LTESniffer: An Open-source LTE Downlink/Uplink Eavesdropper

Tuan Dinh Hoang
tuan.hoangdinh@kaist.ac.kr
KAIST

CheolJun Park
fermioncj@kaist.ac.kr
KAIST

Mincheol Son
mcson@kaist.ac.kr
KAIST

Taekkyung Oh
ohtk@kaist.ac.kr
KAIST

Sangwook Bae
hoops@kaist.ac.kr
KAIST

Junho Ahn
dwg226@kaist.ac.kr
KAIST

Beomseok Oh
beomseoko@kaist.ac.kr
KAIST

Yongdae Kim
yongdaek@kaist.ac.kr
KAIST

ABSTRACT

LTE sniffers are important for security and performance analysis because they can passively capture the wireless traffic of users in LTE network. However, existing open-source LTE sniffers have only limited functionality and cannot decode data traffic. This paper introduces LTESniffer, the first open-source LTE sniffer that can passively decode both uplink and downlink data traffic. Implementing a sniffer is not trivial because one needs to understand detailed configurations and parameters to successfully decode each user's traffic. Using multiple techniques, we found mechanisms to understand these, which improves our decoding performance. We evaluated the performance of LTESniffer on both testbed and commercial network environments. We also compare the performance of LTESniffer with AirScope, a popular commercial LTE sniffer. Additionally, LTESniffer provides a proof-of-concept API with three functions that can be used for security applications, including identity mapping, identity collecting, and device capability profiling. We release LTESniffer as open-source for future research.

CCS CONCEPTS

• **Security and privacy → Mobile and wireless security**; • **Networks** → *Network monitoring*.

KEYWORDS

LTE; Wireless; Network Analysis; Sniffer; Security

ACM Reference Format:
Tuan Dinh Hoang, CheolJun Park, Mincheol Son, Taekkyung Oh, Sangwook Bae, Junho Ahn, Beomseok Oh, and Yongdae Kim. 2023. LTESniffer: An Open-source LTE Downlink/Uplink Eavesdropper. In *Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '23), May 29-June 1, 2023, Guildford, United Kingdom.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3558482.3590196

1 INTRODUCTION

LTE sniffer is a tool that can passively capture the wireless traffic of users in a cell. Due to the nature of LTE traffic being transferred over the air interface, anyone with the appropriate hardware can sniff LTE signals. By mimicking the behavior of both the user equipment (UE) and the base station, an LTE sniffer can decode downlink and uplink traffic, respectively. In contrast to normal UEs who only decode their traffic, a sniffer decodes all the traffic on the air including plaintext and encrypted packets.

The usefulness of the LTE sniffer has been demonstrated in previous works in both security research and network analysis. In terms of security, it was used for attackers to track victim identifiers [6, 11, 16, 18], detect the presence [6, 7, 18] or physical location [11, 13] of victims, fingerprint the video [1] or website [10, 16] that victims are watching and record phone calls [17]. Meanwhile, several works [2, 5, 12] attempted to monitor the network's capabilities, such as the number of users or throughputs, which are useful for debugging and optimizing networks, using LTE sniffers.

Several LTE sniffers have been implemented and open-sourced, but they primarily focused on decoding the control channel of downlink (*i.e.* PDCCH), which only contains the metadata for data channels. Thus, they cannot obtain IP packets as well as cellular protocol packets, which are used to manage the connection between the device and the cellular network (*e.g.* RRC and NAS). The first attempt to decode the downlink channel was LTEye [12], which aimed at security applications and basic network monitoring functions. OWL [2] and FALCON [5] were later proposed to focus on network monitoring by improving decoding performance and reliability. However, their functionalities remain insufficient in fully addressing the applications proposed by prior research, which usually need packets.

There are three commercial sniffers, Wavejudge [9], ThinkRF [22], and Airscope [20] which can also decode data channels. The first two sniffers can decode both uplink and downlink, but, they are hard to use for research due to their price (*e.g.* starting from $25,000 for Wavejudge). The most popular commercial LTE sniffer in the academia is the Airscope, which can decode the control and data channel on the downlink. However, it still cannot decode uplink, and our experiment shows that its decoding performance for the data channels is low. In addition, commercial sniffers are closed-source so it is not easy to modify or add a new feature.

