From 5G Sniffing to Harvesting Leakages of Privacy-Preserving Messengers

Norbert Ludant
Northeastern University
Boston, USA
ludant.n@northeastern.edu

Pieter Robyns
Hasselt University - tUL - EDM
Belgian Cyber Command
Hasselt, Belgium
pieter.robyns@uhasselt.be

Guevara Noubir
Northeastern University
Boston, USA
g.noubir@northeastern.edu

2023 IEEE Symposium on Security and Privacy (SP) | 978-1-6654-9336-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/SP46215.2023.10179353

***Abstract*—We present the first open-source tool capable of efficiently sniffing 5G control channels, `5GSniffer` and demonstrate its potential to conduct attacks on users privacy. `5GSniffer` builds on our analysis of the 5G RAN control channel exposing side-channel leakage. We note that decoding the 5G control channels is significantly more challenging than in LTE, since part of the information necessary for decoding is provided to the UEs over encrypted channels. We devise a set of techniques to achieve *real-time* control channels sniffing (over three orders of magnitude faster than brute-forcing). This enables, among other things, to retrieve the Radio Network Temporary Identifiers (RNTIs) of all users in a cell, and perform traffic analysis. To illustrate the potential of our sniffer, we analyse two privacy-focused messengers, Signal and Telegram. We identify privacy leaks that can be exploited to generate stealthy traffic to a target user. When combined with `5GSniffer`, it enables *stealthy* exposure of the presence of a target user in a given location (solely based on their phone number), by linking the phone number to the RNTI. It also enables traffic analysis of the target user. We evaluate the attacks and our sniffer, demonstrating nearly 100% accuracy within 30 seconds of attack initiation.**

***Index Terms*—5G, Control Channel Sniffing, Tracking, Traffic Patterns, Linkage Attacks, Privacy-preserving Messengers, Privacy Leaks**

I. Introduction

The 3GPP 5G is hailed as a major step towards more ubiquitous and pervasive communications. It is indeed flexible and extensible to support a variety of unique application requirements, from Ultra-Reliable Low-Latency Communications to enhanced Mobile Broadband, as well as specific industry requirements such as V2X, Smart Grid, and Remote Healthcare [3], [8]. This capability to address unique needs is very promising to adequately support a larger number of applications, including critical ones such as self-driving cars, robotics, and remote surgeries [11], [24], [58].

Cellular systems, however, have a history of privacy issues since their second generation (GSM) that took security seriously. Over the years, researchers were able to demonstrate attacks against every generation of cellular systems from 2G to 5G, by preying on design, implementation, and operation flaws. For instance, that it is possible to clone SIM cards [14], decrypt traffic [48], track users [25], [26], [27], [61], [13], identify devices [51], spoof DNS [47], fingerprint traffic [34], [10], conduct denial of service attacks [33], [50], [40], [38], [23], [16], downgrade to insecure versions [50], reinstall old keys [57], and inject malicious messages by over-shadowing legitimate signaling [60], [19], [35], [41].

One of the promises of the 3GPP 5G efforts is to improve privacy. Towards this goal, 5G introduced the Subscription Concealed Identifier (SUCI), encrypting the Subscription Permanent Identifier (SUPI) to prevent sending it in the clear, therefore defending against attacks on user privacy, such as tracking. 5G also requires that the Temporary Mobile Subscriber Identity (TMSI) be regenerated after each paging and communicated to the User Equipment (UE) over an encrypted channel [9, p. 99]. It is today believed that such measures, if deployed, address the key privacy issues present in previous 3GPP systems generations [17]. However, despite the improvements in privacy, we find that sniffing the 5G control channel is still possible today, which enables attacks on user privacy. Such attacks are possible due to a combination of fundamental design issues and optimizations techniques.

In this paper, we analyze the 3GPP 5G control channels and mechanisms towards the goal of developing a reliable real-time 5G sniffer. Our analysis reveals that decoding some of the 5G control channels, in particular the Physical Downlink Control Channel (PDCCH) is significantly *more challenging* than in LTE (intentionally or unintentionally). For instance, decoding requires descrambling with information (RNTI and `pdcch-DMRS-ScramblingID`) provided to the UE in encrypted messages and therefore not available to the sniffer. Attempting to blindly decode control channels would require brute-forcing over 2^{44} combinations of missing information per control channel message (involving at least one polar decoding for each attempt). In addition to the infeasible computational complexity, this leads to a high false-positive rate as some of these combinations lead to a correct Cyclic Redundancy Check (CRC) while they are not actual valid messages, deeming a naive sniffer approach ineffective and unreliable. We devise a set of techniques and optimizations that enable the development of a reliable real-time 5G control channel sniffer, including exploitation of redundancy in the rate matching of the Downlink Control Information (DCI) encoder, side channels, prioritization of RNTI processing, and parallelism. We implemented and evaluated our sniffer, demonstrating more than 3 orders of magnitude performance

DOI 10.1109/SP46215.2023.00110

improvement over a brute-force approach and the elimination of the false positive DCIs (256.3 on average per DCI) encountered through a brute-force approach.

Our 5G sniffer is able to recover all RNTIs active in a cell and the data traffic load of the given RNTIs. To illustrate the privacy implications of such capabilities, we considered two popular privacy-focused messenger applications, `Signal` and `Telegram` (used by over 500 million people). We found new vulnerabilities and features that when combined with our 5G sniffer allow to link the RNTI to a user (defined by its *phone number*) and therefore reveal its presence within the coverage area of the 5G gNodeB (gNB). Specifically, sending a message with an incorrect Message Authentication Code (MAC) or manipulated padding results in it being dropped without notifying the receiver. We call this a stealthy message. This enables an attack where an adversary aiming at determining the presence of a victim sends stealthy messages to their phone number and analyzes the control traffic. Although linking the RNTI to a specific user was previously mentioned for 4G networks in [30], unlike in 4G, 5G control channel parameters are conveyed in secure RRC messages. Therefore, linking a unique RNTI is infeasible without the new techniques we introduce in this paper (Section III-B). Furthermore, no specific techniques were provided on how it can be achieved, nor implementation/evaluation. Authors in [47] show it is possible to link arbitrary RNTIs to TMSIs, but this does not constitute a user-specific attack, and would need to rely on other mechanisms to do so, such as attacks against paging. However, paging security has been significantly enhanced in 5G (e.g., TMSI is changed on every paging and is never reused). Thus, our work is not only the first to provide techniques for such attacks to work in 5G networks, but it also provides the first evaluation in a real-world scenario.

Our results show that the attack is highly effective, and the success rate scales with exposure time to the attack. The attack reaches a 94% true positive rate (TPR) of detection, and 4% of false positive rate (FPR) for only 15 seconds of exposure. The accuracy increases to 100% TPR and 2% FPR within 30 seconds of exposure time. We note that this kind of attack is of independent interest, as it applies to phones connected over other links than 5G (e.g., LTE, or Wi-Fi), to other applications (WhatsApp uses the same protocol as Signal [59]), and enables other attacks such as denial of service and battery draining. Our contributions are:

- To the best of our knowledge, we propose the first real-time and reliable control channel *sniffer* for 5G NR. Our sniffer is capable of, for example, inferring encrypted RRC information such as the scrambling ID of users from information leaks, thereby reducing the computational complexity for decoding a single message from 2^{44} to 2^{19} in the worst case. In addition, our techniques do not suffer from the high false-positive rate of a naive brute-force approach.
- We analyze the 5G control channels and introduce new techniques to optimize the decoding for real-time operations enabling the exposure of, among other things, all RNTI identifiers, and fine-grained user traffic scheduling.
- We analyze privacy-focused messengers Signal and Telegram and identify privacy leaks that enable stealthy exposure of the presence of a user in a given location (based on their phone number), by linking the phone number to RNTI, and traffic analysis.
- We implement and evaluate the attack in a commercial network (while preserving users privacy), demonstrating nearly 100% accuracy within 30 seconds.
- We discuss several defenses to mitigate these attacks as well as ethical disclosures to the relevant parties.

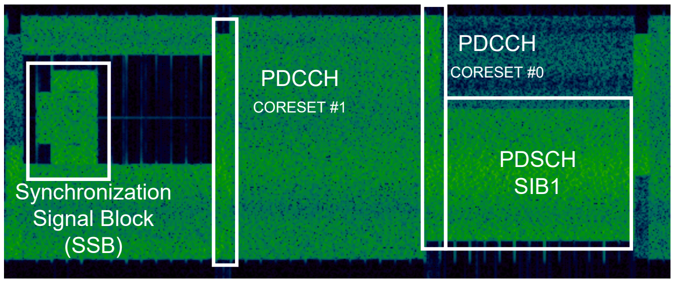


Fig. 1: Recording of a 5G gNB transmission from a live operator's 5G deployment, showing the SSB, two CORESET configurations and SIB1

II. Essential Background Towards a 5G Sniffer

In this section, we describe the essential elements of 5G Radio Access Network (RAN) design, structure and mechanisms. The 5G RAN specifications are complex and span thousands of pages of 3GPP documents. Therefore, we focus on the aspects relevant to understanding the requirements and challenges of developing a promiscuous sniffer for the 5G RAN.

A. *5G New Radio Physical Layer*

5G New Radio (NR) builds on top of the previous generation's physical layer, Long-Term Evolution (LTE). It adds support for new technologies by adding multiple layers of operational flexibility. Such flexibility increases the complexity of a sniffer, in terms of implementation as well as computation, since some of the parameters (e.g., those communicated to the UEs in encrypted messages) need to be inferred efficiently.

The 5G NR and LTE physical layers share many similarities. They both rely on Cyclic Prefix Orthogonal Frequency-Division Multiplexing (CP-OFDM), and the frame structure is unchanged, with a frame length of 10 ms, split in 10 subframes of 1 ms. However, 5G NR introduces flexibility in both time and frequency domains by introducing a new concept, numerology, denoted by $\mu \in \{0, \ldots, 6\}$. Numerology is a parameter that determines the Subcarrier Spacing (SCS), $\Delta f = 2^{\mu} \cdot 15$ kHz, ranging from 15 to 960 kHz [1]. In the time domain, the slot time duration now scales inversely with numerology as follows: $T_{slot} = 1/2^{\mu}$ ms, and the number of slots per subframe with $N_{slots} = 2^{\mu}$. Thus, each subframe is now formed by N_{slots} of duration T_{slot} each one formed by 12 or 14 OFDM symbols, depending on the CP length [1].

