Data-Plane Signaling in Cellular IoT: Attacks and Defense

Zhaowei Tan, Boyan Ding, Jinghao Zhao, Yunqi Guo, Songwu Lu
University of California, Los Angeles
{tan,dboyan,jzhao,luckiday,slu}@cs.ucla.edu

ABSTRACT

In this paper, we devise new attacks exploiting the unprotected data-plane signaling in cellular IoT networks (aka both NB-IoT and Cat-M). We show that, despite the deployed security mechanisms on both control-plane signaling and data-plane packet forwarding, novel data-plane signaling attacks are still feasible. Such attacks exhibit a variety of attack forms beyond simplistic packet-blasting, denial-of-service (DoS) threats, including location privacy breach, packet delivery loop, prolonged data delivery, throughput limiting, radio resource draining, and connection reset. Our testbed evaluation and operational network validation have confirmed the viability. We further propose a new defense solution within the 3GPP C-IoT standard framework.

CCS CONCEPTS

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

KEYWORDS

Mobile Security, Data-Plan Signaling, Security Defense, Cellular Networks, Cellular IoT, NB-IoT, Cat-M

ACM Reference Format:
Zhaowei Tan, Boyan Ding, Jinghao Zhao, Yunqi Guo, Songwu Lu. 2022. Data-Plane Signaling in Cellular IoT: Attacks and Defense. In *The 27th Annual International Conference on Mobile Computing and Networking (ACM MobiCom '21), January 31-February 4, 2022, New Orleans, LA, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3447993.3483255

1 INTRODUCTION

The 4G/5G mobile network intends to support a variety of Internet of Things (IoT) applications. The resulting cellular IoT (C-IoT) technology offers two operation modes: Narrowband IoT (NB-IoT) that provides low-rate, low-cost data connectivity, and LTE Cat-M1 (Cat-M) that enables higher-rate, mobile connectivity for machine-to-machine communications. Since both modes allow for unattended operations of IoT devices, ensuring security is thus important.

C-IoT has deployed multiple security mechanisms. Mutual authentication between the device and the network offers the first fence. After this procedure, the network and the device further encrypt and integrity-protect control-plane signaling and data-plane packets using the agreed keys. Consequently, such control-plane signaling and user packets are all secured. In contrast, data-plane signaling is not secured. It is neither ciphered nor integrity protected (§3). Data-plane signaling provides important control functions to facilitate data-plane packet transfer in C-IoT, including random access, power control, radio resource request, time alignment, reliable transfer, etc.

In this work, we study the insecurity of data-plane signaling in cellular IoT. This known vulnerability is widely believed to be not threatening. However, we combine it with other vulnerabilities in PHY/MAC/RLC protocols to launch attacks that can incur serious damages. Specifically, we address the following issues: (1) How can an attacker forge a data-plane signaling message that is decoded correctly by the receiver? It also needs to ensure that the receiver falsely thinks the forged data is from the authentic sender. (2) Is there any novel attack beyond simplistic DoS threats, which can be achieved by jamming the channel?

We design CDS, which leverages the data-plane signaling vulnerability and inflicts damages on C-IoT devices and networks (§4). CDS addresses both design issues. Firstly, CDS can forge and send a data-plane signaling message in correct time/frequency blocks (§5). It thus passes the scheduling checking at MAC. The CDS attacker also encodes and scrambles the data-plane signaling with correct configurations to be decoded successfully at PHY. Second, CDS carefully arranges the content and the procedure for data-plane signaling forgery at MAC/RLC (§6). With these forged messages, the attacker can inflict diversified attacks beyond simplistic DoS, including radio resource draining, prolonged data delivery, flexible throughput limiting, location privacy breach, packet delivery loop, and connection reset.

We implement and evaluate the attacks at our testbeds with off-the-shelf C-IoT devices (§7). We first validate that a forged data-plane signaling message can be correctly decoded and accepted for both UL and DL. Both UL and DL forgery have a high success rate of >99% when the relative power is 7dB. We also confirm the viability that CDS can arrange one or multiple forged messages to cause various threats. We validated all six proposed attacks and evaluate their damages. Moreover, we also validate several attacks over operational networks.

