. 3) by using that particular security header type. Considering its severity, Qualcomm has assigned a CVE with a critical security rating (CVE-2019-2289).

S2: Accept invalid security header type for certain UE states. The specification defines that a UE needs to handle security header types properly according to its state. Particularly, in **No-SC**, where the security context is not yet established, a UE should discard a message with the security header types — *integrity protected* (`1`) or *integrity protected and ciphered* (`2`) — because UE cannot calculate a valid MAC. Moreover, a UE should not accept a message with the security header type *not security protected* (`0`) after the security context is established (clause 4.4.4.2 in [4]).

DoLTEst discovered three vulnerabilities when handling `Identity Request` and `Attach Reject`. In particular, we found that iPhone 6 and Galaxy Note 5 responded to any `Identity Request` whose security header type is `1` even before the security context is established (*i.e.*, **No-SC**).[4] This causes information leakage because these UEs send privacy-critical identities (*e.g.*, IMEI or IMEISV) in **No-SC**.

Moreover, we discovered that many UEs respond to the message with security header type `0` after **N-SC**. Our negative testing shows that devices with Exynos (all), devices with MediaTek (all), and devices (LG and Samsung) with Qualcomm baseband contain this implementation flaw. When these devices receive a plain `Identity Request` after **N-SC**, they respond to it with an `Identity Response` message that contains the IMSI. Note that this bug only appears in `Identity Request` message when `Identity type 2` IE is set to the reserved *value* (`0`). It also violates a standard because UEs should not accept plain messages after **N-SC**. Previous work has shown that some UEs *mishandle the IE value* on `Identity Request` [43]. However, our result shows that *mishandling an invalid security header type* also leads to the same problem.

S3: Mishandle reserved security header type. The specification does not define usages for security header types from `6` to `11`, which are reserved for future extension. Plus, if a UE receives security header types ranging from `13` to `15` in a message, it should interpret those as `12`, which is dedicated to the `Service Request` message (clause 9.3.1 in [4]).

DoLTEst discovered that some UEs violate the aforementioned behaviors in the specification (S3). We found that all tested UEs have no problem when the security header types are `6`–`11`; they discard all messages with the header types. However, we found that several UEs fail to handle other reserved security header types. Particularly, devices with Qualcomm (all produced before mid-2017) and one Huawei baseband send an `EMM Status` message when they receive a message with security header type `12`. By contrast, they do

[3]Because we reported this issue in early 2019, the issue was patched on devices released after that.

[4]iPhone 6 with the old firmware (iOS 12.1) does not have this flaw, whereas the same model with the latest firmware (iOS 12.5.1) has.

not respond to messages with security header type `15`. In other words, the UEs behave differently to messages with those two security header types, which should be the same. Meanwhile, other devices do not respond to messages with those security header types, conforming to the standard.

6.1.2 Mishandling message type

M1: Accept prohibited message types before security activation. The cellular system is designed to expose minimal information before security activation. Hence, the specification prohibits to process of certain message types depending on the security context. Thus, UEs should not accept those message types in **NO-SC** (clause 5, Annex 6 in [7]).

We discovered that many UEs accept two prohibited message types in **NO-SC**. First, all devices with the Exynos baseband accept a `CounterCheck` and respond with a `CounterCheckResponse`. Second, devices with MediaTek (1 device), Qualcomm (2 devices), and Exynos (all) basebands accept a `UEInformationRequest` and respond with `UEInformationResponse`. Although M1 was widely tested in previous works [36,52,53], we newly uncovered implementation flaws on `CounterCheck`. The flaw in `UEInformationRequest` was previously known, but we were still able to find multiple unpatched devices.

M2: Accept unprotected messages with certain message types after security activation. Once the security context is established, certain message types should only be used in integrity-protected messages. Thus, UEs should discard these messages if their integrity check fails after **NR-SC** (clause 5.3.1.2 in [7]).

DOLTEST also found that several UEs accept unprotected messages with zero MAC even after **NR-SC**. In particular, we observe that devices with Qualcomm (2 devices) baseband accept every unprotected RRC messages types after **NR-SC**. Due to this fatal implementation flaw, an attacker can spoof every NAS/RRC message even after **NR-SC**. Previous studies found these bugs only in **NO-SC**.

6.1.3 Mishandling IE and value.

I1: Accept prohibited IEs. The specification prohibits a UE to handle the messages that contain security-critical IEs in **NO-SC** and **N-SC**. Thus, the UE should not accept messages containing prohibited IE in a specific state (clause 5 and Annex 6 in [7]).

We found the I1 flaws in five test devices that accept three forbidden IEs. These IEs are used for establishing signaling bearer 2 (`srb-ToAddModList`), data bearer (`drb-ToAddModList`), and requesting signal measurement report from UE (`measConfig`). In particular, three devices with the Qualcomm baseband incorrectly accept a message with the first two IEs. DOLTEST newly discovered the bug associated with the first IE. Also, DOLTEST found that two devices with the latest Exynos baseband accept `measConfig` *IE* and send `Measurement Report` to the network. Interestingly, even though this vulnerability was patched in 2016, it has been introduced again in the latest version. It shows that we require negative testing to prevent such regression bugs.