The maximum bandwidth is also increased, and the resource grid in 5G now ranges from 240 to 3300 subcarriers, depending on the bandwidth and subcarrier spacing [6, p. 14]. The smallest physical resource in the grid is referred to as a Resource Element (RE), and corresponds to one subcarrier in one OFDM symbol. REs are grouped into Physical Resource Blocks (PRBs), and one PRB consists of 12 REs.

Naturally, the use of large bandwidths can boost the overall capacity of the network. However, the use of wide bands can significantly impact the operation of low energy devices, such as IoT devices, as they would incur a higher energy usage by operating over larger portions of the spectrum. In order to support different use cases and devices, 5G introduces the concept of Bandwidth Parts (BWPs). A BWP is a set of contiguous physical resource blocks which can be a subset of the total bandwidth of the band. In this way, different UEs can have different BWPs assigned to them with different numerologies, adapting to the user characteristics and requirements. One cell can support up to four BWPs, and a given UE can have only one BWP active at a time [7, p. 359]. A 5G sniffer needs to efficiently infer these various configuration parameters.

B. Synchronization and Cell Information Acquisition in 5G

5G uses a set of always-on signals that are periodically broadcast by the 5G base station (gNB). These dedicated signals serve as a reference for UEs performing the initial steps to connect to the network: cell search, initial synchronization and acquisition of the minimum required system information. 5G reduces the number of always-on signals, e.g., Cell-specific Reference Signal (C-RS) from LTE, and consolidates several signals into a Synchronization Signal Block (SSB). Figure 1 depicts the spectrum of a live 5G gNB downlink transmission. The figure captures the most important channels that our 5GSniffer leverages to extract both cell and user-specific information, the SSB, which is easily recognizable in the spectrum, and carries cell-wide information, and the PDCCH, which includes resource scheduling information per user, and will be discussed in depth in Section II-C.

The SSB occupies 240 subcarriers across 4 OFDM symbols, and contains three key pieces of information: the Primary Synchronization Signal (PSS) and Secondary Synchronization Signal (SSS), well-known BPSK sequences, and the Master Information Block (MIB). During cell search, the UE scans for the SSB and computes the power and quality of the received signal. Once a suitable cell is found, both PSS and SSS signals are used to compute the Physical Cell ID (PCI) and to estimate the time and frequency offsets. A UE (and sniffer) needs to correct for such offsets to adequately synchronize with the 5G gNB downlink transmissions. The next step for a UE is retrieving the MIB. To do so, the UE performs channel estimation, equalization and decodes the MIB, which is encoded with a robust polar coding scheme. MIB is a compact set of 23 bits that carry important cell operation information, such as the System Frame Number (SFN), Cell Barred indication, and a field, `pdcch-ConfigSIB1`, that indicates where to find the next important block of information, System Information

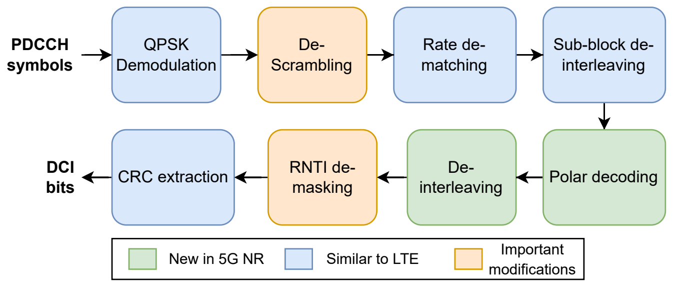


Fig. 2: PDCCH to DCI decoding procedure in 5G NR. Different colors indicate the similitude to its 4G counterpart

Block 1 (SIB1). SIB1 is the last piece of information required by a UE to connect to the network, as this block contains cell selection criteria, information on the random access procedure, and initial BWP and control channel information, which our sniffer leverages. Although SIB1 is also periodically broadcast in a 5G standalone cell, SIB1 is not transmitted through a dedicated channel like the SSB, but through a common channel used for all downlink data transmissions to all UEs [5].

C. Resource Scheduling in 5G NR

User data is carried in physical channels, such as the Physical Downlink Shared Channel (PDSCH) and Physical Uplink Shared Channel (PUSCH). All UEs receive and transmit data on assigned channels. The gNB is in charge of resource scheduling and communicating to the UEs where to find their assigned resources. This (meta) information is conveyed to the users in the DCI, which is carried on a dedicated channel, the PDCCH [5, p. 29]. A downlink portion of a BWP where the PDCCH is transmitted is called a Control Resource Set (CORESET). CORESETs were introduced in 5G NR in order to provide a variable configuration for different use-cases and can be configured through upper layer signalling, i.e., Radio Resource Control (RRC). For instance, CORESETs can span multiple OFDM symbols, with variable number of contiguous subcarriers, and with variable offset and periodicity. A CORESET is configured within a BWP and the maximum number of CORESET configurations per BWP is 3 [7, p. 770]. In Figure 1 we see two different CORESET configurations from a real deployed network, where CORESET 0 is a special type that is configured through the MIB and only contains scheduling information for SIB1. Although both CORESETs seem similar at first glance, they are offset in frequency and their internal configuration, i.e., how the DCIs are recovered from the PDCCH symbols, is substantially different. This is a challenge for a passive sniffer, which needs to infer the control channel configuration.

DCI includes the frequency and time resource allocation, i.e., the number of PRBs and over how many OFDM symbols, and the Modulation and Coding Scheme (MCS) used for the transmission [4, p. 118-121]. Different DCI formats are defined in the standard, with separate DCI formats for uplink and downlink, and for different use cases. For example, PDSCH scheduling uses DCI Formats 1_0 and 1_1, whereas

Format 3_0 and 3_1 are used for sidelink scheduling [4, p. 86]. Naturally, UEs need to know which DCIs are directed to them, this is indicated by the RNTI. The RNTI spans 0 to $2^{16} - 1$, with a subset of values being reserved for common cell information, for instance, values 65534 and 65535 for paging and system information respectively. For user-specific communications, the RNTI used is the Cell-RNTI (C-RNTI), which is assigned to a UE during random access [5, p. 56,94]. This C-RNTI is linked to a RRC connection, and a new C-RNTI is assigned every time an RRC connection is setup. Unless mentioned otherwise, we will refer to C-RNTI as simply RNTI for the remainder of the paper. The RRC connection, and consequently the RNTI, is released by the gNB after the UE is inactive for a period of time configurable by the operator. As scheduling information is broadcast in the cell, and not protected, any passive sniffer can theoretically obtain the resource scheduling of all UEs in a cell. However, as discussed later, the RNTI is not sent explicitly. A UE knowing its own RNTI can confirm a DCI match, but it is a more challenging task for a sniffer as explained in Section III-A.

The first step a UE needs to perform to retrieve a DCI is to determine to which physical resources it has been mapped to. The minimum physical resource unit in PDCCH is termed Control Channel Element (CCE), and it consists of six Resource Element Groups (REGs), where each REG is one PRB in one OFDM symbol. Out of the 12 subcarriers that form one REG, three are dedicated to the PDCCH Demodulation Reference Signal (DMRS), which are used as pilot subcarriers to perform channel estimation. REGs can be bundled according to a given REG bundle size $L \in \{2, 3, 6\}$ and bundles can in turn be interleaved over the bandwidth of the CORESET. CCEs can be aggregated, depending on the Aggregation Level (AL), to accommodate different DCI sizes and adapt to channel conditions by adding redundancy. The number of aggregated CCEs to form one DCI can be 1, 2, 4, 8 or 16 CCEs. The PDCCH DMRS can be configured per UE, through the upper layer parameter `pdcch-DMRS-ScramblingID` $\in \{0, \ldots, 2^{16} - 1\}$, which is used as part of the seed value for initializing the pseudo-random DMRS sequence of OFDM symbols [6, p. 98]. We will use the notation ScramblingID for `pdcch-DMRS-ScramblingID` for the rest of the paper.

The process to obtain the DCI from PDCCH symbols is depicted in Figure 2. Additional details can be found in the 5G standard [4], [6]. For a given AL and CCE-REG mapping, the QPSK symbols are demodulated and the output bits are descrambled with a sequence generated from a Gold sequence of length 31 initialized with $c_{init} = (n_{RNTI} \cdot 2^{16} + n_{ID}) \mathrm{mod} 2^{31}$, where for a user-specific case, n_{ID} is the upper layer parameter ScramblingID, or the cell ID otherwise. Note that in 4G, the scrambling sequence was cell-specific, whereas in 5G this sequence is user-specific, and 4G used convolutional coding instead of polar coding. Subsequently, de-scrambling, rate matching, sub-block de-interleaving and polar decoding are performed. Polar decoder parameters and rate matching, performed by either puncturing, shortening or repetition, depend only on the aggregation level and DCI size. After an additional level of de-interleaving, used to distribute the CRC bits among information bits, the UE confirms that the DCI is intended for it by XORing its RNTI with the last 16 bits of the 24-bit CRC and checking correctness. The final DCI bits are obtained after extracting the 24-bit CRC in 5G, whereas in 4G this CRC consisted of only 16 bits. These differences provide both challenges and opportunities.

III. 5G Sniffer: Challenges and Techniques

We introduce our tool `5GSniffer`, which is, to the best of our knowledge, the first open-source[1] implementation of a promiscuous 5G control channel sniffer for 5G networks. 5GSniffer provides access to the resource scheduling information of all users operating within a live 5G deployed gNB. Given the flexibility in 5G deployments, our tool is configurable for different 5G configurations, such as different SCSs, CORESET configurations, and BWPs. Moreover, it can decode the control channel from both Non-Standalone (NSA) and Standalone (SA) 5G deployments.

As we will discuss in Section III-A, decoding 5G control channel of all users in a cell raises significant *new* challenges that did not exist in LTE. In order to address these challenges, and some of the maturity and stability issues of existing 5G RAN open source implementations [20], [55], we opted for a clean-slate design and implementation supported by our optimizations and scalability techniques. This decision also frees us from design limitations that we would encounter had we implemented our tool on top of existing 5G RAN open source code. As such, our tool is designed for a dedicated purpose, efficiently decoding the 5G control channels of all users in a cell. Note that developing a sniffer is very different from implementing a control channel decoder for a single user, where all parameters for decoding are known.

5GSniffer is a passive tool and follows the principles that a UE, connecting to the network, would perform to obtain system information, as described in Section II-B. However, it will lack some of the control information communicated to the UEs in encrypted RRC messages and requires additional mechanisms to efficiently overcome such limitations. At a high-level, in order to decode the control channel, 5GSniffer requires time and frequency synchronization with the gNB downlink transmission, as well as basic cell information. This is achieved by making use of the information provided by the always-on unprotected 5G signals. Once this is achieved, 5GSniffer searches for possible DCIs while continuously maintaining synchronization with the gNB cell. Further details on the operation of our tool are provided in Section III-C.