In this paper, we introduce LTESniffer, the first open-source LTE sniffer that can passively decode the data traffic of uplink and downlink. To decode data traffic without knowing the obscure and encrypted UE-specific configurations, LTESniffer adopts several techniques to infer that information for each UE. Obtaining this information is crucial for the sniffer to improve the accuracy of decoding data traffic, particularly for high throughput scenarios. Also, LTESniffer compensates for the unknown propagation delays of uplink signals, because synchronizing the uplink and downlink signal is important for successful decoding. We implemented LTESniffer

WiSec '23, May 29-June 1, 2023, Guildford, United Kingdom

ACM ISBN 978-1-4503-9859-6/23/05…$15.00
https://doi.org/10.1145/3558482.3590196

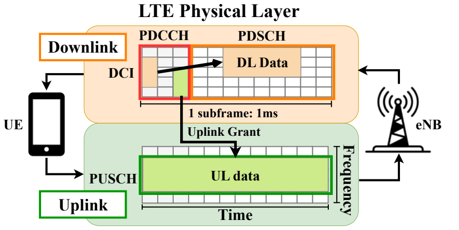

Figure 1: Physical layer channel structure.

by adding a significant amount of functions on FALCON, with the help of libraries of the open-source LTE stack [21]. LTESniffer is also implemented to support multi-threading to optimize the sniffing performance.

We evaluate the performance of LTESniffer on two environments, a testbed, and a commercial network. Our evaluation on the downlink shows that LTESniffer decodes LTE packets better than AirScope. Note that we used the second latest version of Airscope (release 19.09) for comparing the performance. We could not buy the latest version (release 21.11) due to the budget issue, so we don't know its functionality or performance. Moreover, thanks to the capability of LTESniffer that can decode the data traffic, LTESniffer can also support security applications proposed by previous works, which require cellular protocol packets. As a proof of concept, we provide an API with three functions, identity mapping [8, 16], identity collecting [7], and device capability profiling [15, 19].

Target. The target of LTESniffer is to capture the wireless packets between the base station and the user. It can only obtain encrypted packets in most cases because it can't know the cryptographic keys of users. However, some packets are transferred in plaintext by design. Moreover, encrypted packets are required in certain security research.

In summary, our contributions are three-fold:

- We present LTESniffer, the first open-source LTE sniffer that can passively decode data traffic of both the downlink and uplink, which can be used for network security and performance analysis. We release LTESniffer as open-source for future research at https://github.com/SysSec-KAIST/LTESniffer.
- We evaluate the performance of LTESniffer in the testbed and commercial environment for old and new smartphones using various internet services. We show that our sniffer decodes LTE packets better than Airscope, the most popular commercial LTE sniffer in academia.
- We also support API for security applications proposed by previous works. As proof-of-concept, we provide three functions; identity mapping, identity collecting, and capability profiling.

2 BACKGROUND

2.1 Cellular network architecture

LTE architecture. LTE network consists of three main components: User Equipment (UE), Evolved Node B (eNB), and Evolved Packet Core (EPC). UE indicates cellular devices used by an end-user to access the mobile network. eNB is a base station that provides UE with a wireless connection to the EPC or internet by managing the radio resources. EPC represents the core network responsible for authenticating users and managing user mobility.

Radio Network Temporary Identifier (RNTI). RNTI is an identifier of UE in the radio channel, which is assigned when UE connects to the eNB. It plays a critical role in wireless communication in LTE because it enables the network to manage and control the radio connection between the mobile device and the base station. In particular, it enables UE to distinguish signals for itself from others. Note that RNTI is re-assigned whenever UE stops using services.

2.2 Physical layer channel structure

LTE frame. In the wireless channel of LTE, the uplink transmission (*i.e.* UE to eNB) is carried out using SC-FDMA technology, whereas the downlink transmission (*i.e.* eNB to UE) is performed using OFDMA technology. LTE frame is a time interval used for transmitting and receiving data carried through uplink or downlink signals. One frame consists of ten subframes, each lasting for one millisecond, where the subframe is the basic unit in the physical layer for transmitting the packet. Each subframe is comprised of fourteen OFDM symbols. Fig. 1 shows the subframe for downlink and uplink. The subframe also includes a reference signal at a fixed position for signal synchronization.

Physical downlink channels. The eNB transfers data traffic to UE through two downlink channels, Physical Downlink Control Channel (PDCCH) and Physical Downlink Shared Channel (PDSCH), as shown in Fig. 1. PDCCH carries the metadata for PDSCH called Downlink Control Information (DCI), which contains information about resource allocation, modulation and coding scheme, and retransmission. Each UE utilizes RNTI to identify which DCI is being transmitted to it. PDSCH carries IP packets and cellular protocol packets according to the information indicated in DCI messages.