I2: Accept prohibited values. The standard also explicitly prohibits the use of certain IE values in **N-SC** (subclause 4.4.4.2 in [4], 5.3.1.2 in [7]). We uncovered three implementation flaws of this type in six test devices: i) `Identity Request` message containing an IE value to request the UE's privacy-sensitive identity (IMEI and IMEISV), ii) `Attach Reject` whose reject cause is set to `#25`, iii) `Security ModeCommand` in RRC, which contains null integrity protection algorithm (EIA0).

In particular, we found that the two devices with the latest Exynos baseband accept the EIA0 value in `Security Mode Command` in RRC, which is explicitly prohibited by the standard. Note that the same bug was found in the USB dongle with the Huawei baseband previously [48]. Of note, none of the other basebands from the same manufacturer had this problem. We got high severity from Samsung for this vulnerability.

I3: Mishandle reserved values. The specification contains reserved *values*, whose operation is not clearly defined. Thus, we analyzed UE's behavior against messages with reserved values.

Our negative testing exposed implementation flaws of I3 in three *IE values*. First, UE mishandle reserved security algorithms in NAS and RRC. The security algorithm is a 3-bit value indicating the security algorithm type for the security context; the value from 0 to 3 have dedicated algorithms, whereas the values from 4 to 7 are reserved. We found that devices with Qualcomm (two) and Exynos (one) accept `security mode command` with these reserved values and respond with `security mode complete` containing zero MAC. This implies that those devices nullify the integrity protection on the control plane, which results the same security problem to the use of prohibited value EIA0 (clause 4.4.4.1 in [4], clause 5.3.1.2 in [7]). Second, we also discovered inconsistency among devices in handling the `Identity Request` message. In particular, if a device receives the `Identity Request` message whose `identity type 2` value is zero, devices with Huawei and Intel baseband had no response, whereas other devices respond it with `Identity Response` containing IMSI. By further referring another specification document, we found that the latter is correct implementation because the reserved value `0` should be interpreted as IMSI (clause 10.5.5.9 in [3]).

6.2 Ambiguous Statements in Specification

During our research, we found that the specification contains ambiguous or often conflicting statements, which could be interpreted in multiple ways. In particular, we had to discuss three issues with a representative from 3GPP SA3 to understand the exact intention and historical meaning of the specification. We briefly summarize our discussion below.

CounterCheck. We found that the specification states the handling of the `CounterCheck` message differently in two parts.

In the RRC specification [7], the table in the appendix (Annex 6) specifies that the UE should not accept unprotected `CounterCheck`, whereas the procedure description in the body (clause 5) does not prohibit it. Due to this inconsistency, it is ambiguous whether accepting a `CounterCheck` message before security activation is prohibited. Meanwhile, another specification spells out that `CounterCheck` and its response are integrity protected (clause 7.5 in [6]). According to the representative, Annex 6 is an "informative" annex, so it does not mandate integrity protection of `CounterCheck`. However, he added that because the `CounterCheck` message is used to verify the amount of data sent/received, it is nonsense to use it before the data bearer activation (*i.e.*, before **REGI**). Should this prohibited? Except Samsung, everyone does (§6.1.2).

UECapabilityEnquiry. We also found that it is ambiguous to determine the correct operation of UE against `UECapabilityEnquiry`. This is because this behavior is written only on the network-side without clearly defining the UE's behavior. The specifications after v15.9.0 say that "*E-UTRAN should retrieve UE capabilities only after AS security activation*" for the `UECapabilityEnquiry` message. This means that eNB can request the UE's radio capability information only in **NR-SC** and **REGI**. However, it is unclear that what UE should do when it receives such a message in **NO-SC**. Interestingly, our test results show that all UEs respond in **NO-SC**. Meanwhile, a previous work regards such unauthenticated UE capability [53] as a security vulnerability in terms of information leakage.

The representative said that the specification does not explicitly prohibit the UE to transmit UE capability to eNB before security activation, but only recommends to prohibit. This is because the term "should" means it is highly recommended but not mandatory, whereas "shall" means it is mandatory, according to the specification drafting rule [1]. Also, he added that considering security implications of unauthenticated UE capability transfer procedure, the standard body tried to prohibit the use of `UECapabilityEnquiry` before security activation by using the term "shall." However, due to the performance optimization issue, some network equipment manufacturers opposed the prohibition. Because the current standard allowing exceptional cases with the term "should" is the result of a compromise, UE capability transfer before security activation is not an implementation vulnerability. As a result, we exclude this message during the refinement process.