Although 4G control channel sniffers have been implemented in the past [36], [15], [22], devising an efficient 5G control channel sniffer is not a trivial task. This is due to peculiarities of the 5G NR scheduling and configuration flexibility. For instance, the encrypted signaling of the RRC layer of 5G NR information element fields such as the ScramblingID as well as CORESET and PDCCH, makes developing a control

[1]The code will be made available at: https://github.com/NorbLd/5GSniffer

channel sniffer significantly more challenging compared to LTE, and a brute-force approach proves unreliable. Thus, our proposed techniques and optimizations are vital to realize an effective 5G sniffer.

A. Challenges

In 4G LTE, the control channel configuration was conveyed through the Physical Control Format Indicator Channel (PCFICH), which was transmitted at the start of each subframe, and was easily accessible to a passive sniffer. The information contained in the PCFICH informs UEs of the number of OFDM symbols used for PDCCH. In contrast, 5G NR's physical layer flexibility offers multiple configurable parameters, which requires dedicated signaling. As such, 5G removed the PCFICH, and parameters such as CORESET and PDCCH configuration, or search space configuration, are explicitly configured through the upper-layer RRC. This information can not be accessed by a passive sniffer, because the RRC messages are encrypted. Therefore, our sniffer needs to blindly determine the configuration of the control channel for a given cell and user. The challenges are the following:

CORESET Physical Configuration. A passive sniffer first needs to determine where to find the PDCCH. The unknown parameters in a CORESET configuration are the frequency and time domain resources allocated to the CORESET, i.e., number of PRBs and number of OFDM symbols the CORESET spans, the ALs used in the CORESET, and the CCE to REG mapping.

DCI Size. DCI formats that include scheduling for a given UE are scrambled with the UE C-RNTI. There are four formats with different DCI sizes that include scheduling in a cell: Formats 1_0 and 1_1 for the downlink, and 0_0 and 0_1 for the uplink. Formats 1_0 and 0_0 are referred to as fallback formats, because they are known without further configuration, and can be used to schedule resources for a given UE or for common resources (System Information, paging, random access). On the other hand, formats 1_1 and 0_1 have variable sizes depending on the cell configuration, and the DCI size is not known a priori to the passive sniffer. The DCI size ranges from 12 to 140 bits, and will determine parameters in the rate matching and polar decoding blocks of the DCI decoder.

RNTI and ScramblingID. Apart from cell-specific parameters, a sniffer needs to blindly decode user-specific parameters such as RNTI and ScramblingID. This raises two key issues: computational complexity, and a high false positive rate. The RNTI is a 16-bit identifier that uniquely identifies a user within a cell, and it is XORed with the 16 last bits of the 24-bit CRC for 5G, or 16-bit CRC for LTE. However, there is a particularity in 5G NR that greatly complicates the task of retrieving a DCI. In LTE, the RNTI was only used to XOR with the CRC, whereas in 5G NR the RNTI is also used as one of the inputs of the scrambling sequence used to de-scramble the PDCCH bits after QPSK demodulation, as shown in Figure 2 [6, p. 99-100]. Thus, although in 4G the RNTI was not a necessary input to perform DCI decoding, and it was obtained after de-masking the RNTI from the CRC, this is not the case for a 5G DCI decoder. To further complicate the task of a passive control channel decoder, the other input to the scrambling sequence used is the ScramblingID, which is also 16 bits and configured through an RRC-encrypted message. Therefore, a 5G sniffer decoder would have to brute-force 2^{31} scrambling sequence combinations to determine the RNTI and ScramblingID, while not knowing if the scrambling sequence was correct until checking the CRC in the last step of the DCI decoding, which involves executing the polar decoder.

Due to the number of unknown parameters to a passive listener, a naive approach, such as brute forcing, would require to perform the decoding process, assuming the CORESET is known, for: each aggregation level (4 bits), each ScramblingID and RNTI (16 bits each) and DCI size, which ranges from 12 to 140 (8 bits). This adds up to 2^{44} combinations, in order to decode each DCI from the PDCCH. This, apart from a computational complexity problem, creates also a reliability problem. As described in Section II-C, a user will consider a DCI valid when the CRC check is correct. The likelihood of obtaining a correct CRC check when the data is in fact not correct (false positive) is very low in the normal operation of a user, as all parameters for decoding are known. However, this is not the case for a sniffer that tries to blindly decode a large set of unknown variables. We find that the number of false positives increases significantly when trying all combinations by brute-forcing. In order to quantify the false positive rate, we carry out a set of experiments by performing brute-force decoding of all combinations of RNTI and ScramblingID (2^{32} values), over a set of DCIs. We find that, on average, the number of false positives found per DCI is 256.3. A passive attacker can not discern which -if any- of these found correct DCIs are valid. Thus, additional techniques and optimizations are needed to implement a reliable control channel sniffer.

Channel Estimation. Lastly, apart from the difficulties of retrieving the DCI, a passive sniffer needs to maintain correct time and frequency synchronization with the 5G gNB and perform channel estimation and equalization to decode PDCCH. Although maintaining synchronization can be achieved by tracking the broadcast SSB signal, an additional challenge arises when it comes to estimating the channel for PDCCH equalization. 5G does not transmit a periodic cell-wide reference signal. Instead, the data and control channels are interleaved with DMRS. Therefore the PDCCH region is interleaved with PDCCH DMRS, that can be used for channel estimation. However, the sequence of pilots is generated with ScramblingID as an input parameter, which is unknown to the passive sniffer, and the sniffer needs to estimate the channel with unknown PDCCH-DMRS pilots. As we will see in Section III-B1, this is both a challenge and a side-channel attack opportunity for a passive sniffer.

B. Developing an Efficient 5G Control Channel Sniffer

Due to the aforementioned challenges, developing a 5G sniffer requires multiple levels of optimization and novel techniques. We now describe some of the techniques we developed to meet the challenges of an efficient control channel sniffer.

1) Optimizing DCI decoding: One of the biggest challenges is to efficiently obtain scrambling parameters, RNTI and ScramblingID, without brute forcing the 31 bit scrambling sequence. We devise novel techniques and optimizations by carefully analyzing 5G NR mechanisms details.

Determining ScramblingID. As discussed earlier, determining this parameter is crucial as it is part of the scrambling sequence following QPSK demodulation. Thus, brute forcing would requires executing the entire DCI decoding chain (see Figure 2) for each possible value. However, we note that the ScramblingID is also used as an input parameter for the pseudo-random sequence used to generate the DMRS accompanying the PDCCH [6, p. 107]. The PDCCH DMRS pseudo-random sequence is generated with the following seed:

$$c = (2^{17}(N_{sym}^{slot} \cdot n_{s,f}^{\mu} + l + 1)(2N_{ID} + 1) + 2N_{ID}) \bmod 2^{31} \quad (1)$$

where l is the OFDM symbol within the slot and $n_{s,f}^{\mu}$ is the slot number within a frame. N_{ID} is equal to the ScramblingID for user-specific DCIs, or the Cell ID value otherwise. Note that here, N_{ID} is the only unknown parameter, as the sniffer is synchronized to the 5G gNB base station. We exploit the fact that ScramblingID is used as an input to generate the DMRS sequence in two ways: (1) Our sniffer determines the value of ScramblingID by computing the correlation between the 5G gNB DMRS pilots and offline pre-computed DMRS signals generated from all ScramblingIDs. In this way, the ScramblingID can be determined before starting DCI decoding, and brute forcing this value is substituted by computing a correlation, a dot product that can be implemented very efficiently with SSE/AVX support. (2) We speed-up the decoding of DCIs and reduce the number of false positives by reducing the number of CCEs and ALs where DCI needs to be decoded from the output of the correlation instead of blindly trying all possibilities, as an acceptable DCI will be accompanied by a valid DMRS. The speed-up provided by this optimization can be of up to 2 orders of magnitude.

Exploiting redundancy in DCI encoding. After de-scrambling, PDCCH bits are passed through the de-matching block, which matches the size of the input bits E, to the size of the input bits of the polar decoder, N. Rate de-matching input spans a set of values depending on the aggregation level, $E = AL \cdot 108$ bits (since each CCE carries 54 QPSK PDCCH symbols). Thus, E can take values 108, 216, 432, 864 or 1728 bits. On the other hand, the input size of the polar decoder, N, can only take specific standardized values for the downlink control channel polar decoder, $N = 128, 256, 512$ input bit rate, which depends on the DCI size, K, and the aggregation level, AL, and is calculated through a set of formulas defined in the standard [4]. Rate matching can be performed either by puncturing, shortening, or repetition. We will focus on the repetition mode, which is used when $E > N$. In this scheme, after polar coding, the first $E - N$ bits are repeated at the end of the bit sequence to add redundancy and match the number of bits necessary for a given aggregation level. We exploit this redundancy of the repetition scheme to determine the correct scrambling sequence. To do so, the guessed scrambling sequence is determined as the sequence that provides maximum likelihood between repeated bits after de-scrambling. This removes the need to complete the entire DCI decoding chain and speeds up the decoding per each DCI by more than one order of magnitude. As ScramblingID is known from our previous technique, only the RNTI needs to be extracted. Naturally, this optimization can only be employed when there is a repetition. As AL values 8 and 16 contain $E = 864$ and 1728 bits respectively, this optimization always works for these aggregation levels. For AL values 4 and 2 this technique depends on the DCI size, K, which needs to be between 12 and 33 bits long for AL 4 and between 12 and 17 for AL 2. For AL 1, repetition is never used. Note that this technique is more beneficial the higher the AL, as the computation effort of blindly decoding DCI increases with the number of bits to process.

Prioritization of RNTI processing. In order to speed up the detection of a valid RNTI, we implement a priority list for RNTIs to be tried, which is reordered based on recently successfully decoded RNTIs and their frequency. In this way, the RNTI list is ordered by the likelihood of RNTI appearing again. To determine this likelihood we verify if the RNTI is still active by how recent the last communication with the user was, i.e., below the RRC idle timing, and how frequent it has been recently. This is motivated by the fact that users communicating non-sporadically, e.g. consuming media services or during a video call, will have resources scheduled more frequently [15], [22]. The speed-up provided by this method depends on the RNTI distribution in the cell, but it can provide a speed gain of approximately 4 orders of magnitude.