We further devise a lightweight solution to protecting data-plane signaling messages within the 3GPP C-IoT specification framework (§8). Our proposed solution leverages the synchronized timers between the device and the network. The sender uses the timers to derive keystream for data-plane signaling encryption and integrity protection. This time-based key derivation incurs limited modification of the current standard. The solution is efficient as it incurs few cross-layer operations. We implement our solution in the testbed. Our evaluation and security analysis indicate that the solution can secure the data-plane signaling procedure without new incurred vulnerability.

ACM ISBN 978-1-4503-8342-4/22/01.
https://doi.org/10.1145/3447993.3483255

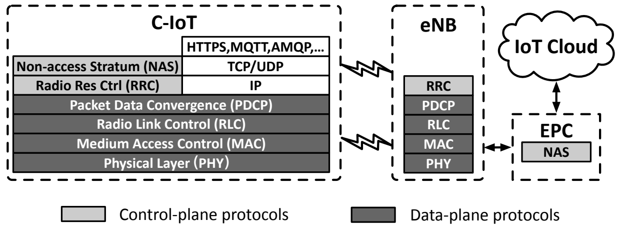


Figure 1: C-IoT architecture.

2 BACKGROUND

2.1 Cellular IoT Primer

Cellular IoT (C-IoT) provides "anytime, anywhere" wireless Internet access for IoT devices. It has defined two operation modes of Cat-M and NB-IoT through 3GPP specifications. They support low-rate, power-constrained machine-to-machine and Internet of Things applications. Cat-M operates at 1.4 MHz bandwidth. It can support data rates of 1 Mbps at maximum and voice call applications. Meanwhile, NB-IoT supports ultra-low complexity devices with extreme narrow bandwidth of 200 kHz. By the end of 2020, Cat-M and NB-IoT have been deployed by 73 operators in 73 countries and regions [26]. The current Cat-M and NB-IoT designs already meet 5G's requirements [1]. 5G RAN allows Cat-M and NB-IoT transmissions to be placed directly in a 5G frequency band [13]. 3GPP plans to rollout the standalone C-IoT in 5G from 2023 [47]. It is projected that 52% of IoT applications will run on Cat-M and NB-IoT at the end of 2025 [15].

Figure 1 illustrates the architecture for cellular IoT. Base stations (eNB in 4G or gNB in 5G[1]) provide wireless access for IoT devices. A C-IoT device exchanges data packets and control signaling with the eNB. The EPC (core network) performs signaling functions of authentication, roaming, billing, etc. It also relays data packets to the rest of the Internet.

2.2 Data Transfer in C-IoT

We present C-IoT protocol stack in Figure 1. For the control plane, NAS (Non-Access Stratum) carries control-plane messages between the device and EPC. These messages facilitate secure connection setup, mobility, etc. RRC (Radio Resource Control) is a control-plane protocol that manages the connection between the C-IoT device and the eNB. For the data plane, PDCP (Packet Data Convergence Protocol) encrypts and integrity protects the packets. RLC (Radio Link Control) provides reliable transfer and data segmentation. MAC manages radio access for both Uplink (UL) and Downlink (DL). PHY supports wireless signal processing.

Data transfer eNB manages unicast data-plane data transfer in C-IoT for both UL and DL. It sends grants to the device in the form of DCI (Downlink Control Information) through PDCCH[2] (Physical Downlink Control Channel). A grant in DCI specifies the resource blocks (RB, used in Cat-M) or resource units (RU, used in NB-IoT) that the device is authorized to transfer UL/DL data, where RB and RU are time-frequency resources. C-IoT device decodes the DCI and checks the parameters it carries. Depending on the DCI's type, the device uses the information within the DCI to either decode DL data in PDSCH (Physical Downlink Shared Channel), or to send UL data over PUSCH (Physical Uplink Shared Channel).