Identity Request. Lastly, we found an ambiguous description for the `Identity Request` message that contains a reserved value. According to the NAS specification, 1) a UE can respond to a plain `Identity Request` message *if request identification parameter is IMSI* before security activation [4]. Concurrently, another specification [3] indicates that 2) the UE should interpret the reserved value as IMSI [3]. Our testing revealed that UEs with Huawei or Intel baseband did not respond to the plain `Identity Request` message containing the reserved value. When we reported this issue to the Huawei PSIRT (Product Security Incidence Response Team), they argue that having no response is correct behavior, because they think the reserved value is not exactly an IMSI parameter. If it is, they argue that statement 1) should have referred the statement 2). However, according to the 3GPP representative, having a response with IMSI is the correct behavior.

6.3 Unspecified Behaviors in Specification

During the test, we observed that devices exhibit inconsistent behaviors when handling the same messages. Unlike the previous implementation flaws, these inconsistencies are all legitimate because they are unspecified in the specification. However, we believe that such inconsistencies can be used to fingerprint devices, which will be shown in §7.2. After investigating these issues, we summarized them into two types.

Implicit vs explicit reject. UE's rejecting behaviors can be divided into two types: implicit reject and explicit reject. In particular, UEs can silently drop a message without any action if the message is prohibited or invalid (implicit reject). In contrast, they can use certain messages to report errors, which include `Security Mode Reject`, `EMM Status`, and `RRCConnection ReestablishmentRequest` (explicit reject).

We found that rejecting behaviors for certain messages are different depending on manufacturers. For example, devices with Huawei baseband send `EMM status` when they receive a NAS message with invalid security header type `3`. However, devices with old Qualcomm baseband send the message when they receive a NAS message with reversed security header type `12` instead of `3`. Meanwhile, devices with MediaTek, Exynos, Intel, and recent Qualcomm basebands never send that message during our test.

Different causes in explicit reject. We discovered that devices use different *cause*s to report an error. These causes offer a detailed description of the erroneous circumstances. For example, we found that when devices receive `NAS Security Mode Command` message with null integrity protection, devices with MediaTek baseband use cause #23 (UE security capabilities mismatch), whereas others use cause #24 (security mode rejected, unspecified). This implies that the UEs regard the reason for the error differently for the negative messages.

Fundamental reasons for inconsistencies. We argue that the fundamental reason for this inconsistent behavior among test devices is the lack of specifications for handling invalid or prohibited messages. In other words, the specification does not clearly define whether to use error messages or which causes to use when the UE receives certain invalid or prohibited messages. Thus, handling undefined error cases is up to the developers, leading to different implementations for manufacturers. Meanwhile, these inconsistencies enable device fingerprinting attacks (§7.2).

7 Attack Scenarios

In this section, we introduce several representative attacks that can be performed using the implementation flaws discussed in §6. The complete list of possible attacks is shown in Tab. 3. We implemented each attack on a testbed by slightly modifying srsLTE [24] with USRP B210/X310 and validated them on COTS UEs.

7.1 Attacks using Implementation Flaws

7.1.1 NITZ (Network Identity and Time Zone) spoofing

In this attack, an **FBS attacker** can manipulate the date, local time, UTC offset, network name, and daylight saving time of the victim device [45].

Vulnerability. An attacker can bypass security protection verification by altering a security header type to `3` (S1). Under normal circumstances, this message should be transmitted with proper security protection. Although we validated this attack on a Xiaomi Black Shark, all devices with the Qualcomm baseband prior to the Snapdragon X24 are vulnerable.

Attack Procedure. When the victim UE is connected to a malicious eNB, the attacker sends an `EMM Information` message with security header type `3` containing a manipulated time or date. Although the `EMM Information` message contains an invalid MAC value, the victim UE accepts the message.

Impact. When a UE receives this message, it updates its date and time based on the message data, which are controlled by the attacker. This attack is valid while the victim is connected to the FBS. However, if the UE reconnects to the network through `Service Request`, this attack can be sustained because the `EMM Information` message is optional in this process, unlike `Attach Request`. Note that we have not seen the message in the `Service Request` process of our commercial network.

7.1.2 SMS injection

The goal of this attack is to deliver an SMS with manipulated sender ID, timestamp, and text content. The **signal injection attacker** can execute this on a victim UE connected to commercial networks.

Vulnerability. An attacker can perform this attack by combining two vulnerabilities. She first uses the vulnerability (M2) that some devices do not check security protection for RRC messages even after the security activation. Using this, the signal injection attacker can make a UE in **REGI** state to receive a plain RRC message. Then, the attacker can bypass the NAS security context by using the second vulnerability (S1); UEs accept some NAS messages without protection if we use the security header type `3`. We validated this attack on Galaxy S4/S5 with a Qualcomm baseband.

Attack Procedure. The attacker first creates an SMS PDU encoded in GSM 03.40 format [2] containing malicious contents. She then encapsulates the PDU into a plain `DL Information Transfer` message containing `Downlink NAS Transport` with security header type `3`. Afterward, the attacker performs a signal injection attack to transfer the crafted message. To do this, the attacker additionally needs to know the victim UE's radio identifier (RNTI) which can be acquired from identity mapping [32, 49] using downlink sniffing tools [12, 14, 21].

Table 4: Different responses of devices to negative messages.