Utilizing prior/side information. We conducted a measurement campaign of 5G configurations for different operators (e.g., T-Mobile, AT&T, Verizon, Vodafone), 5G configurations (SA and NSA), and at different locations (North America and Europe). We obtained interesting findings about the configuration of CORESET parameters. For the ScramblingID, we found only three different configurations used across countries. We conjecture that these configurations are set to certain manufacturer-specific defaults. Beyond the implications for fingerprinting of manufacturers, and enabling manufacturer-specific attacks, the limited configuration enables further optimizations of the sniffer. The first configuration we found applies to all users equally, namely that ScramblingID = 1008 + CellID. The second configuration we found is also cell-wide; here the ScramblingID is set to the Cell ID value. The last configuration is the most interesting, as the manufacturer sets the ScramblingID to the user's RNTI plus the Cell ID. This has very interesting implications, as in this case by only computing the correlation with the DMRS symbols, we can simultaneously infer the RNTI and ScramblingID. Therefore the RNTI does not need to be blindly decoded through the DCI decoding procedure. Moreover, by analyzing a large number of RNTI assignments, we observed that some operators only use a subset of the RNTI space to assign RNTIs, for example, RNTI $\in \{15000, \cdots, 25000\}$. Prioritizing this subset then reduces the search time as expected.

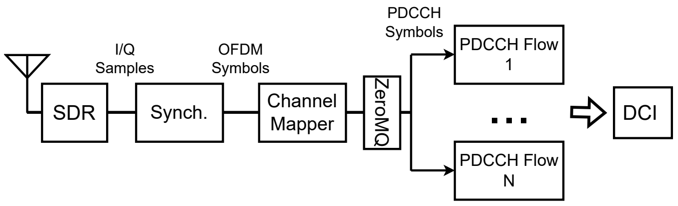


Fig. 3: Simplified schematic of the sniffer operation

2) Determining the cell configuration: Before decoding the DCI, our sniffer needs to learn the cell CORESET configuration. This can be achieved by initially blindly scanning the network, or by obtaining prior information from a device with access to the network. In case that prior information is not available, our sniffer obtains initial information from the MIB, which configures CORESET 0, and from SIB1, which already contains information of the initial BWP and CORESET configuration. Additional configured CORESET information can be obtained by performing an initial attach procedure to the gNB, and listening to a RRC Setup message from the gNB [7, p. 283]. This does not require an operator valid SIM card, as this message is transmitted prior to UE registration. However, some additional CORESETs might not be configured until successful registration, in these cases our 5GSniffer can be used to infer this information blindly. This is performed by scanning for DCIs with all possible CORESET configurations, which will map the cell configuration. We note that in order to map the CORESET configuration, including parameters such as CORESET bandwidth and duration, the sniffer only needs to initially scan for the PDCCH DMRS sequence by correlating with pre-computed values. Although the configuration mapping could take minutes, we found CORESET configurations to be static in the current state of 5G deployments, with the same configuration being used within a span of 6 months. Therefore, even if the cell configuration would have to be determined again, this would be a rare event.

Alternatively, it is possible to extract cell configuration information from smartphones or USB modems with a Qualcomm chip by using the Qualcomm Diag protocol to communicate with the device baseband. Tools such as QCSuper [45] or MobileInsight [39] use the Diag protocol and extract raw cellular frames, which includes RRC protocol messages. Thus, a device such as an inexpensive 5G modem or smartphone with an operator SIM card (e.g. an expired pre-paid SIM) can extract the encrypted RRC messages carrying the cell configuration and use this input for 5GSniffer initial configuration.

3) Architectural design optimizations: In addition, we also designed and optimized the 5GSniffer software and implementation for efficiency and scalability. 5GSniffer is written in C++ and the software is designed for parallelism and even distributed deployment. We designed it with multi-threading as a core functionality to enable computation speed-up. We use ZeroMQ to send different PDCCH symbols to different threads and perform DCI decoding computations in parallel.

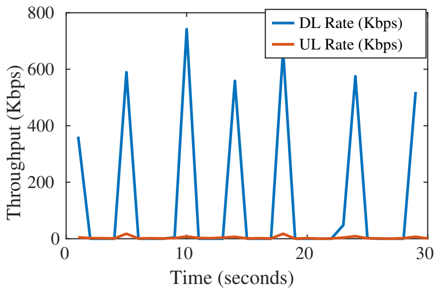


Fig. 4: Target user decoded data rate pattern

Matlab-based	5GSniffer Unoptimized	5GSniffer
15.3 hours	4min27sec	172 ms

TABLE I: Decoding time of a 200 ms recording for a Matlab-based implementation, and 5GSniffer with and without optimizations

Moreover, ZeroMQ can be potentially used to distribute the processing load across multiple servers.

C. Sniffer Flow of Operation & Performance

Figure 3 shows a process flow representation of our sniffer. It can operate in an optimized real time mode or an in-depth mode, taking either an I/Q sample stream from an Software Defined Radio (SDR) or file source as input. To decode the DCIs from the I/Q stream, the following steps are performed.

- The SDR block captures I/Q samples at the bandwidth and center frequency configured and passes them to the Synchronization block.
- The Synchronization block is in charge of finding a cell and synchronizing to it by finding SSB as described in Section II-B. Then, after performing OFDM demodulation, the symbols are passed on to the next block.
- The Channel Mapper block determines which OFDM symbols contain PDCCH symbols based on the configured CORESETs. This block manages the thread pool to distribute the PDCCH symbols across multiple threads.
- The different parallelized PDCCH blocks will process each assigned PDCCH symbol based on their configuration. This includes correlating with PDCCH-DMRS and decoding possible DCIs using the optimizations described in Section III-B1. Finally, the DCIs output by the PDCCH blocks are logged or passed to attacks processing blocks.

In order to provide a decoding time benchmarking comparison, we leverage the MATLAB 5G Toolbox capabilities and develop a basic Matlab-based brute-force sniffer. We then process the same 200 ms recording on Matlab and on our 5GSniffer with and without optimizations. The results provided in Table I display a computational speedup of more than 5 orders of magnitude faster than an implementation based on the Matlab 5G Toolbox, and 3 orders of magnitude faster than a brute force approach. In order to validate our DCI decoding accuracy, we generate 5G waveforms containing DCIs and use it as input to our sniffer. Using this ground truth information, we verify that our sniffer does not miss DCIs under ideal channel conditions.

IV. Attacks on 5G Privacy

Our tool enables decoding of the control channel information, and outputs resource scheduling information of all UEs served by target cells. To illustrate both the security capabilities of our tool and vulnerabilities in the 5G ecosystem, we disclose a set of attacks that leverage the information extracted with our tool.

A. Network Traffic Analysis

Control channel scheduling information includes the allocated resources in time and frequency, and how many bits each resource carries (derived from the MCS). From this information, a passive sniffer can compute the number of bits allocated to a user for an instant of time, i.e., the data rate, for both downlink and uplink. By tracking this information over time, an adversary can extract traffic patterns and monitor a user of interest. This can also serve as a stepping stone for follow-up attacks, such as encrypted traffic fingerprinting [34], website fingerprinting [47] or video identification attacks [10]. Figure 5 represents an example of traffic patterns recovered by our sniffer by listening to the resource allocation of a 5G gNB which our test device is attached to. We perform different types of traffic in our test device, a voicecall, which is characterized by a sustained downlink and uplink traffic between 20 and 100 Kbps, a videocall, which requires higher throughputs depending on the video quality, video streaming, which can be differentiated by consistent downlink traffic, between 150 and 300 Kbps in this case, and low uplink traffic, and finally a file transfer, which is characterized by a substantial spike in data rate over a short period of time. In the next section we describe how our tool and traffic analysis capabilities can be used to carry out a more sophisticated attack, identifying a target user through the RNTI.

B. Abstract User Identification

In a user identification attack, the goal of the adversary is to determine whether a given user is present in a certain area. To achieve this, the adversary must be able to link the identity of the user to the identity of their device. Such attacks have been demonstrated in previous work for LTE, UMTS and GSM using both active and passive approaches. Active approaches typically operate by setting up an International Mobile Subscriber Identity (IMSI) catcher [56] and intercepting the victim's IMSI or TMSI whenever they connect to the attacker's base station. On the other hand, passive attacks attempt to link these identifiers to some application-layer identifier, for example by generating traffic to a phone number and observing the resulting downlink paging message [18].

We now describe an attack that allows an adversary to identify the presence of a particular user connected to the 5G RAN using our sniffer. This attack relies solely on the physical-layer RNTI and therefore does not require knowledge of any higher-layer identifiers such as the Globally Unique Temporary Identifier (GUTI), IMSI, or International Mobile Equipment Identity (IMEI) as in previous attacks. As such, usage of the SUCI, IMEI randomization, or changing the SIM card do not prevent the victim from being tracked. Furthermore, our attack is completely passive on the radio level, and cannot be detected by inspecting messages transmitted on the 5G RAN.

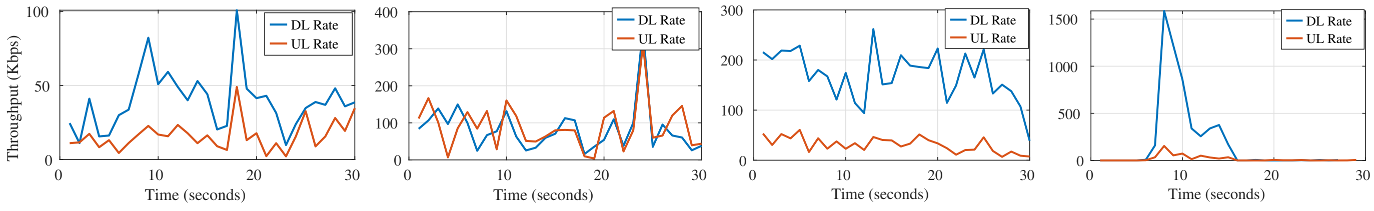
Adversary Model. The following requirements must be met in order to perform the attack:

- The adversary is able to successfully receive downlink messages intended for the victim, i.e., they are within range of the gNB that is serving the victim's UE. This requirement is more relaxed compared to for example IMSI catching [56], downgrade [52], or other Man-in-the-Middle (MitM) attacks [17], [49], where the adversary must be in proximity to the *victim* (e.g. to have them connect to a rogue base station).
- The adversary has knowledge of at least one virtual identity of the victim, for example a Twitter handle, Facebook account, email address, or phone number. These can typically be acquired through Open-Source Intelligence (OSINT) or social engineering. In addition, the victim must have an app installed on their smartphone that automatically interacts with any one of these identities. Most modern smartphones, by default, receive push notifications whenever an identity is interacted with.
- The victim has mobile data enabled, and is able to receive IP traffic for the apps associated with their virtual identities through 5G RAN at a rate higher than the idle timer. This restriction is discussed later in more detail.