Physical uplink channels. In the uplink, UE transfers data traffic to eNB through Physical Uplink Shared Channel (PUSCH) as shown in the lower box of the Fig. 1. Note that the DCI in PDCCH also defines how UE should encode uplink traffic in PUSCH. Although there is another channel called Physical Uplink Control Channel (PUCCH), it is not used for transferring data traffic; it carries control information such as acknowledgment or channel quality indication.

Scheduling physical layer communication. To schedule radio communication with each UE, eNB broadcasts PDCCH containing DCI messages. Once the DCI messages are transmitted, each UE looks for the DCI assigned to itself by verifying the PDCCH payload. In detail, UE checks CRC (Cycle Redundancy Check) bits consecutive of each payload using its RNTI. Once the CRC check is passed, UE confirms that the payload is the DCI assigned to itself. After finding its DCI, UE can determine 1) the resource blocks in PDSCH allocated for its downlink data and 2) the resource blocks in PUSCH allocated for the transmission of its uplink data by decoding the DCI. Based on the decoded DCIs, UE can properly decode its downlink data and transmit uplink data. Fig. 1 shows the overview of the communication schedule.

2.3 Physical layer modulation scheme

UE and eNB utilize various modulation schemes in data channels (*i.e.* PDSCH and PDCCH) to maximize the reliability and efficiency of data transmission based on the channel quality. To be specific, the modulation and coding scheme (MCS) is determined by the MCS table and MCS index that are exchanged between UE and eNB. For downlink, there are two different MCS tables, one that supports up to 64QAM, and another that supports up to 256QAM. UE determines

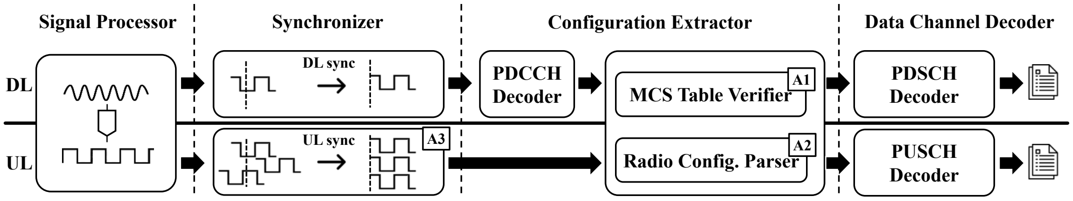

Figure 2: Architecture of LTESniffer.

the proper MCS table by decoding `RRCConnectionReconfiguration` message from eNB, which is encrypted.

On the other hand, a single MCS table is used for the uplink; nevertheless, its interpretation varies depending on the UE's capability to support 64QAM for uplink transmission. Note that this capability information is usually transmitted only when UE first connects to the network, through the `UECapabilityInformation` message.

3 LTESniffer

We introduce the first open-source LTE sniffer capable of monitoring both uplink and downlink traffic. Our ultimate goal is to develop a new sniffer that can resolve the limitations of current LTE sniffers, from that enabling its application in the security and analysis research in the LTE network. For this purpose, we analyze the crucial considerations involved in decoding traffic from all UEs. We propose three techniques to address these considerations. Lastly, we describe the design and implementation details of LTESniffer.

3.1 Considerations and our approach

In cellular networks, eNBs use different radio configurations (*e.g.* modulation schemes and physical layer parameters) for each user to efficiently allocate a radio resource and ensure reliable quality of services. Obtaining configurations using a passive sniffer is challenging as parameters are assigned differently for each user by the networks and most messages are encrypted. To successfully decode both uplink and downlink traffic, our sniffer needs to address the following three considerations.

[C1] Obscure modulation schemes. A sniffer needs to identify which modulation scheme each user uses to decode the traffic successfully. In the downlink, even though a sniffer can retrieve an MCS index from the DCI message, it cannot determine which modulation scheme the index represents, because a sniffer does not know about the MCS table. Also, the situation is similar in uplink because the message that carries the configuration of the uplink MCS table is mostly encrypted. Thus, it is challenging for a passive sniffer to acquire the MCS table configurations for each user.