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	·

• **Request–** **#1**: CounterCheck, **#2**: GUTI reallocation command with security header type 3, **#3**: Identity Request with reserved value (0), **#4**: Security mode command with security header type 3 and reserved value, **#5**: EMM information with security header type 12
• **Response–** ·: No response, $\mathbf{A_1}$: CounterCheckResponse, $\mathbf{A_2}$: GUTI reallocation complete, $\mathbf{A_3}$: EMM status with cause #97 $\mathbf{A_4}$: Identity Response $\mathbf{A_5}$: Security mode reject with cause #24 $\mathbf{A_6}$: Security mode reject with cause #23

Impact. When a victim UE receives this message, it displays a notification the same way as in the case of a normal SMS. Owing to security component mishandling, the victim UE accepts the attacker's plain message while it is connected to the commercial network. Because the LTE device still supports legacy *SMS over NAS* through the `Downlink NAS Transport`, it also accepts malicious SMS messages.

7.1.3 Eavesdropping and manipulating data traffic

In this attack, a **MitM attacker** can eavesdrop or manipulate data plane packets between a victim UE and networks [48].

Vulnerability. An attacker can set the null integrity protection algorithm because the UE accepts the forbidden security algorithm (I2). We validated this attack on Galaxy S10, Note10, and A71 with the latest Exynos chipset.

Attack Procedure. When the victim UE is connected to the malicious eNB, the attacker initiates the normal `ATTACH` procedure with the UE. Although the attacker does not have a valid key, she can pass mutual authentication and the NAS security activation procedure by relaying messages from a legitimate network. Thereafter, the attacker sends an `RRC SecurityMode Command` message with EIA0. After that, the UE uses null integrity protection in its subsequent communication.

Impact. When a device receives a `SecurityModeCommand` message with EIA0, it uses null ciphering and null integrity protection from a subsequent communication (subclause 5.3.1.2 [7]). Because the data plane packets of the wireless end are transmitted in plain text in subsequent communication, an attacker can eavesdrop or manipulate the packets. Note that EIA0, in which an UE accepts EIA0 (Null algorithm) for the integrity checking logic, is an old and well-studied issue. However, DoLTEst uncovered that brand new devices (*e.g.*, Galaxy S10/Note10/A71) still have the same implementation flaw.

7.2 Fingerprinting

The different behaviors of UEs against the same negative message enable an attacker to fingerprint the baseband of the victim device in the wireless domain. To do this, an attacker can leverage two types of responses from UE: non standard-compliant behavior (presence of vulnerability) (§6.1) and inconsistent error handling (§6.3). During the test, we found that more than 80 of our test messages can induce those behaviors.

As shown in Tab. 4, an attacker can classify devices up to the baseband manufacturer level using these different behaviors. Moreover, it is possible to classify some devices with Qualcomm, Samsung, and MediaTek basebands up to the chipset-level, because they have chipset-dependent features. Once the UE is identified, the attacker can further execute targeted attacks by combining known vulnerabilities at the OS-level or baseband-chipset-level [53].

Unlike previous works, our fingerprinting uses inconsistencies in implementations. Previous work used the unauthenticated capability information of users (`Attach Request` and `UECapabilityInformation`) as fingerprint features, which are allowed to be exposed in the air [53]. While the root cause of previous work is due to flaw of the standard design, recent standard tries to protect such information from being exposed before security activation. If UE information is explicitly protected in the specification, those features cannot be used for fingerprinting. However, our features remain valid unless the implementation inconsistency remains.

8 Discussion & Limitations

Adopting NLP. A natural next step seems to adopt the NLP on analyzing the large-sized and complex specifications for test case generation. Accordingly, DARPA also has issued a call for a project [18] that adopts NLP on 3GPP specification analysis. Recently, Chen *et al.* presented a technique to discover a few particular vulnerabilities by applying NLP on the specification analysis [15]. While NLP looks promising, adopting it on generating negative test cases still needs to address several challenges according to our detailed specification analysis. First, it should be capable of extracting the prohibited procedure from the sentences spread over the distinct locations. A single sentence does not solely contribute to extract prohibited operations. During the refinement procedure §5.4, we had to cross-check multiple sentences to understand the context of the procedure and to determine the implementation flaw (§5.4). Note that such a case where the description of a procedure is scattered far away from each other is one of the key limitations of the prior work [15]. Second, the NLP has to handle ambiguities in the specification. In §6.2, we discuss three different cases of ambiguities, all of which seem challenging for NLP to resolve. For example, we observed that the specification only describes the behavior in the network-side perspective without clearly defining the UE's behavior. In summary, unless the NLP addresses these challenges effectively, we believe that the manual analysis on the specification is still required.