Data-plane signaling for data transfer C-IoT adopts data-plane signaling messages to facilitate data transfer. These data-plane signaling messages are also unicast data, transferred over PUSCH/PDSCH similar to data packets. The receiver of the signaling messages subsequently processes them based on the 3GPP standards. Data-plane signaling messages are designated to provide functions including power control, radio resource management, time alignment, etc. that are critical for successful and reliable data transfer between an eNB and a C-IoT device. The data-plane signaling messages are different from control-plane signaling messages, which manage the devices' connection or mobility state. RLC control is the data-plane signaling for reliable transfer. It has only one type, STATUS PDU [12]. MAC encodes its data-plane signaling in a special structure, named Control Element (CE). There are 14 DL CEs and 14 UL CEs in the latest release [11], with each one used for a different purpose.

2.3 Deployed Security for C-IoT

The C-IoT networks have deployed multiple security measures. We will first introduce the basic 4G/5G security practices that are inherited in C-IoT contexts. Then we further describe the additional techniques designed for C-IoT.

Common security practices in cellular networks C-IoT inherits all security protection mechanisms deployed for broadband devices (e.g., mobile phones). The core of the security measures is the mutual authentication procedure: the device and network perform mutual authentication when setting up the connection through the secure Authentication and Key Agreement (AKA) procedure. Upon a successful AKA, the device and the network agree on a few session keys for encryption and integrity protection. Control-plane messages are both encrypted and integrity-protected after mutual authentication [3]. Every data-plane packet is encrypted after the mutual authentication procedure. Future 3GPP standards will soon enforce its integrity protection [6]. Note that the cipher key and integrity protection key are updated for every packet to prevent keystream reuse. PDCP protocol is responsible for both integrity protection and encryption. In short, C-IoT enjoys the full protection for data and control signals within 4G/5G system.

C-IoT specific security measures In addition to common 4G/5G security mechanisms, C-IoT employs additional security techniques. Most notably, a C-IoT device can encapsulate its data packets in NAS messages. Therefore, the data packets will be both encrypted and integrity protected as a control-plane signaling message. A C-IoT device can specify that it enables this feature during the connection setup. The network will accept the request and notify the device if the feature is supported. This mechanism provides a quick alternative to protect C-IoT data packets before the full roll-out of the data packet integrity protection. It makes the attacker increasingly difficult to target C-IoT data-plane packets.

[1]For simplicity, we will use eNB in the rest of the paper.
[2]Cat-M uses channel mPDCCH, while NB-IoT uses nPDCCH. We use PDCCH for simplicity; the same applies to other physical channels.

3 THREAT MODEL & VULNERABILITY

3.1 Threat Model

We consider an *active* attacker seeking to break the C-IoT service. It is located in the same cell as the victim device. The adversary can eavesdrop on the radio channels and transmit forged signaling messages (including noises), while complying with the 3GPP standards. It disguises itself as a legitimate node and avoids being detected by either eNB or the victim device. The attacker can be implemented with low-cost SDR hardware and open-source 4G/5G software (shown in §7).

We assume the victim device is *directly connected to the legitimate eNB.* It successfully passes the mutual authentication procedure and acquires the keys for integrity protection and encryption. The adversary cannot compromise these security keys. The victim does not hand over to another cell, unless forced by our attacks. No false base station (FBS) is accessible to the attacker. An FBS appears as a legitimate eNB and establishes RRC connection with the C-IoT device. It can forge data to the victim device or relays data to the authentic eNB. Although FBS has been the key attack technique used by recent studies to forge data in 4G LTE [28, 37–39, 41, 42], the latest 3GPP releases for 5G have eliminated FBS by enhancing the connection setup procedure [4, 5].

3.2 Focus of this work

In this paper, we answer a simple, yet interesting question: *Does C-IoT mutual authentication procedure secure all transmissions in data and control planes?* As described in our threat model, we consider a victim that is mutually authenticated with the legitimate network. We aim at designing an attack against such targets without FBS or key leakage.

The answer seems to be positive: As we introduced in §2.3, security keys are exchanged in the mutual authentication procedure. After a victim finishes this procedure, both control and data planes encrypt and/or integrity protect the data with the keys. The attacker cannot launch any attack unless it breaks encryption or integrity protection algorithms.