To perform the attack, the adversary uses our 5G sniffer to eavesdrop on the PDCCH of the gNB serving the victim. This can be achieved by placing a SDR running our tool within range of the gNB. Next, the adversary artificially generates downlink traffic towards the victim by interacting with any of their online identities. For example, the adversary could send a series of instant messages, tweets, or emails to the victim.

Since the adversary knows both the interval and size of the transmitted messages and since they are received automatically by the victim's smartphone (e.g., through push notifications), this activity can be correlated to the downlink resource allocations to determine the RNTI used by the victim. Consequently, the adversary can confirm the presence of a certain online identity in an area, without relying on high-layer identifiers. Figure 6 shows a graphical representation of an example attack where an adversary transmits Signal messages at a 2.5-second interval towards a target phone number. The resulting downlink resource allocation pattern for the targeted user's RNTI is easily recognizable as shown in Figure 4.

It is important to note that in order for the attack to succeed, the apps running on the device of the victim must be able to receive data at a higher rate than the idle timer configured by the mobile operator. This timer determines when a UE is considered idle and a new random access procedure must be performed, thereby assigning a new RNTI to the UE. Since the adversary wishes to identify a user based on the sequence



(a) Voicecall (b) Videocall (c) Video streaming (d) File transfer, 10 MB

Fig. 5: Measurements carried out with 5GSniffer of throughput traffic patterns over time enable network traffic analysis

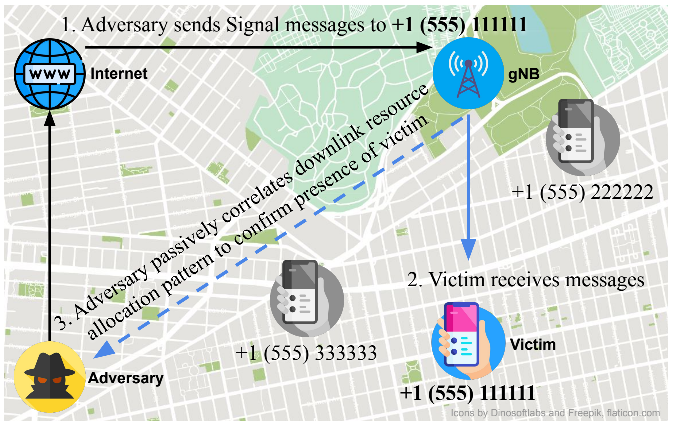

1. Adversary sends Signal messages to +1 (555) 111111
Internet
gNB
+1 (555) 222222
3. Adversary passively correlates downlink resource allocation pattern to confirm presence of victim
2. Victim receives messages
+1 (555) 333333
Victim
Adversary
+1 (555) 111111
Icons by Dinosoftlabs and Freepik, flaticon.com

Fig. 6: Example of a user identification attack performed by sending Signal messages and passively sniffing using our tool. The adversary infers the victim's RNTI by correlating downlink resource allocations to transmitted Signal messages

of downlink messages containing the victim's RNTI, it should remain the same for at least the length of this sequence.

Note that the attack described is vastly different from a purely-physical attack, where an adversary using a setup similar to a radar, with highly directional antennas capable of discerning the location of signals emitted by other devices, would focus on the uplink (lower power) to find a specific user. In such scenario, an attacker would have to scan all directions with very narrow beams and resolve all users in a cell. Also, an attacker should be able to predict the exact time slot at which a message will be acknowledged by a specific user. All these challenges are hard to overcome in real scenarios.

C. Stealthy Tracking by Exploiting Privacy-Focused Messengers

While the above methods for generating downlink traffic towards a specific user are effective, they are not very stealthy; most apps interacting with the online identities will show a notification to the user whenever a new message is received. For example, on most modern smartphones, sending an email to a victim's email address will by default result in a notification from the installed mail app, thereby tipping off the target.

In previous works, several methods have been proposed that allow an adversary to inject traffic towards a user, such that they are not notified. These methods include placing silent phone calls [37], sending silent SMS [44], [18], sending a WhatsApp typing notification [50], and sending messages to the Facebook "Other" folder [50]. However, these approaches have several shortcomings in the context of tracking a user on the 5G physical layer, as they were primarily designed to trigger paging messages rather than generate downlink traffic. Specifically, placing a silent phone call or triggering a WhatsApp typing message does not generate significant downlink resource allocations, which makes it hard for an attacker to uniquely differentiate the target RNTI. Further, silent SMS and phone calls can be trivially detected on phones with an exposed Qualcomm DIAG interface, using mobile security apps such as SnoopSnitch [54]. Finally, while sending a Facebook message to strangers does not raise a notification, the number of messages that can be sent in this way is limited [21], and the victim can still see the messages sent to them in the Facebook app [50], which could raise suspicion.

To overcome these issues, we show that an adversary can exploit a design vulnerability in privacy-focused instant messaging apps in order to stealthily inject traffic onto the network, without triggering notifications. This allows an adversary to track users over long periods of time without their knowledge. We provide a proof-of-concept implementation for Signal and Telegram. These apps were chosen because their client implementations are open source, and can therefore be modified to incorporate custom functionality. They also have a very large user base in excess of 500 million users. However, other messenger apps may be vulnerable as well. For instance, WhatsApp relies on the Signal Protocol [59].

1) Signal: The Signal protocol uses the Double Ratchet algorithm for exchanging messages using a shared root key. If the two communicating parties do not yet possess a root key, it is first established using the X3DH protocol [42]. The recipient does not need to be online for the X3DH key exchange to take place, as the sender can download the recipient's "prekey bundle" containing their public key from the Signal server. As such, the sender can immediately send an initial message that is end-to-end encrypted with an AEAD cipher to any receiver [43, p. 6]. Since the phone number is used as the account username in Signal, only knowledge of the recipient's phone number is required. We discovered that the Signal client's implementation of this protocol can be exploited by an adversary in order to generate stealthy downlink traffic towards the victim without triggering a notification. The attack was successfully performed against the latest Signal version at the time of writing (5.39.3 on Android and 6.40 on iOS).

To perform the attack, the adversary creates a modified version of the Signal client where the initial message in a chat is transmitted with a non-empty text field and an invalid MAC. Alternatively, we found that the adversary can also create a message that has an empty text field and valid MAC, but is padded with arbitrary junk data. The larger the message, the easier it will be for the adversary to detect peaks in the downlink resource allocations. In practice, we found the size of a single message to be limited to about 196 kB. Larger sizes are rejected by the Signal server with a rate limiting error.

Since the message is end-to-end encrypted with an AEAD cipher, the MAC or format of the message cannot be validated by the Signal server and is accepted. Next, the server forwards the message to the victim, resulting in an increase in resource allocations. Upon decryption, the receiver's Signal client finally discards the message due to its invalid format, i.e. an empty text field or invalid MAC. Interestingly, with this approach, delivery reports are still transmitted back to the adversary, meaning the adversary can check whether or when the stealth message was received. If the victim owns multiple devices that use the same Signal account, a delivery report will be sent for each device that received the stealth message. Therefore, the adversary also learns the number of devices the victim has Signal installed for the targeted phone number.

2) Telegram: Telegram makes a distinction between private chats and secret chats. The latter being end-to-end encrypted whereas the former is not. In order to start any type of chat, the phone number of the victim can be used, analogously to Signal. We first attempted a similar approach to the one we applied for Signal, where invalid messages are sent through an end-to-end encrypted secret chat towards the victim. On Android, there is no notification for setting up the secret chat, and the Telegram client replaces empty messages with the string "Empty message", while also displaying a notification. However, by sending a message that is corrupted inside the encrypted envelope, the message is silently discarded. On iOS, a notification is shown that disappears after about two seconds. The maximum size for messages in Telegram is 65535 bytes, and since Telegram does not support end-to-end encrypted group chats, they can only be directed towards a single user.

In order to overcome these limitations, we investigated private chats as well. These chats are only ciphered between the client and Telegram server. Unfortunately, since the server can now inspect the contents of the message, the corrupted message is detected and removed before forwarding it to the victim. Therefore a different strategy must be employed. By inspecting the source code of the Telegram client, we discovered that messages have a `disable_notification` flag, which when set prevents a notification from showing at the receiver. Another feature in Telegram is that messages can be remotely removed from a chat by the sender. An adversary can combine these two features to send stealthy downlink traffic of arbitrary size to any recipient by sending a large message, e.g. an image, with the `disable_notification` flag set, and then remotely deleting the message programatically immediately after receiving the delivery report. We experimentally confirmed that the above method increases the resource allocations for the victim proportionally to the size of the message, without showing a notification on Android. Similarly to our attack against secret chats, on iOS a notification is briefly displayed despite the `disable_notification` flag, but it disappears shortly afterwards. This was tested against the latest versions of Telegram at the time of writing, i.e., version 8.7.4 on Android and 8.7.1 on iOS.

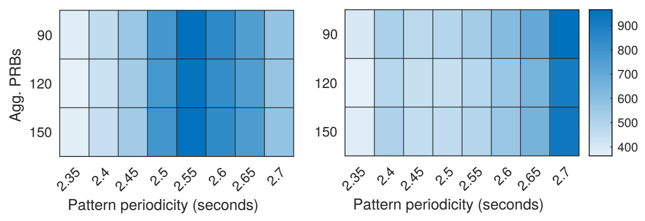


Fig. 7: Pattern identification correlation with varying sine wave frequencies at two different times, shows a distortion of the transmitted pattern frequency as a function of load

D. Identifying Traffic Patterns from Scheduling Information

An attacker can use stealthy messages to generate downlink traffic to a target user with unique patterns. The pattern needs to take into account distortion artifacts of the scheduler, e.g., delay and reduced number of resources allocated if a cell is congested, and noise from background traffic due to other apps. Therefore, the attacker has to design a traffic pattern robust against noise and variable scheduler delays.