Negative testing beyond LTE. Although DoLTEst aims to find UE implementation flaws in the LTE domain, we believe that the overall philosophy and approach could be adopted to the other layers of cellular generations (2G, 3G, and 5G) after specification analysis. In particular, DoLTEst requires the following considerations of new features in 5G [5, 10]; the 5G-SA has 1) newly defined messages (*e.g.*, `REGISTRATION REQUEST`), 2) newly defined or modified IE (*e.g.*, `SUCI`) and its value, and 3) newly added UE's state. This requires changes in our security abstracted states because there is a new RRC state (`RRC INACTIVE`), where the RRC security context is maintained even after the radio connection is suspended. Also, one can implement a negative testing framework based on the publicly available 2G/3G open stack, with another substantial analysis. We leave it as future work.

Limitations. First, DoLTEst inherently requires a lot of manual effort on its building. This is because the specification is written in natural language without being formally described. However, our approach requires only one-time efforts and can be adopted without duplicating the laborious manual analysis. Second, DoLTEst cannot guarantee any completeness because of its manual analysis. Therefore, it would not be surprising if someone can discover additional flaws in baseband firmware even after applying DoLTEst. Third, DoLTEst suffers from inherited limitations of blackbox testing. As blackbox testing, DoLTEst cannot understand the internals of a UE; we simply believe that the UE rejects messages if it has no explicit behaviors (e.g., responses or state changes in DM or ADB). However, it is possible that baseband vendors implement hidden behaviors even though we have not found any indications for this during our testing. Finally, DoLTEst relies on internal logs (*i.e.*, DM, ADB) that are occasionally limited. This is mainly because 1) this access is proprietary to the baseband manufacturers (*e.g.*, Huawei) and 2) the access is not open to all types of UEs (*e.g.*, IoT devices). This characteristic limits the negative testing of DoLTEst in terms of 1) the test coverage (limited test message types) and 2) the root cause analysis using detailed internal information (*e.g.*, stored configuration values). We believe that this limitation will not apply to manufacturers who adopt the negative testing approach, as they should have full access to the UE and its debugging capabilities.

9 Related Work

Implementation flaws in LTE. Researchers have widely adopted over-the-air testing to discover implementation flaws of the LTE network [36,41,45,48,52]. Shaik *et al.* [52] studied security implications when a UE blindly accepts certain prohibited messages due to its implementation or design flaws. Rupprecht *et al.* [48] proposed the first testing framework for LTE baseband and discovered that several UEs accept insecure security algorithms. Kim *et al.* [36] introduced a semi-automatic tool that systematically tests basic security properties of LTE networks, both in uplink and downlink. Similar to these works, DoLTEst also aims at discovering implementation flaws in LTE; however, it widens its coverage by considering UE states and detailed message components based on specification.

Firmware analysis on baseband implementations is also an effective approach to uncover implementation vulnerabilities [22, 34, 42, 57]. Recently, Maier *et al.* [42] proposed emulation-based fuzzing to find memory corruption vulnerabilities in UE implementations. Kim *et al.* [34] examined the standard-compliance of the baseband software by comparing extracted binary-embedded properties with the specification. Unlike DoLTEst, these approaches require firmware-specific analysis. Moreover, static analysis (e.g., BaseSpec [34]) can suffer from false positives due to misprediction of runtime environments.

LTE attack models. Fake Base Station (FBS) has been a predominant attack model for exploiting design or implementation vulnerabilities in LTE, such as IMSI catching [17, 44], fake emergency alert [38], or information leak [53]. Recent works have demonstrated the practicality of MitM attack in LTE networks [48, 49]; Rupprecht *et al.* [49] successfully demonstrate data manipulation attack by exploiting the absence of integrity protection in layer 2. Moreover, Yang *et al.* [59] presented the SigOver attack, which enables an adversary to inject a malicious message to the victim's UE even without a radio connection establishment. To reflect these emerging attacks in our negative testing, DoLTEst adopts the most powerful attack model, which is the Dolev-Yao model, and considers every attack model in our abstracted states.

Design vulnerabilities in LTE. Formal verification has shown its effectiveness in uncovering design flaws in the standards [13, 29, 31]. Hussain *et al.* [29] presented a model-based adversarial testing approach on critical procedures of LTE to detect design flaws. Moreover, Karim *et al.* [33] adopted a property-guided formal verification framework to examine LTE implementations. Unlike these works, DoLTEst targets finding non standard-compliant behaviors in implementations. Some other works also have revealed numerous design vulnerabilities in LTE, including LTE data plane [37, 51], broadcast channel [23, 30, 38], identity mapping logic [32, 49], and VoLTE domain [35, 39].

Operational issues in LTE. Various works have uncovered operational issues by analyzing the control plane procedures in the cellular network [16, 27, 28, 37, 47, 50, 56]. These works span various applications and use passive analysis on commercial logs. Tu *et al.* [56] proposed a signaling diagnosis tool to examine inter-protocol interactions. Hong *et al.* [27] revealed an unsafe identity management scheme of the commercial network, resulting in privacy leakage. Chlosta *et al.* [16] analyzed the security configuration of commercial LTE networks.