[A1] Inferencing MCS table based on the decoding result. We leverage the decoding results of the data channel (*i.e.* PDSCH and PUSCH) to infer the correct MCS table configurations for each UE. First, we decode the data channel using all possible MCS tables to calculate the potential packet sizes and modulation schemes. Then, we validate the MCS table configuration by using the decoding results (success or failure) on the data channel. This is because the correct configuration can only successfully decode the data channel. Note that LTESniffer needs to infer the MCS table only once for each user because it is fixed within the same radio connection. Thus, we save the appropriate MCS table configuration for each UE to use when decoding its subsequent transmissions, thereby reducing the computational cost.

[C2] Diverse radio configurations. Obtaining three configurations related to radio connection, downlink power allocation, uplink control information, and uplink channel state information is crucial for a sniffer to decode data traffic using a high MCS index. While establishing a radio connection, eNB delivers essential radio configurations that are required for processing physical signals, including these. In addition, these configurations are not fixed, as they depend on the channel quality of each UE and may be changed by the eNB for each radio connection.

[A2] Adopting UE-specific configurations. A sniffer can obtain radio connection configurations if it decodes the cellular protocol messages. Because these messages are transmitted at the beginning of the connection, they are not encrypted. To this end, we record the configurations for each user and use them for decoding data traffic using a high MCS index.

[C3] Misaligned uplink signals. In the uplink direction, each UE transmits its own signal to the eNB, and since there can be multiple UEs, there will be as many uplink signals as the number of UEs. On the sniffer side, the signals transmitted by different UEs will have different propagation delays due to the varying distances between each UE and the sniffer. Thus, a sniffer may receive misaligned signals, which makes it difficult to correctly decode the signals.

[A3] Compensating time delay. We implement an algorithm that first captures uplink-downlink subframes at the same time and then leverages functions from srsRAN [21] to calculate the time delay for each uplink signal and align them. Note that contrary to UE-specific values in **A2** that are transferred by eNB, this delay should be calculated by the sniffer.

3.2 Design of LTESniffer

We designed LTESniffer as illustrated in Fig. 2. LTESniffer consists of four components: 1) Signal processor, 2) Synchronizer, 3) Configuration extractor, and 4) Data channel decoder. The aforementioned approaches were directly applied to the synchronizer and configuration extractor blocks.

Signal processor. LTESniffer captures analog signals and converts them into digital samples. Definitely, mobile communication equipment (*i.e.*, UE and eNB) initiates this process as the first step. We adopt the signal processing modules of srsUE and srseNB [21] to process both downlink and uplink signals: downlink processing at srsUE, and uplink processing at srseNB.

Synchronizer. The synchronizer performs two functions: downlink synchronization and uplink synchronization. First, LTESniffer synchronizes the downlink channel by looking for the broadcast signals (*i.e.*, primary synchronization signal and secondary synchronization signal) to decode the downlink traffic correctly. In addition, we designed the uplink synchronizer block to align the time delay of the UEs' uplink signal. As mentioned in §3.1, LTESniffer receives multiple uplink signals with various propagation delays. In

Table 1: Capability comparison of LTE sniffers.

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

the uplink synchronizer block, LTESniffer first captures the uplink-downlink subframes at the same time. After that, LTESniffer calculates the time delay of each UE's uplink signal to align by using the received downlink signal. Note that these two synchronizer sub-blocks should be preceded to correctly decode both signals.

Configuration extractor. Prior to decoding the data channel, LTESniffer needs to learn the UE-specific configurations (*i.e.*, modulation scheme and radio configurations). For this purpose, this block determines the MCS table used by each UE for uplink and downlink signals (**A1**), and also extracts radio configurations during the initial radio connection between a UE and an eNB (**A2**). There are three sub-blocks to perform the extractor process: PDCCH decoder, MCS table verifier, and radio connection parser. The PDCCH decoder has the role of decoding DCI messages and identifying RNTIs. After PDCCH decoding, LTESniffer obtains the data-related information such as the location of user data in PDSCH, the MCS index for decoding data channel, and the packet size of downlink data. This information is used by sub-blocks (*i.e.*, MCS table verifier and radio configuration parser) for helping data channel decoder. If LTESniffer doesn't have the correct MCS table of a UE, it conducts the MCS table verifier and stores the configuration of that UE's MCS table. In addition, by listening to the radio connection setup messages, the radio connection parser extracts the essential configurations used to process both data channels.

Data channel decoder. Finally, the data channel decoder processes both signals in each direction based on the configurations obtained from the configuration extractor block. We designed it to support multi-threading to achieve real-time decoding of all LTE signals, which are uplink and downlink signals of the entire active users.