In our evaluation, we use the pattern depicted in Figure 4. It sends messages every 2.5 seconds, ensuring the user remains with an active RRC connection, as the communication is more frequent than the RRC inactivity timer [15], and the peaks are sharp and distanced, thus distinguishable under traffic noise. This pattern can be modeled as a modulated on/off signal with amplitude A and period T. To identify the pattern, the attacker computes the resource allocation over time for a given RNTI, and correlates with the expected pattern. We use a sine wave for the filter-matching pattern, to tolerate variations in the cell load. In particular, when the traffic peaks spread over time, due to increased traffic, they still partially contribute to the correlation unlike when using a square wave function. Moreover, we find that due to scheduler delays, specially under higher loads, the periodicity of the computed pattern has a variable offset as high as half a second, and the amplitude of the signal can also vary. This can be seen depicted in Figure 7. As such, the correlation is computed over a set of sine waves with varying frequency around the expected pattern's frequency, and the maximum correlation across frequencies is taken. Furthermore, the mean of the pattern is subtracted to eliminate background traffic noise.

V. Evaluating 5G Tracking in the Wild

To validate the practicality and reach of our attack, we perform an experimental validation in a real world scenario, such as the one represented in Figure 6.

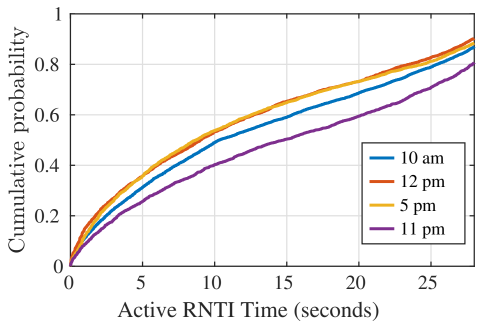


Fig. 8: Empirical cumulative distribution function of the time span of RNTI active connections at different times of the day

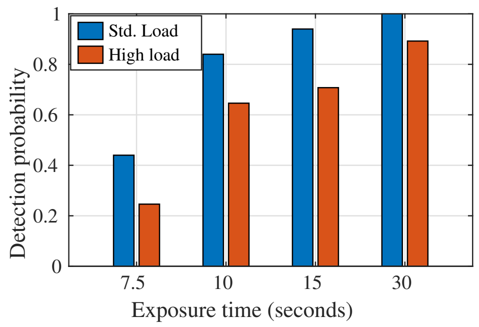


Fig. 9: Probability of detecting the victim for high (peak hour) and standard (off-peak) cell load for different exposure times

A. Experimental Setup

Our experimental setup is composed of the following elements: the adversary, which consists of a laptop running both our modified messenger (Signal/Telegram) client sending stealthy messages and our control channel sniffer, 5GSniffer, connected to a SDR, a USRP B210. We record and store the raw IQ samples in volatile memory (ramdisk) until the successful conclusion of each of our experiments, and run our decoder offline. This ensures that users' privacy is never compromised. Further information on ethical considerations can be found in Section VI-D. As for the victim device, we choose a smartphone with 5G capabilities, a Google Pixel 5, due to its popularity and 5G SA mode compatibility. Moreover, its Qualcomm chipset allows us to extract the ground truth RNTI allocated to the device using Network Signal Guru [46]. We use the ground truth RNTI to validate if our attack was successful. Lastly, the commercial 5G gNB giving service to our 5G smartphone operates in 5G SA mode in band 71 (600 MHz) operating over a 10 MHz bandwidth with SSB periodicity 20 ms and SCS 15 kHz. The cell is configured with 2 different CORESET configurations over this bandwidth and 4 different DCI sizes, two for uplink and two for downlink.

B. Results

To estimate for how long an adversary needs to perform the attack, we first analyze how long RNTIs stay active in an operator cell. We use our 5G sniffer to derive the time span of each RNTI active connection. We perform this measurement for different times of the day to measure the RRC connection time as a function of number of users and traffic activity. Figure 8 presents the empirical cumulative distribution function of the RRC connection time. The average number of users over the measurements are between 150 and 170 active users. We find some noticeable differences in the duration of RRC connections at different times of the day. Noon and 5 pm exhibit similar distributions, with 64% of the communications lasting less than 15 seconds, having sporadic communications. This value drops to 58% and 49% for 10 am and 11 pm respectively, and the proportion of communications that last longer at night is higher; although there are less communications, 40% of them last longer than 20 seconds for 11pm, whereas 32% of the communications at 10 am, and 27% of them at 12 pm or 5 pm last longer than 20 seconds.

Next, we evaluate the performance of our end-to-end attack in a real world scenario. We place our target device under coverage of a 5G gNB, the Signal-to-noise Ratio (SNR) of the target device is between 8 and 20 dB SNR. While the phone is attached to the network, we use our modified Signal app to transmit our pattern described in Figure 4 by sending stealthy 196 kB messages and using Network Signal Guru (for ground truth purpose), we record the RNTI assigned to our test device. We take spectrum recordings and using our tool we infer the resource scheduling for all RNTIs in the cell. We then compute the detection probability or true positive rate (TPR), as the number of times we succeeded in identifying correctly our test RNTI from the scheduling pattern across all experiments per scenario. We carry out the experiments 100 times per scenario.

We analyze the impact of the network load and its implications on resource scheduling and on the attack performance. We perform our experiment attack at two different times, when the load of the cell is relatively high in our given location, i.e., peak hour, and during off-peak hours. Figure 9 depicts the detection probability as a function of the victim exposure time to the attack for high and standard load. We find that for very short exposure times, i.e., below 10 seconds, the pattern is barely recognizable and the detection probability is below 50%. However, already at 10 seconds of attack, the probability of detecting the target user increases to 84% and 64% for standard and high load respectively. The detection probability of the standard load case increases to 94% and 100% as the exposure time increases to 15 and 30 seconds respectively, whereas the detection rate achieves 90% for 30 second exposure time for the high load scenario.

Similarly, we perform measurements to determine the False Positive Rate (FPR) of user detection, defined as the probability of detecting a user when the user is not present. Our results show that the FPR is greatly reduced as the exposure time increases. In this way, we find that for a very short exposure time, 7.5 seconds, the FPR is 14.5%, and decreases to 7% already for 10 seconds. This values further decrease to 4% and 2% when the exposure time is increased to 15 and 30

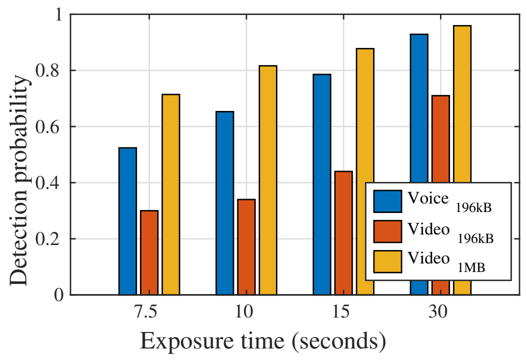


Fig. 10: Probability of detecting the victim while the user is actively using the network, during a voicecall and video streaming for different exposure times. For videostreaming, our traffic pattern is generated with 196 kB or 1 MB messages

seconds respectively, indicating high accuracy with low false positive rate for recordings of more than 15 seconds.

In order to analyze the impact of background traffic in our attack performance, we actively generate traffic in our target device while we perform our attack. We generate traffic through two different means, a voice call and video streaming, which has higher downlink resource requirements. For the video streaming experiments, we evaluate the detection probability by transmitting a pattern generated with 196 kB and a pattern generated with 1 MB transmissions. Figure 10 shows the probability of detecting the victim UE for different recording times for background traffic video and voice. From the results we can see that background traffic has an impact on the detection probability of our attack. However, for voice traffic, an attacker is still able to detect the presence of its victim 92% and 78% of the time for 30 and 15 second of exposure time respectively. A user generating video traffic is a more challenging scenario, as it is using considerable more resources in the downlink, possibly overshadowing our pattern. In this way, with a 30 second recording our attacker is able to succeed 71% of the times, but this probability drops to 44% for a 15 second recording. Our results show that the attack can be performed reliably even in this challenging scenarios if the traffic transmitted is increased. By transmitting 1 MB, the probability of the attack is increased above 80% already for a 10 second recording, and this value increases to 96% for a 30 second recording. Note that although transmissions using the modified Signal client have a limit of 196 kB, this can be achieved using Telegram as described in Section IV-C2.

VI. Discussion and Countermeasures

A. Practicality of User Identification Attacks

In order to perform a user identification attack in practice using the approach detailed in this paper, a number of challenges must be overcome, which we will now briefly discuss.

First, the attacker must continuously transmit data towards the UE of the victim to prevent the RNTI from changing, without hitting any rate limit of the used application. In practice, this can be trivially achieved: the volume of data does not need to be large, provided data can be delivered on a frequent basis to prevent the RRC inactivity timeout (which we observed to be configured between 10 and 30 seconds). In our Signal and Telegram experiments, we did not hit a rate limit for these apps as long as the instantaneous rate and single message size is below some vendor-defined threshold, even after several hours of transmitting. Nevertheless, it should be noted that the user may use any combination of applications or protocols to deliver data to a victim. In this work we only provide a proof-of-concept for stealth messages using Telegram and Signal, but we surmise the true attack surface to be much larger, especially if the attacker is not interested in remaining covert.

Second, the attacker must be located near the victim's current gNB in order to sniff downlink allocations for their RNTI. In practice, ideal locations to sniff could be determined based on OSINT related to the victim (e.g. home town, EXIF data, visited places on social media, etc.). In some cases, note that the attacker may also be interested in determining whether the victim is *not* in a particular location.

Finally, while this paper considered tracking a single target, we postulate that tracking multiple targets would also be practical. For example, an attacker could create Signal accounts using Google Voice or temporary SMS accounts, and send different modulated traffic patterns towards each targeted user.

B. Additional Stealthy Messages Attacks

In this paper, we limited our experiments related to stealthy tracking attacks detailed in Section IV-C to the identification and tracking of a specific target through the 5G RAN. However, we conjecture that the consequences of the discovered vulnerabilities are more far-reaching, and can lead to additional attacks outside the scope of this paper.

Resource exhaustion attacks. A side-effect of stealthy messages such as the ones described in previous sections, is that the receiving client application will perform some processing in order to parse the message contents. In the case of Signal for example, the client will first perform a MAC validation and decryption of ˜196 kB of data, before the discarding the data as invalid. Since these messages can be programatically transmitted by an adversary on demand over a long period of time, various resource exhaustion attacks can be performed, such as exceeding a victim's mobile data plan limits, waste of memory and CPU cycles, remote draining of a victim's phone battery, and saturation of the operator cellular network.