10 Concluding Remarks

Despite numerous implementation vulnerabilities reported, a lack of negative testing in specification still leaves other implementation vulnerabilities unchecked. To address this, we present DoLTEst, a negative testing framework to uncover the non standard-compliant behaviors in LTE implementations. After in-depth analysis of the specification, we build enumerable test cases with deterministic oracle along with the over-the-air testing tool. As a result, we applied DoLTEst on 43 devices from five baseband manufacturers and uncovered 26 implementation flaws, of which 22 were previously unknown. We also found several ambiguous and unspecified cases in the specification that can confuse even experienced professionals. We recommend 3GPP to provide formally verified standards possibly with sample code and negative test cases as a fundamental resolution to address this problem. Until then, applying NLP for automatic test case generation is left as an interesting yet formidable challenge for the future work.

Acknowledgement

We sincerely appreciate our shepherd Marius Muench and anonymous reviewers, for valuable comments and suggestions. This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2020-0-00428, automated security diagnosis framework for cellular network protocol using formal and comparative analysis, and No.2019-0-01343, regional strategic industry convergence security core talent training business)

References

[1] 3GPP. TR 21.801, v17.1.0. Specification drafting rules, 2021.

[2] 3GPP. TS 23.040, v15.3.0. Technical realization of the Short Message Service (SMS), 2019.

[3] 3GPP. TS 24.008, v16.3.0. Mobile radio interface Layer 3 specification; Core network protocols; Stage 3, 2019.

[4] 3GPP. TS 24.301, v16.5.1. Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS); Stage 3, 2020.

[5] 3GPP. TS 24.501, v16.3.0. Non-Access-Stratum (NAS) protocol for 5G System (5GS); Stage 3, 2019.

[6] 3GPP. TS 33.401, v16.3.0. Security architecture, 2020.

[7] 3GPP. TS 36.331, v15.10.0. LTE RRC Protocol specification, 2020.

[8] 3GPP. TS 36.523-1, v15.5.0. User Equipment (UE) conformance specification; Part 1: Protocol conformance specification, 2019.

[9] 3GPP. TS 36.523-3, v15.5.0. User Equipment (UE) conformance specification; Part 3: Test suites, 2019.

[10] 3GPP. TS 38.331, v16.3.1. 5G NR RRC Protocol specification, 2021.

[11] Android debug bridge. https://developer.android.com/studio/command-line/adb.

[12] AirScope. http://www.softwareradiosystems.com/tag/airscope.

[13] D. Basin, J. Dreier, L. Hirschi, S. Radomirovic, R. Sasse, and V. Stettler. A formal analysis of 5G authentication. In *ACM Conference on Computer and Communications Security*, 2018.

[14] N. Bui and J. Widmer. OWL: A reliable online watcher for lte control channel measurements. In *All Things Cellular: Operations, Applications and Challenges*, 2016.

[15] Y. Chen, Y. Yao, X. Wang, D. Xu, X. Liu, C. Yue, K. Chen, H. Tang, and B. Liu. Bookworm game: Automatic discovery of LTE vulnerabilities. In *IEEE Symposium on Security and Privacy*, 2021.

[16] M. Chlosta, D. Rupprecht, T. Holz, and C. Pöpper. LTE security disabled: misconfiguration in commercial networks. In *Conference on Security and Privacy in Wireless and Mobile Networks*, 2019.

[17] A. Dabrowski, N. Pianta, T. Klepp, M. Mulazzani, and E. Weippl. IMSI-catch me if you can: IMSI-catcher-catchers. In *ACM Conference on Computer and Communications Security*, 2014.

[18] Open Programmable Secure 5G (OPS-5G). https://sam.gov/opp/6ee795ad86a044d1a64f441ef713a476.

[19] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on information theory*, 1983.

[20] Ericsson2021report. "https://www.ericsson.com/en/mobility-report/dataforecasts/mobile-subscriptions-outlook".

[21] R. Falkenberg and C. Wietfeld. FALCON: An accurate real-time monitor for client-based mobile network data analytics. In *IEEE Global Communications Conference*, 2019.

[22] N. Golde and D. Komaromy. Breaking band: reverse engineering and exploiting the shannon baseband. *REcon*, 2016.

[23] N. Golde, K. Redon, and J.-P. Seifert. Let me answer that for you: Exploiting broadcast information in cellular networks. In *USENIX Security Symposium*, 2013.

[24] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, and D. J. Leith. srsLTE: An Open-Source Platform for LTE Evolution and Experimentation. In *ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization*, 2016.

[25] LTE and 5G Subscribers. "https://gsacom.com/paper/lte-and-5g-subscribers-march-2021-q4/".

[26] 5G implementation guidelines. "https://www.gsma.com/futurenetworks/wp-content/uploads/2019/03/5G-Implementation-Guideline-v2.0-July-2019.pdf".

[27] B. Hong, S. Bae, and Y. Kim. GUTI reallocation demystified: Cellular location tracking with changing temporary identifier. In *Network and Distributed System Security Symposium*, 2018.

[28] B. Hong, S. Park, H. Kim, D. Kim, H. Hong, H. Choi, J.-P. Seifert, S.-J. Lee, and Y. Kim. Peeking over the cellular walled gardens-a method for closed network diagnosis. In *IEEE Transactions on Mobile Computing*, 2018.