3.3 Implementation

LTESniffer is implemented by adding numerous new features on top of FALCON. In detail, we newly implemented functions for decoding data channels for uplink and downlink by using the srsRAN [21] library, because FALCON has functions only for decoding PDCCH. In addition, we implemented functions for handling UE-specific configurations and delays explained in §3.2. Also, we re-based the functions of FALCON to a new version of srsRAN [21] and added several new functions to support decoding data packets that follow LTE-A (from 3GPP release 10). Overall, LTESniffer is implemented to use multithreading to optimize the performance. LTESniffer is operated on an Ubuntu desktop using an Intel Core i7 processor and a software-defined-radio (SDR), USRP X310, which has two antenna ports as an RF-frontend. In Tab. 1, the overall capability of LTESniffer is compared with that of other sniffers.

4 EVALUATION

In this section, we evaluate the performance of LTESniffer in two different environments, a testbed, and a commercial network. We first describe our experimental setup in §4.1. Then, we present the performance when decoding downlink (§4.2) and uplink (§4.3).

4.1 Experimental setup

To show the performance of LTESniffer, we use two experimental setups; a testbed and commercial LTE networks. We demonstrate the feasibility of our sniffer in the testbed environment and examine its practicality in a crowded commercial environment.

We use the **success rate** to evaluate the capability of a sniffer in decoding downlink and uplink traffic, which is defined as follows.

$$\textit{Success rate} = \frac{\textit{\# of successfully decoded PUSCH or PDSCH msgs}}{\textit{Total \# of detected DCIs for UL or DL}}$$

If a sniffer has a high success rate, it means that it is successfully capturing most of LTE traffic, which is the main focus of the sniffer. As a numerator, we use the number of successfully decoded PUSCH or PDSCH messages from detected DCI messages. Note that one PUSCH or PDSCH message is mapped to one DCI message. As a denominator, we use the number of detected DCI messages to represent the total number of transmitted packets. It is worth noting that this is the best estimation, because one can not know if there is a missing DCI message unless one has full access to the base station. To ensure its reliability, we further compare the performance of LTESniffer and AirScope in terms of DCI detection in §4.2.

There are several factors that affect the success rate: 1) the number of active users, 2) the throughput of the network, 3) the signal quality of the radio channel, and 4) the synchronization of the signal. 1) and 2) may affect it, because the sniffer might not be able to process all of the messages when receiving a large number of messages. For 3), bad signal quality might damage the packets during signal propagation. At last, misaligned uplink signal also degrades the performance as discussed in §3.1.

Testbed environment. Our experimental setup is based on an open-source LTE platform, srsRAN [21], for evaluating the performance of LTESniffer in a controlled environment. It comprises test eNB and test EPC. We utilized three identical Linux machines, each of which was equipped with a CPU Intel i7-11700K and a USRP X310 to run srsRAN, AirScope, and LTESniffer. We used two COTS UEs, Samsung Galaxy Note 20 Ultra and Note 5 that supports up to downlink 256QAM / uplink 64QAM (LTE-A) and downlink 64QAM / uplink 16QAM modulation scheme, respectively. To generate downlink and uplink traffic, we make UEs use three LTE services; web surfing, watching Youtube videos, and data download/upload.

Commercial environment. We operate LTESniffer in the commercial network to validate our system and demonstrate its practicality. Note that there are more users and data traffic than in the testbed environment. Also, channel conditions and data throughput are unpredictable. We used the same UEs and services as the testbed environment. For downlink experiments, we connected the test UEs to a base station (20 MHz bandwidth) that served around 150 active users. For uplink experiments, we connected the test UEs to a base station (10 MHz) located near a busy road with many pedestrians. In addition, when comparing the performance of LTESniffer with Airscope, we always used an identical hardware setup (same CPU Intel i7-11700K and USRP X310 at the same location).

Ethical considerations. In the testbed environment, we operated our own base station in a Faraday cage under the basement with unused frequency to avoid interfering with other users. Also, we carefully checked if any other user connects to our testbed. In the commercial environment, we used XCAL [23] to identify RNTIs

Table 2: Downlink sniffing success rate (%) of LTESniffer and AirScope in testbed and commercial environment.