Based on this common understanding, previous work did not directly answer this question. [28, 50] design attacks against a victim that is *not* mutually authenticated with the network. Some authors use FBS to directly hijack the mutual authentication procedure [38, 39]. Yet, this work focuses on the victims that are *directly* connected with the authentic network after mutual authentication.

However, our study yields a negative answer. The mutual authentication in C-IoT does not protect all data-plane data transfer. Specifically, the security keys exchanged in the mutual authentication do not encrypt or integrity protect the data-plane signaling messages. We detail this vulnerability in the next subsection, which inflicts severe damages on C-IoT devices.

3.3 Vulnerability in Data-plane Signaling

C-IoT does not protect data-plane signaling at RLC/MAC layers even after mutual authentication. Such messages are sent in *cleartext*, either as individual packets or as embedded packet headers. The examples include RLC Control (for data acknowledgment) and MAC control elements (for power control, exchanging scheduling information, etc.) If an attacker exploits them with other C-IoT vulnerabilities in PHY and MAC sublayers, it can both eavesdrop on these unencrypted messages and forge fake ones (detailed in later sections).

This design choice is an engineering trade-off as encrypting or integrity protecting data-plane signaling is nontrivial. Since current 4G/5G encryption and integrity check are enforced at PDCP, lower layers (RLC/MAC/PHY) cannot leverage such protections. Second, a keystream has to be adopted to avoid possible key reuse. PDCP uses the sequence number (SN) to generate consistent keys between the device and the eNB. In contrast, MAC does not have SN for each data-plane signaling message.

Unfortunately, such design choice greatly compromises the security. Although conventional wisdom does not anticipate much damage beyond the simplistic DoS, cleartext data-plane signaling may cause a range of severe and unexpected damages with other vulnerabilities.

Differences with other link layer vulnerabilities [38, 39] leverage the vulnerabilities in 4G/5G link layers (PDCP, RLC, MAC) to launch attacks. However, they target *data packets* and their *metadata.* Meanwhile, the core of our attack is the vulnerability of data-plane signaling messages, which complement the data packets that were studied before. To the best of our knowledge, this paper is the first work that designs attacks using data-plane signaling messages.

4 ATTACK OVERVIEW

In this section, we present our attack that leverages the vulnerability in data-plane signaling. We provide an overview in §4.1 and discuss the two major challenges in §4.2.

4.1 CDS Overview

We design and implement CDS (**C**ellular-IoT attacks with **D**ata-plane **S**ignaling), a series of cross-layer attacks that leverage the vulnerability of data-plane signaling messages. CDS leverages C-IoT-specific vulnerabilities, and it is applicable to both Cat-M and NB-IoT networks.

CDS is capable of forging a data-plane signaling message for both UL and DL. CDS encodes the data with correct coding scheme and configurations. This ensures that the forged messages could be correctly decoded at PHY. An attacker in CDS also carefully decides the proper time and radio frequency to send the forged data over-the-air. This ensures that the receiver falsely regards the forged data is from the victim (UL) or the authentic eNB (DL) at MAC.

CDS further eavesdrops on and forges one or more data-plane signaling messages to cause various damages beyond simplistic DoS. CDS determines the content in the forged messages. It also arranges the forgery with correct sequence when multiple messages are necessary to launch the attacks.

Table 1 summarizes CDS's attacks with forged data-plane signaling. They are classified into two categories. *Single-protocol attacks* aim at inflicting damage on a single protocol by forging a single data-plane signaling message. Radio resource draining allows the adversary to *persistently* drain radio resources and prevent other users from accessing the network. Prolonged packet delivery forces the victim device to turn on the sleep mode and prolong receiving

Table 1: Overview of data-plane signaling message attacks and their damages.

Category	Attack Damage	Message(s)	Impacted Protocol(s)	Direction
Single-Protocol Attacks (§6.1)	Radio resource draining	Buffer Status Report (MAC)	MAC	UL
	Prolonged packet delivery	DRX Command (MAC)	MAC	DL
	Flexible throughput limiting	Power Headroom (MAC)	MAC	UL
Cross-Layer Attacks (§6.2)	Device localization	Time Advance (MAC), RACH (MAC)	MAC & RRC	UL/DL
	Packet delivery loop	RLC Control (RLC)	RLC & MAC	UL/DL
	Connection reset	AS RAI (MAC)	MAC & RRC	UL

downlink packets. Flexible throughput limiting allows the attacker to throttle the victim device's uplink data speed at will.