Public chat group privacy. Both Signal and Telegram support the creation of private and public chat groups ("New Groups" with a shareable link for Signal, and "Public Groups" for Telegram). An adversary could join a specific public group and perform stealth messaging attacks in order to generate downlink traffic towards all group members. Since certain online communities use group chats to communicate and coordinate (e.g., activists groups), the techniques described in this paper could be abused by an adversary to identify, exhaust resources, and locate individuals who are members of specific communities of interest.

Non-cellular technologies. Stealth messaging attacks could also lead to the linking of a phone number to other, non-cellular identifiers, as long as the adversary is in a position to passively intercept downlink traffic. For example, note that the same principles discussed in Section IV-C could be applied to identify the random MAC address of a phone connected to an encrypted Wi-Fi network. This may even extend to wired networks if the victim has a messenger app installed on a desktop for example. We believe further exploration of this topic would be an interesting avenue for future research.

C. Countermeasures

The fact that the RNTI remains static during an RRC connection, and that the victim's correct RNTI can be blindly derived as described in Section III-B plays an important role in enabling privacy attacks on 5G users. Given this observation, we now propose a number of countermeasures to mitigate these vulnerabilities. While some of these countermeasures require changes on the specification level that would need to be incorporated in a future 3GPP release, others can be readily implemented by application developers.

RNTI obfuscation and randomization. Instead of using a unique RNTI per RRC connection, the network could assign multiple RNTIs and choose randomly which RNTIs to use for each transmission, hindering traffic analysis per user for an attacker. However, this approach would increase computation overhead at the UE, who has to scan for multiple RNTIs. Alternatively, the RNTI could be changed frequently during an active RRC connection through secure communications. We believe this to be feasible, as a new field `newUE-Identity`, which is sent encrypted through upper-layers, has been included recently for other purposes in the standard, and effectively assigns a new C-RNTI [7, p. 28].

Increase scrambling sequence seed size. By increasing the bit size of ScramblingID and the RNTI, which together form the scrambling sequence seed, our attack would require more correlations to uncover the ScramblingID and more decoding iterations to brute-force the RNTI. Depending on the resources of the attacker, this can make blind decoding the DCIs computationally infeasible.

Alerting users and raising awareness. The previous two countermeasures we discussed would require changes to the 5G standard in order to mitigate privacy attacks, which may take a long time to implement. In the meantime, developers of popular messaging apps could roll out a patch that notifies users whenever an invalid message is received by the messenger client. At the same time, awareness regarding identification and tracking attacks should be raised such that a user can recognize when receiving such notifications is cause for concern. While these measures do not prevent an attack from occurring, the fact that the adversary can no longer stealthily perform the attack means that a user can decide to switch off their mobile phone or report the incident to authorities. Similar countermeasures were previously implemented for silent SMS attacks. An example is SnoopSnitch, an app that warns users whenever a silent SMS is received [54].

Traffic shaping applications. In order to link a device identity to a user's online identity, our attacks rely on IP data delivered towards the user's device reliably. Application developers could offer a secure mode where the notifications are postponed or delayed by a random offset, mitigating the traffic shaping capabilities of an attacker. While such countermeasures have an impact on user experience, this may be a trade-off that privacy-conscious users are willing to take.

D. Ethical Considerations

Since our experiments were performed on commercial 5G networks, we put the highest level of care to ensure that the users' privacy is never compromised and no disruption of regular service occurred. Specifically, for the operators, we note that passive sniffing cannot disrupt their networks. For the privacy attacks discussed in Section IV, experiments were only performed using test phones under our control. Stealth messages were only transmitted to phone numbers owned by the authors, and traffic analysis was performed only on the RNTIs of test phones. The correct RNTI to monitor in this case was determined by extracting it from the phone using Network Signal Guru [46]. The remainder of collected data, such as the active RNTI time shown in Figure 8, consists only of *aggregate* statistics about regular users, which does not affect their individual privacy. While we acknowledge that, in the process of deriving these statistics, sensitive encrypted information of users could inadvertently be contained within the gathered I/Q samples, these samples were stored on a temporary in-memory storage medium (ramdisk), processed only for the purposes outlined in this study, and subsequently deleted after completion of our measurements. This ensures no invasion of user privacy occurred or can occur in the future. Finally, we disclosed the identified messenger vulnerabilities to Signal and Telegram, following responsible disclosure guidelines.

VII. Related Work

Cellular control channel sniffers. The first work to present a fine-grained control channel sniffer capable of decoding active RNTIs and monitor downlink traffic is LTEye [36]. It was followed by multiple open-source tools that improved the decoding performance, reliability and speed, such as OWL [15] and FALCON [22]. Commercial sniffers are available, although scarce and expensive, such as Airscope [53] or Wavejudge [32]. To the best of our knowledge, our work is the first open-source efficient 5G control channel sniffer in the literature that solves the new challenges in 5G DCI sniffing.

Detecting the presence of a user. User detection can be performed actively, for instance by means of IMSI catching [56], downgrade [52], or other MitM attacks [17], [47], [49], [18], and recently, 4G stealthy user tracking by overshadowing both downlink and uplink channels [35]. Passive localization through unprotected broadcast channels has also been extensively researched in cellular networks. Paging has been exploited for user tracking since 2G. The authors in [37] show its possible to track a specific user by performing multiple silent phone calls to a target phone number and linking it to a TMSIs that appears several times in the observed sniffed paging messages, as TMSI is changed infrequently. Authors in [50] demonstrate tracking paging attacks are also possible in 4G networks, as the GUTI is not reallocated between paging instances. They further expand paging tracking by using stealthy messages generated through social media apps instead of silent SMS, due to the availability of multiple tools to detect such attacks. Even when GUTI is indeed re-allocated, authors in [25] find that operators' implementation of GUTI reallocation in 4G is predictable and can be exploited to track users. Moreover, authors in [27] show that the fixed paging occasions (computed from IMSI) in 4G networks can be exploited to associate a victim's identity, e.g. phone number, to paging occasion, allowing an attacker to determine the presence of a user even when GUTI reallocation is enabled. Furthermore, the authors also disclose a new attack that forces the network to page a user with its IMSI in some exceptional cases. Note that 5G enhanced the security of the paging channel to mitigate given attacks. The paging occasion in 5G is now computed from the TMSI [2, p. 35] instead of IMSI, and the TMSI is now compulsorily changed after each paging [9, p. 99]. Moreover, long-term identifiers such as IMSI cannot be used as paging identities in 5G.

In the context of identifying users using radio network identifiers, the authors in [47] show it is possible to map the RNTI to the TMSI by listening to the random access channel and retrieving the RRC Connection Request, which includes the TMSI. However, the authors indicate it is not a targeted attack as it does not link to a specific user unless used in conjunction with previously mentioned paging attacks. Moreover, as a precondition, the user needs to be idle and have a valid TMSI. Specifically, our user detection attack does not require any knowledge of such identifiers, and relies solely on the RNTI. Furthermore, as the attack does not rely on paging vulnerabilities, it is still valid in 5G systems, whereas other paging attacks are no longer valid due to the paging enhanced protection previously mentioned. Furthermore, our attack is not constrained to idle users, and can be performed if the user is currently active. Similarly to [47], the work by Jover [30] mentions that it is possible in 4G to map the RNTI to the TMSI by means of silent SMS, but does not provide experimental evaluation of its effectiveness in the real-world, where many challenges such as other user traffic interference or scheduling inconsistencies can deem the mapping ineffective.

Additional user-privacy threats. Apart from compromising user privacy through tracking users, cellular networks have been target of multiple other attacks on user-privacy. For instance, traffic fingerprinting, such as video identification [10] or website identification [47], [34], eavesdropping calls [48], DNS redirection to malicious websites [47] or exposing device capabilities [51]. Many threats on security and privacy are a consequence of cellular network design, such as unprotected broadcast channels [28] or lack of integrity protection of the control plane [49]. Thus, considerable efforts have been devoted into developing mechanisms to automatically detect vulnerabilities in cellular networks [31], [29], [33], [12].

VIII. Conclusion

In this work, we present the first open-source tool capable of efficiently sniffing the 5G control channel, 5GSniffer. We discuss the challenges of decoding the 5G control channel, how it is significantly harder than in LTE, and we present a series of techniques we devise to provide real-time reliable decoding. We also discover that privacy-preserving messengers, such as Signal and Telegram, can be exploited to generate stealthy traffic to a target user. We pair the traffic analysis of our control channel decoder with stealthy messages to determine the presence of a user in a cell. Our experimental evaluation shows that our attack is reliable, and the success rate scales with exposure time to the attack. The attack reaches a 94% true positive rate of detection, and 4% false positive rate for only 15 seconds of exposure, which improves to 100% and 2% respectively within 30 seconds of exposure time.

Acknowledgment

This work was partially supported by grants NAVY/N00014-20-1-2124, NCAE-Cyber Research Program, and NSF/DGE-1661532.

References

[1] 3GPP. LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical channels and modulation (3GPP TS 36.211 version 14.2.0 Release 14), 2017.
[2] 3GPP. 5G; NR; User Equipment (UE) procedures in idle mode and in RRC Inactive state (3GPP TS 38.304 version 16.1.0 Release 16), 2020.
[3] 3GPP. Technical specification group services and system aspects; release 16 description; summary of rel-16 work items (tr 21.916 release 16), 2020.
[4] 3GPP. 5G; NR; Multiplexing and channel coding (3GPP TS 38.212 version 16.4.0 Release 16) , 2021.
[5] 3GPP. 5G; NR; NR and NG-RAN Overall description; Stage-2 (3GPP TS 38.300 version 16.4.0 Release 16), 2021.
[6] 3GPP. 5G; NR; Physical channels and modulation (3GPP TS 38.211 version 16.2.0 Release 16) , 2021.
[7] 3GPP. 5G; NR; Radio Resource Control (RRC); Protocol specification (3GPP TS 38.331 version 16.1.0 Release 16), 2021.
[8] 3GPP. 5G; Vehicle-to-Everything (V2X) services in 5G System (5GS); Stage 3 (3GPP TS 24.587 version 16.3.0 Release 16), 2021.
[9] 3GPP. 5G; Security architecture and procedures for 5G System (3GPP TS 33.501 version 16.10.0 Release 16), 2022.
[10] Sangwook Bae, Mincheol Son, Dongkwan Kim, CheolJun Park, Jiho Lee, Sooel Son, and Yongdae Kim. Watching the Watchers: Practical Video Identification Attack in LTE Networks. In *31st USENIX Security Symposium (USENIX Security 22)*, Boston, MA, 2022.
[11] Alcardo Alex Barakabitze, Arslan Ahmad, Rashid Mijumbi, and Andrew Hines. 5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges. *Computer Networks*, 2020.
[12] David Basin, Jannik Dreier, Lucca Hirschi, Saša Radomirovic, Ralf Sasse, and Vincent Stettler. A formal analysis of 5g authentication. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, New York, NY, USA, 2018. Association for Computing Machinery.
[13] Ravishankar Borgaonkar, Lucca Hirschi, Shinjo Park, and Altaf Shaik. New Privacy Threat on 3G, 4G, and Upcoming 5G AKA Protocols. *Proceedings on Privacy Enhancing Technologies*, 2019(3):108–127, July 2019.
[14] Marc Briceno, Ian Goldberg, and David Wagner. GSM Cloning, 1998.
[15] Nicola Bui and Joerg Widmer. OWL: A Reliable Online Watcher for LTE Control Channel Measurements. In *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*, ATC '16, New York, NY, USA, 2016. Association for Computing Machinery.