Environment	LTESniffer / AirScope					
	Galaxy Note 5			Galaxy Note 20 Ultra		
	Web	Video	Data	Web	Video	Data
Testbed	91 / 50	75 / 36	71 / 19	83 / 1	72 / 1	68 / 1
Commercial	83 / 53	75 / 40	52 / 23	73 / 1	28 / 1	19 / 1

Table 3: Downlink sniffing success rate (%) of LTESniffer and AirScope in different SNR in commercial environment.

SNR (dB)	LTESniffer / AirScope					
	Galaxy Note 5			Galaxy Note 20 Ultra		
	Web	Video	Data	Web	Video	Data
30	92 / 92	83 / 74	61 / 40	93 / 1	70 / 1	60 / 1
25	83 / 53	75 / 40	52 / 23	73 / 1	28 / 1	19 / 1
22	52 / 46	31 / 19	12 / 9	44 / 1	19 / 1	5 / 1

assigned to our devices. Thereafter, we modified LTESniffer to only save packets for our devices and to discard other traffic. To evaluate the decoding performance in general, we only checked the CRC of the packets for other users. This ensures that our sniffer does not collect any privacy-related information from other users.

4.2 Results in downlink direction

We evaluate the success rate for the downlink to analyze the effect of service that UE uses, signal quality, and number of active UEs.

First, we evaluate the success rate of LTESniffer and compare it with AirScope in two environments: testbed and commercial environments. As shown in Tab. 2, LTESniffer achieves a much higher success rate than AirScope regardless of the service that the UE uses in both environments. When the UE uses high throughput services such as data downloading, the success rate of LTESniffer drops to 52% in the commercial experiment. Despite the significant effect of the service on the success rate, LTESniffer's success rate is more than twice that of AirScope, which is 23%. Additionally, the results show that AirScope cannot decode the downlink traffic to the Galaxy Note 20 Ultra. The reason for this low success rate is that AirScope release 19.09 does not support LTE-A. In contrast, LTESniffer supports LTE-A and utilizes the MCS table verifier to decide the correct modulation scheme to decode the downlink traffic of both Galaxy Note 5 and Galaxy Note 20 Ultra.

We also conducted an additional experiment where we operated LTESniffer in different signal qualities in the commercial network. As a result, the higher the signal quality, the higher the success rate of the sniffer, as shown in Tab. 3. We also observed that the SNR significantly affects the success rate, especially for high throughput services such as watching videos or data downloading. Note that in overall, LTESniffer demonstrated a success rate of about 90% for decoding downlink traffic of all active UEs in the commercial eNB, with an average of 95 UEs simultaneously receiving downlink signals. This result implies that LTESniffer is capable of effective decoding even in scenarios where multiple UEs are present. Furthermore, we confirmed that our multi-threading function in the data channel decoder considerably improves the decoding rate compared to AirScope when the network is congested. Fig. 3 shows the performance of two sniffers that run concurrently on two PCs (Intel i7-4770) during the peak hour on a cell with around 200 active UEs.

Table 4: Uplink sniffing success rate (%) of LTESniffer

Smartphone	Testbed			Commercial		
	Web	Video	Data	Web	Video	Data
Galaxy Note 20 Ultra	97	98	85	87	95	90
Galaxy Note 5	99	96	98	95	96	70

Performance of DCI detection. We run LTESniffer and AirScope for 10 minutes to see the reliability of our results on DCI detection, as discussed in section §4.1. Both sniffers had similar number of detected DCIs, with a correlation of 0.994. On average, LTESniffer and AirScope detected 1635 and 1671 DCIs every second, respectively. This result implies that the DCI detection result from LTESniffer is reliable for calculating the success rate.

4.3 Results in uplink direction

The results of the uplink sniffing experiment in the testbed and commercial environments are shown in Tab. 4. Since there are no other uplink sniffers available, we only evaluated LTESniffer for uplink traffic. In both environments, LTESniffer effectively decoded uplink messages. This result indicates that our approach, which involves extracting configurations of the uplink signal and compensating for time delay, enables the sniffer to effectively decode the uplink signal. However, in the uplink direction, we cannot decode all UEs' uplink traffic, unlike downlink traffic, since the signal power transmitted from the smartphones is quite weak to decode with a low-cost antenna. We discuss this limitation in detail in §6.

5 SECURITY APPLICATIONS

LTESniffer also supports an API with three functions for security applications and research. Many LTE security research assumes a passive attacker (*i.e.* sniffer) that can capture privacy-related packets on the air. However, non of the current open-source sniffers satisfy their requirements as they cannot decode protocol packets in PDSCH and PUSCH. Thanks to the capability of LTESniffer, we developed a proof-of-concept security API that supports three tasks that were proposed by previous works: 1) *Identity mapping* [8, 16], 2) *IMSI collecting* [7], and 3) *Capability profiling* [15, 19].