CDS also includes *cross-layer attacks*, which incur damages spanning multiple layers. Device localization allows the attacker to localize the C-IoT device within a cell. Forging RLC control causes the victim device to repeatedly send the same packets at MAC and thus drain the power. Connection reset attack leverages the AS RAI message at MAC layer and terminates the RRC connection on the control plane.

Is CDS applicable to 4G/5G broadband? Although broadband and C-IoT share some common features, they bear major differences in channel design and device capability. Consequently, the attacks are not interchangeably applicable. CDS cannot be simply adapted for 4G/5G broadband for three reasons. 1) The damage of radio resource draining, throughput limiting, and packet delivery loop have limited impacts on broadband devices, as they have much more available radio resource and less stringent power requirement. 2) Some data-plane signaling messages are C-IoT-specific (AS RAI in our design). The attacks with these messages are not applicable to broadband devices. 3) Broadband DL has a different scheduling mechanism, which makes it difficult to forge data-plane signaling in time (§5.1).

Besides, previous attacks might not work for the C-IoT scenario. Attacks targeting data-plane packets (e.g., ALTER [38]) are voided as C-IoT data packets can be further integrity protected with data over NAS technique discussed in §2.3.

4.2 Attack Requirements

To launch the attacks with data-plane signaling messages, CDS addresses two critical requirements. First, an attacker is required to correctly forge data-plane signaling and convince the receiver it is from the authentic sender. The forged messages could be correctly decoded by the receiver. Second, the forged messages need to cause a wide range of damages that cannot be realized by simple attacks such as channel jamming.

Requirement 1: Forging a valid data-plane signaling message (§5) Although the data-plane signaling messages are neither encrypted nor integrity protected, the attacker must force the receiver to believe the packet is from a legitimate sender (victim or eNB). At first glance, it can be achieved with previous techniques (such as [50]). However, they focus on forging DL broadcast messages, where the timing to forge these messages can be acquired from eNB broadcast SIB messages. Forging *unicast* data-plane signaling in the *connected state* has different challenges.

We use UL forgery as an example. The attacker attempts to forge a UL data-plane signaling message which pretends to be from the victim device. For unicast messages, eNB allocates radio time-frequency blocks (RB/RU) for both UL and DL transmissions. The eNB forwards this RB/RU allocation information through PDCCH in the form of DCI. Therefore, the forged data-plane signaling message must be sent at the correct RB/RU that is assigned to the victim. Otherwise, the receiver's MAC will immediately reject the packet. The attacker cannot use the similar timing control method as for the broadcast messages; instead, it must promptly find the right RB/RU that the eNB grants the victim to send the unicast data-plane signaling.

In addition, the forged data-plane signaling needs to be correctly decoded by the receiver. This requires the attacker to encode the data with correct configurations and send the forged signaling with appropriate power. Otherwise, the receiver might fail to receive the forgery or discard the forged data if the decoding is unsuccessful.

Requirement 2: Leverage forged data-plane signaling messages (§6) A forged data-plane signaling message might cause little or no impact even if it is accepted and decoded correctly. Moreover, if the message content is out of context, it could be directly ignored or discarded by MAC/RLC. Therefore, CDS needs to properly decide the content and the sequence to forge messages to inflict serious damages.

CDS leverages the data-plane signaling forgery to launch lightweight attacks with one or multiple data-plane signaling messages. CDS achieves the salient features that cannot be achieved by simple attacks such as channel jamming.

- CDS is lightweight. It causes various damages by forging only a couple of messages. Unlike channel jamming or other approaches that need to actively send signals for a time period, CDS stays passive for most of the time. It only forges messages that occupy a few subframes.
- CDS can cause damages that are beyond simplistic DoS, such as device localization or packet delivery loop (shown in Table 1). An attacker that only jams the channel or forges the broadcast messages cannot inflict such damages.