[29] S. Hussain, O. Chowdhury, S. Mehnaz, and E. Bertino. LTEInspector: A systematic approach for adversarial testing of 4G LTE. In *Network and Distributed System Security Symposium*, 2018.

[30] S. R. Hussain, M. Echeverria, O. Chowdhury, N. Li, and E. Bertino. Privacy attacks to the 4G and 5G cellular paging protocols using side channel information. In *Network and Distributed System Security Symposium*, 2019.

[31] S. R. Hussain, M. Echeverria, I. Karim, O. Chowdhury, and E. Bertino. 5GReasoner: A property-directed security and privacy analysis framework for 5g cellular network protocol. In *ACM Conference on Computer and Communications Security*, 2019.

[32] R. P. Jover. LTE security, protocol exploits and location tracking experimentation with low-cost software radio. *arXiv preprint arXiv:1607.05171*, 2016.

[33] I. Karim, S. Hussain, and E. Bertino. ProChecker: An automated security and privacy analysis framework for communication protocol. In *IEEE International Conference on Distributed Computing Systems*, 2021.

[34] E. Kim, D. Kim, C. Park, I. Yun, and Y. Kim. BASESPEC: Comparative analysis of baseband software and cellular specifications for L3 protocols. In *Network and Distributed System Security Symposium*, 2021.

[35] H. Kim, D. Kim, M. Kwon, H. Han, Y. Jang, D. Han, T. Kim, and Y. Kim. Breaking and fixing VoLTE: Exploiting hidden data channels and mis-implementations. In *ACM Conference on Computer and Communications Security*, 2015.

[36] H. Kim, J. Lee, E. Lee, and Y. Kim. Touching the untouchables: Dynamic security analysis of the LTE control plane. In *IEEE Symposium on Security and Privacy*, 2019.

[37] K. Kohls, D. Rupprecht, T. Holz, and C. Pöpper. Lost traffic encryption: fingerprinting LTE/4G traffic on layer two. In *Conference on Security and Privacy in Wireless and Mobile Networks*, 2019.

[38] G. Lee, J. Lee, J. Lee, Y. Im, M. Hollingsworth, E. Wustrow, D. Grunwald, and S. Ha. This is your president speaking: Spoofing alerts in 4G LTE networks. In *ACM International Conference on Mobile Computing Systems (MobiSys)*, 2019.

[39] C.-Y. Li, G.-H. Tu, C. Peng, Z. Yuan, Y. Li, S. Lu, and X. Wang. Insecurity of voice solution VoLTE in LTE mobile networks. In *ACM Conference on Computer and Communications Security*, 2015.

[40] M. Lichtman, R. P. Jover, M. Labib, R. Rao, V. Marojevic, and J. H. Reed. LTE/LTE-A jamming, spoofing, and sniffing: threat assessment and mitigation. *IEEE Communications Magazine*, 2016.

[41] H. Lin. LTE REDIRECTION: Forcing targeted LTE cellphone into unsafe network. In *Hack In The Box Security Conference (HITBSec-Conf)*, 2016.

[42] D. Maier, L. Seidel, and S. Park. BaseSAFE: baseband sanitized fuzzing through emulation. In *Conference on Security and Privacy in Wireless and Mobile Networks*, 2020.

[43] B. Michau and C. Devine. How to not break lte crypto. In *ANSSI Symposium sur la sécurité des technologies de l'information et des communications (SSTIC)*, 2016.

[44] P. Ney, I. Smith, G. Cadamuro, and T. Kohno. SeaGlass: enabling city-wide IMSI-catcher detection. *Privacy Enhancing Technologies*, 2017.

[45] S. Park, A. Shaik, R. Borgaonkar, and J.-P. Seifert. White rabbit in mobile: Effect of unsecured clock source in smartphones. In *Workshop on Security and Privacy in Smartphones and Mobile Devices*, 2016.

[46] D. PAULI. Samsung S6 calls open to man-in-the-middle base station snooping. *The Register*, 12, 2015.

[47] S. Rosen, H. Luo, Q. A. Chen, Z. M. Mao, J. Hui, A. Drake, and K. Lau. Discovering fine-grained RRC state dynamics and performance impacts in cellular networks. In *International Conference on Mobile Computing and Networking*, 2014.

[48] D. Rupprecht, K. Jansen, and C. Pöpper. Putting LTE security functions to the test: A framework to evaluate implementation correctness. In *USENIX Workshop on Offensive Technologies*, 2016.

[49] D. Rupprecht, K. Kohls, T. Holz, and C. Pöpper. Breaking LTE on layer two. In *IEEE Symposium on Security and Privacy*, 2019.

[50] D. Rupprecht, K. Kohls, T. Holz, and C. Pöpper. Call me maybe: Eavesdropping encrypted LTE calls with ReVoLTE. In *USENIX Security Symposium*, 2020.

[51] D. Rupprecht, K. Kohls, T. Holz, and C. Pöpper. Imp4gt: impersonation attacks in 4G networks. In *Network and Distributed System Security Symposium*, 2020.