Identity mapping. This function enables us to trace and map two identifiers of arbitrary users in eNB, Temporary Mobile Subscriber Identity (TMSI) and RNTI. TMSI is an identity of UE used by EPC for identification, which is assigned during the initial network connection. Previous works have shown that mapping TMSI and RNTI can be used for several security applications, such as device tracking [8], location tracking [11], or a starting point for the website fingerprinting attack [16]. LTESniffer can acquire these identities by decoding uplink `RRCConnectionRequest` or downlink `RRCConnectionSetup` messages, which are not encrypted. Consequently, LTESniffer provides TMSI-RNTI pairs of users connected to eNB.

IMSI collecting. This function collects a permanent identifier of UE called International Mobile Subscriber Identity (IMSI) from the LTE traffic. Leaking IMSI causes many privacy issues such as surveillance [3, 7, 18], because it is a permanent identity bonded to the USIM. IMSI is leaked in plaintext through two procedures; 1) paging and 2) the initial network connection process. LTESniffer collects the IMSI exposed in the air by monitoring uplink `Attach Request`, `Identity Response`, and downlink `Paging` messages. Note that although this function can capture IMSI only when it is transmitted over the air, it has an advantage in terms of stealthiness.

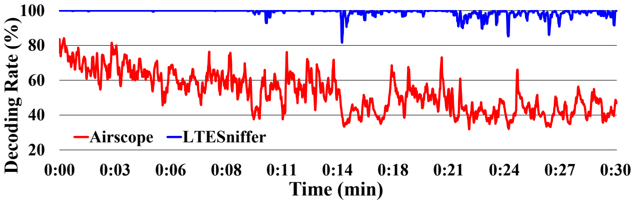

Figure 3: Subframe decoding rate at peak time

Capability profiling. This function enables users to capture the capability of UE, which can further be used to fingerprint the type or even the model of the device [11, 15, 19]. When UE connects to the network, UE delivers its capability information to the network for connection optimization. To this end, LTESniffer captures the uplink `Attach Request` and `UECapabilityInformation` messages of each user for further analysis.

6 DISCUSSION

Distance for uplink sniffing. The effective range for sniffing uplink is limited in LTESniffer due to the capability of the RF front-end of our hardware (*i.e.* SDR). The uplink signal power from UE is significantly weaker compared to the downlink signal because UE is a handheld device that optimizes battery usage, while the eNB uses sufficient power to cover a large area. To successfully capture the uplink traffic, LTESniffer can increase the strength of the signal power by i) being physically close to the UE, or ii) improving the signal reception capability with specialized hardware, such as a directional antenna, dedicated RF front-end, and signal amplifier.

Supporting carrier aggregation. Currently, the LTESniffer does not support LTE carrier aggregation (CA), a technique that increases the data rate by sending traffic over multiple frequencies simultaneously. This stems from the fact that LTESniffer is implemented to synchronize with one frequency at a time for each uplink and downlink. To overcome this challenge, one can operate multiple LTESniffer concurrently, each tuned to a different frequency. After then, one needs to distinguish the victims' traffic decoded from each band and combine the data streams in order to obtain complete traffic. Alternatively, one can implement a function to support CA and use multi-channel SDR. We leave this as future work.

Performance comparison. Recently, a sniffer called LTEProbe that can decode both uplink and downlink of LTE data channels was proposed by Kotuliak et al. [11]. However, the performance and implementation details are unknown because the authors have focused the application of LTEProbe on the localization attack. We could not check its functionality because it is not open-sourced. Also, while we used the second latest version of AirScope in §4, the performance could be improved on the latest version. According to their web page, the latest release added experimental support for decoding 256-QAM, which can enhance the success rate of the sniffer. Nevertheless, AirScope still can not decode the uplink.

Supporting 5G sniffer. To decode 5G signals, a sniffer needs new software (5G protocol stack) and better hardware. Recently, an open-source software radio suite, srsRAN [21]) supports the physical layer of 5G. Thus, we believe that one can implement software for 5G sniffers by using it. However, fully supporting 5G is challenging in terms of the hardware. The 5G sniffer should be able to process RF signals above 24GHz because 5G utilizes two frequency ranges, sub-6GHz and millimeter wave. Note that one of the popular high-end SDR [4], USRP X410, can support only up to 8 GHz. Recently, Ludant et al. [14] implemented a 5G sniffer from the scratch. However, it can only decode the PDCCH, only supports sub-6GHz frequencies, and is not open-sourced yet.