[16] Agnes Chan, Xin Liu, Guevara Noubir, and Bishal Thapa. Broadcast control channel jamming: Resilience and identification of traitors. In *2007 IEEE International Symposium on Information Theory*, 2007.

[17] Merlin Chlosta, David Rupprecht, Christina Pöpper, and Thorsten Holz. 5G SUCI-Catchers: Still Catching Them All? In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '21, page 359–364, New York, NY, USA, 2021. Association for Computing Machinery.

[18] Simon Erni, Martin Kotuliak, Patrick Leu, Marc Röschlin, and Srdjan Capkun. AdaptOver: Adaptive Overshadowing Attacks in Cellular Networks. *arXiv*, pages 2106–05039, 2021.

[19] Simon Alexander Erni. Protocol-aware reactive lte signal overshadowing and its applications in dos attacks. Master's thesis, Department of Computer Science, ETH Zürich, 2020.

[20] EURECOM. Openairinterface 5G Wireless Implementation. https://gitlab.eurecom.fr/oai/openairinterface5g, 2020.

[21] Facebook. Limits to sending messages on Messenger. https://www.facebook.com/help/messenger-app/1425082951086094, August 2022. Retrieved: August 18, 2022.

[22] Robert Falkenberg and Christian Wietfeld. FALCON: An Accurate Real-Time Monitor for Client-Based Mobile Network Data Analytics. In *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019.

[23] Koorosh Firouzbakht, Guevara Noubir, and Masoud Salehi. On the performance of adaptive packetized wireless communication links under jamming. *IEEE Transactions on Wireless Communications*, 13(7), 2014.

[24] Caroline Frost. 5g is being used to perform remote surgery from thousands of miles away, and it could transform the healthcare industry. *Business Insider*, 2019.

[25] Byeongdo Hong, Sangwook Bae, and Yongdae Kim. GUTI Reallocation Demystified: Cellular Location Tracking with Changing Temporary Identifier. In *Proceedings 2018 Network and Distributed System Security Symposium*, San Diego, CA, 2018. Internet Society.

[26] Syed Rafiul Hussain, Omar Chowdhury, Shagufta Mehnaz, and Elisa Bertino. LTEInspector: A Systematic Approach for Adversarial Testing of 4G LTE. In *Proceedings 2018 Network and Distributed System Security Symposium*, San Diego, CA, 2018. Internet Society.

[27] Syed Rafiul Hussain, Mitziu Echeverria, Omar Chowdhury, Ninghui Li, and Elisa Bertino. Privacy attacks to the 4g and 5g cellular paging protocols using side channel information. *Network and Distributed Systems Security (NDSS) Symposium*, 2019.

[28] Syed Rafiul Hussain, Mitziu Echeverria, Ankush Singla, Omar Chowdhury, and Elisa Bertino. Insecure connection bootstrapping in cellular networks: The root of all evil. In *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '19. Association for Computing Machinery, 2019.

[29] Syed Rafiul Hussain, Imtiaz Karim, Abdullah Al Ishtiaq, Omar Chowdhury, and Elisa Bertino. Noncompliance as deviant behavior: An automated black-box noncompliance checker for 4g lte cellular devices. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, New York, NY, USA, 2021. Association for Computing Machinery.

[30] Roger Piqueras Jover. LTE security, protocol exploits and location tracking experimentation with low-cost software radio. *CoRR*, abs/1607.05171, 2016.

[31] Imtiaz Karim, Syed Rafiul Hussain, and Elisa Bertino. Prochecker: An automated security and privacy analysis framework for 4g lte protocol implementations. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, pages 773–785, 2021.

[32] Keysight. SJ001A WaveJudge Wireless Analyzer Toolset. https://www.keysight.com/us/en/product/SJ001A/wavejudge-5000.html.

[33] Hongil Kim, Jiho Lee, Eunkyu Lee, and Yongdae Kim. Touching the untouchables: Dynamic security analysis of the LTE control plane. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*. IEEE, 2019.

[34] Katharina Kohls, David Rupprecht, Thorsten Holz, and Christina Pöpper. Lost traffic encryption: Fingerprinting lte/4g traffic on layer two. In *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '19, page 249–260, New York, NY, USA, 2019.

[35] Martin Kotuliak, Simon Erni, Patrick Leu, Marc Roeschlin, and Srdjan Capkun. LTrack: Stealthy tracking of mobile phones in LTE. In *31st USENIX Security Symposium (USENIX Security 22)*, Boston, MA, August 2022. USENIX Association.

[36] Swarun Kumar, Ezzeldin Hamed, Dina Katabi, and Li Erran Li. LTE Radio Analytics Made Easy and Accessible. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, 2014.

[37] Denis Foo Kune, John Kölndorfer, Nicholas Hopper, and Yongdae Kim. Location leaks over the GSM air interface. In *NDSS*, 2012.

[38] Mina Labib, Vuk Marojevic, and Jeffrey H. Reed. Analyzing and enhancing the resilience of LTE/LTE-A systems to RF spoofing. In *IEEE Conference on Standards for Communications and Networking, CSCN 2015, Tokyo, Japan, October 28-30, 2015*. IEEE, 2015.

[39] Yuanjie Li, Chunyi Peng, Zengwen Yuan, Jiayao Li, Haotian Deng, and Tao Wang. Mobileinsight: Extracting and Analyzing Cellular Network Information on Smartphones. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*, MobiCom '16, 2016.

[40] Marc Lichtman, Roger Piqueras Jover, Mina Labib, Raghunandan Rao, Vuk Marojevic, and Jeffrey H. Reed. LTE/LTE-A jamming, spoofing, and sniffing: threat assessment and mitigation. *IEEE Communications Magazine*, 2016.

[41] Norbert Ludant and Guevara Noubir. SigUnder: a stealthy 5G low power attack and defenses. *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2021.

[42] Moxie Marlinspike and Trevor Perrin. The Double Ratchet Algorithm. https://signal.org/docs/specifications/doubleratchet/doubleratchet.pdf, 2016.

[43] Moxie Marlinspike and Trevor Perrin. The X3DH Key Agreement Protocol. https://signal.org/docs/specifications/x3dh/x3dh.pdf, 2016.

[44] Karsten Nohl and Sylvain Munaut. GSM Sniffing. 27th Chaos Communication Congress, https://fahrplan.events.ccc.de/congress/2010/Fahrplan/attachments/1783\\\_101228.27C3.GSM-Sniffing.Nohl_Munaut.pdf, December 2010. Retrieved: August 18, 2022.

[45] P1 Security. QCSuper. https://github.com/P1sec/QCSuper, 2021.

[46] Xiaohong Pei. Network Signal Guru. https://www.qtrun.com/en/?page_id=34. Retrieved: June 4, 2022.

[47] David Rupprecht, Katharina Kohls, Thorsten Holz, and Christina Pöpper. Breaking LTE on layer two. In *IEEE Symposium on Security & Privacy (SP)*, 2019.

[48] David Rupprecht, Katharina Kohls, Thorsten Holz, and Christina Pöpper. Call Me Maybe: Eavesdropping Encrypted LTE Calls With ReVoLTE. In *USENIX Security Symposium (SSYM)*, 2020.

[49] David Rupprecht, Katharina Kohls, Thorsten Holz, and Christina Pöpper. IMP4GT: IMPersonation Attacks in 4G NeTworks. In *ISOC Network and Distributed System Security Symposium (NDSS)*. ISOC, February 2020.

[50] Altaf Shaik, Ravishankar Borgaonkar, N. Asokan, Valtteri Niemi, and Jean-Pierre Seifert. Practical attacks against privacy and availability in 4G/LTE mobile communication systems. In *23rd Annual Network and Distributed System Security Symposium (NDSS 2016)*, 2016.

[51] Altaf Shaik, Ravishankar Borgaonkar, Shinjo Park, and Jean-Pierre Seifert. New vulnerabilities in 4g and 5g cellular access network protocols: Exposing device capabilities. In *WiSec '19*. ACM, 2019.

[52] Haoqi Shan and Wanqiao Zhang. LTE Redirection: Forcing Targeted LTE Cellphone into Unsafe Network. In *DEF CON 24*, 2016.

[53] Software Radio Systems. SRS AirScope. https://www.srs.io/products/#SRS-airscope.

[54] SRLabs. SnoopSnitch - SRLabs Open Source Projects. https://opensource.srlabs.de/projects/snoopsnitch. Retrieved: June 6, 2022.

[55] SRS. Software Radio Systems. Open source SDR 4G/5G software suite. https://github.com/srsran/srsRAN, 2020.

[56] Daehyun Strobe. IMSI Catcher. *Seminararbeit Ruhr-Universitat Bochum*, 2007.

[57] Muhammad Taqi Raza and Songwu Lu. On Key Reinstallation Attacks over 4G/5G LTE Networks: Feasibility and Negative Impact, 2018.

[58] Robotics Online Marketing Team. 5g-powered medical robot performs remote brain surgery. *Robotics Online*, 2019.

[59] WhatsApp. WhatsApp Encryption Overview: Technical White Paper, version 6. https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf, November 2021. Retrieved: June 6, 2022.

[60] Hojoon Yang, Sangwook Bae, Mincheol Son, Hongil Kim, Song Min Kim, and Yongdae Kim. Hiding in plain signal: Physical signal overshadowing attack on LTE. In *28th USENIX Security Symposium (USENIX Security 19)*, Santa Clara, CA, 2019. USENIX Association.

[61] Chuan Yu, Shuhui Chen, Zhiping Cai, and Jesús Díaz-Verdejo. LTE Phone Number Catcher: A Practical Attack against Mobile Privacy. *Sec. and Commun. Netw.*, 2019.