[52] A. Shaik, R. Borgaonkar, N. Asokan, V. Niemi, and J.-P. Seifert. Practical attacks against privacy and availability in 4G/LTE mobile communication systems. In *Network and Distributed System Security Symposium*, 2016.

[53] A. Shaik, R. Borgaonkar, S. Park, and J.-P. Seifert. New vulnerabilities in 4G and 5G cellular access network protocols: exposing device capabilities. In *Conference on Security and Privacy in Wireless and Mobile Networks*, 2019.

[54] TESCOM TC-5910D. `http://en.tescom.co.kr/product/product2?code=020101`.

[55] Representative statements used for the guideline generation. `https://github.com/SysSec-KAIST/DoLTEst/blob/main/table7.md`.

[56] G.-H. Tu, Y. Li, C. Peng, C.-Y. Li, H. Wang, and S. Lu. Control-plane protocol interactions in cellular networks. *ACM SIGCOMM Computer Communication Review*, 2014.

[57] R.-P. Weinmann. Baseband attacks: Remote exploitation of memory corruptions in cellular protocol stacks. In *USENIX Workshop on Offensive Technologies*, 2012.

[58] Accuver XCAL. `http://www.accuver.com/`.

[59] H. Yang, S. Bae, M. Son, H. Kim, S. M. Kim, and Y. Kim. Hiding in plain signal: Physical signal overshadowing attack on LTE. In *USENIX Security Symposium*, 2019.

Appendix

A LTE Attack Models

We discuss representative LTE attack models in detail.

FBS mimics a commercial cell tower to lure nearby cellular devices by operating at a high radio signal power. After a radio connection is established between the malicious base station and a victim device, it transmits an adversarial message to the UE. Because the FBS attacker does not have valid information to accomplish the AKA procedure with the UE, she can transmit the control plane messages only before the UE establishes the security context. Most previous attacks [29, 36, 38, 41, 48, 52, 53] have adopted this threat model, which includes IMSI catching, UE capability hijacking, and fake emergency alerts.

MitM Although an MitM attacker operates a malicious base station as an FBS attacker, she establishes additional radio connections with a legitimate eNB after a victim connects. For a legitimate eNB, the attacker impersonates the victim UE by relaying the victim's messages. Likewise, the attacker impersonates the legitimate eNB to the victim UE. MitM attacker can transmit adversarial messages to the victim UE even after the UE establishes the security context. This is because the MitM attacker accomplishes the AKA procedure by relaying authentication messages of the eNB and the UE. Although the attacker can transmit messages anytime, she can only transmit messages with invalid security protection due to the absence of security keys. Several works [48, 49] have utilized this threat model.

Signal Injection is an intuitive threat model introduced by SigOver attack [59]. Rather than establishing a radio connection with the victim, the signal injection attacker precisely overwrites the signal containing an adversarial message in the physical domain. Unlike the above threat models, the attacker neither requires the victim UE to connect to her FBS nor to relay messages between UE and an eNB. Similar to the MitM attack, the signal injection attacker can inject adversarial control plane messages to the victim UE at any time. The attacker can pass the authentication procedure by allowing the victim UE to communicate with legitimate eNB.

B Handling Silent Test Message Acceptance

In this section, we describe how we identify the test messages being accepted or rejected by the UE via ADB/DM if a UE has no corresponding response (*i.e.*, `Attach reject`, `EMM information`, and `RRCConnectionRelease`).

EMM information. We use ADB to determine whether the test case of `EMM information` is accepted by the UE. This is because `EMM information` is used for changing NITZ information, which ADB can monitor by using `date` ADB command.

Attach reject. We use DM to determine whether the test cases of `Attach reject` are accepted by the UE. The UE's corresponding action to the legitimate `Attach reject` is to changes its EMM state to DEREGISTERED which the NAS specification defines. We monitor such state changes by using DM tools, which show the UE's current EMM state. For example, when the state of UE is changed, the chipset produces DM logs notifying the current UE state information.

RRC connection release. We use DM to identify the UE's acceptance to the test message of `RRCConnectionRelease`. Once the UE accepts the `RRCConnectionRelease`, it destroys radio connection to eNB including all radio bearers and signaling radio bearers. Its RRC state is then changed to RRC IDLE, which we can monitor by DM.

Table 5: Generated guidelines and the corresponding number of test cases. Note that if the guideline involves two IEs (*i.e.*, handover procedure by using RRCConnectionReconfiguration), DOLTEST adds both IEs in generating test cases. We omitted non-trivial IE values for brevity. The * represents the wildcard explained in §5.2.2. We used RRC specification version 15.10.0 NAS specification version 16.5.1.

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

Table 6: A full list of tested devices including device name, phone vendor, baseband vendor, chipset model, firmware version, last updated date, and vulnerabilities of each. We used the latest firmware of a device when we started testing it and have not patched it since then to reproduce our experiments. Please refer to Tab. 3 and §6.1 for a detailed description of the implementation flaws.

#	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