7 CONCLUSION

We present LTESniffer, the first open-source LTE sniffer for decoding data traffic of both uplink and downlink. To this end, we adopted techniques to address several considerations that stem from the limited information about the user's physical layer configuration on the sniffer. Our evaluation shows that LTESniffer decodes LTE packets better than AirScope, the most popular sniffer in academia. We release LTESniffer for future research and also provide an API for security applications proposed by previous LTE security works.

8 ACKNOWLEDGEMENTS

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.2022-0-01202, regional strategic industry convergence security core talent training business).

REFERENCES

[1] S. Bae, M. Son, D. Kim, C. Park, J. Lee, S. Son, and Y. Kim. Watching the Watchers: Practical Video Identification Attack in LTE Networks. In *31st USENIX Security Symposium*, 2022.
[2] N. Bui and J. Widmer. OWL: A reliable online watcher for LTE control channel measurements. In *USENIX ATC*, 2016.
[3] M. Chlosta, D. Rupprecht, C. Pöpper, and T. Holz. 5G SUCI-catchers: still catching them all? In *WiSec*, 2021.
[4] Ettus Research, USRP X410. https://www.ettus.com/all-products/usrp-x410/.
[5] R. Falkenberg and C. Wietfeld. FALCON: An accurate real-time monitor for client-based mobile network data analytics. In *IEEE GLOBECOM*, 2019.
[6] B. Hong, S. Bae, and Y. Kim. GUTI Reallocation Demystified: Cellular Location Tracking with Changing Temporary Identifier. In *NDSS Symposium*, 2018.
[7] 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 *NDSS Symposium*, 2019.
[8] R. P. Jover. LTE security, protocol exploits and location tracking experimentation with low-cost software radio. *arXiv preprint*, 2016.
[9] Keysight, WaveJudge Wireless Analyzer Solutions. https://www.keysight.com/us/en/products/wireless-analyzers/wavejudge-wireless-analyzer-solutions.html.
[10] K. Kohls, D. Rupprecht, T. Holz, and C. Pöpper. Lost traffic encryption: fingerprinting LTE/4G traffic on layer two. In *WiSec*, 2019.
[11] M. Kotuliak, S. Erni, P. Leu, M. Röschlin, and S. Čapkun. LTrack: Stealthy tracking of mobile phones in LTE. In *31st USENIX Security Symposium*, 2022.
[12] S. Kumar, E. Hamed, D. Katabi, and L. Erran Li. LTE radio analytics made easy and accessible. In *ACM SIGCOMM*, 2014.
[13] N. Lakshmanan, N. Budhdev, M. S. Kang, M. C. Chan, and J. Han. A Stealthy Location Identification Attack Exploiting Carrier Aggregation in Cellular Networks. In *30th USENIX Security Symposium*, 2021.
[14] N. Ludant, P. Robyns, and G. Noubir. From 5G Sniffing to Harvesting Leakages of Privacy-Preserving Messengers. In *IEEE S&P*, 2023.
[15] B. Oh, J. Ahn, S. Bae, M. Son, Y. Lee, M. Kang, and Y. Kim. Preventing SIM Box Fraud Using Device Model Fingerprinting. In *NDSS Symposium*, 2023.
[16] D. Rupprecht, K. Kohls, T. Holz, and C. Pöpper. Breaking LTE on layer two. In *IEEE S&P*, 2019.
[17] D. Rupprecht, K. Kohls, T. Holz, and C. Pöpper. Call me maybe: Eavesdropping encrypted LTE calls with ReVoLTE. In *29th USENIX Security Symposium*, 2020.
[18] 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 *NDSS Symposium*, 2016.
[19] A. Shaik, R. Borgaonkar, S. Park, and J.-P. Seifert. New vulnerabilities in 4G and 5G cellular access network protocols: exposing device capabilities. In *WiSec*, 2019.
[20] Software Radio Systems. Airscope, 2019. http://www.softwareradiosystems.com.
[21] Software Radio Systems. srsRAN, 2021. https://github.com/srsran/srsRAN.
[22] ThinkRF. ThinkRF, 2023. https://thinkrf.com/.
[23] Accuver XCAL, 2023. http://www.accuver.com/.