5 FORGING DATA-PLANE SIGNALING

As discussed in §4.2, CDS has to forge a message that is accepted by the receiver. To do so, we need to answer two questions: First, *what is required for the forged data-plane signaling so that the receiver's MAC accepts the message as a valid one from an authenticated sender (§5.1)?* Second, *what is required for the message to be decoded correctly at PHY (§5.2)?*

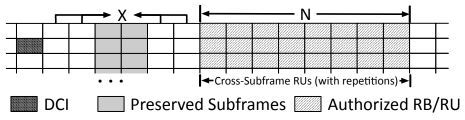


Figure 2: Cross-subframe scheduling for C-IoT.

5.1 Forge Messages with Correct RB/RU

In 4G/5G, the eNB allocates radio resources for both UL and DL transmissions, with RU and RB being time-frequency resources scheduled to a device. An attacker sending data in the unauthorized RB/RU will be immediately rejected. Therefore, CDS must acquire the scheduling information to send the forged data in the correct RB/RU. This includes three concrete steps. First, the attacker needs to synchronize with the network. It then learns the scheduling information for the victim. Finally, the attacker derives the correct RB/RU for data forgery from the scheduling information.

Time synchronization The attacker must first synchronize with the eNB to eavesdrop and forge data. Our approach resembles the one in [50] but with one difference: C-IoT adopts an extra clock HFN (Hyper Frame Number) to increase the time span for synchronization. C-IoT devices need this timer as they could be in sleep mode for a much longer time compared to a broadband device. The attacker can synchronize this HFN from eavesdropping on the SIB2 broadcast messages.

Acquire scheduling information After synchronization, the attacker must acquire the scheduling information (i.e., DCI) for forging the data at correct RB/RU. Since DCI is transmitted without encryption over-the-air, the adversary eavesdrops on the PDCCH and finds the DCI scheduled for the victim. However, in 4G broadband, the DCI and the corresponding message content are sent in the same subframe (1ms). If C-IoT adopts the same design, an attacker cannot infer these parameters in time and fabricate malicious data-plane signaling.

Another choice is to forge a DCI. However, forging a DCI with UL grant will certainly fail, as the eNB notices that the RB/RU used by the forged data is incorrect. Meanwhile, forging a DCI with DL grant is highly likely to fail. The half-duplex C-IoT device can be in DRX OFF or in the state of preparing DL/UL data, when the device cannot receive the forged DCI. In addition, the forged DCI will have incorrect NDI and trigger unwanted ACK, where a smart receiver can potentially detect the attacker.

However, C-IoT adopts cross-subframe scheduling; the timing to use a DCI is specified by its included parameters. A DCI indicates a set of RB/RU that a device can use to send UL or receive DL data. The RBs/RUs are not in the same subframe as the DCI. Instead, a delay exists between DCI and the authorized RB/RU. This is depicted in Figure 2. The delay is either a constant or can be calculated from DCI [2]. Its value is based on direction (UL or DL), spectrum usage techniques (TDD or FDD), access technologies (Cat-M or NB-IoT), and the delay field in the DCI (scheduled by the eNB). In any case, it is no less than 2ms. This leaves sufficient time for the attacker to prepare the signaling forgery.

The above design choice is not a mistake. C-IoT adopts cross-subframe scheduling for two reasons. First, IoT devices are limited in processing capacity. Handling control and data channels separately in time helps reduce processing complexity. Second, the channel bandwidth is low for C-IoT (especially NB-IoT). A single subframe is often insufficient to carry both data and control information.

Note that the attacker has to infer two more parameters for forging authorized data. First, a C-IoT DCI can grant a device with resources across multiple subframes. Due to the narrow channel in C-IoT, it is common that a data transfer exceeds the maximum RBs/RUs in a single subframe. Second, C-IoT designs repetition method [2] for coverage enhancement. Doubling the transmission results in 3 dB of coverage gain [21]. The eNB allows the device to repeatedly send the same data multiple times with a single grant. The attacker can infer both timing span and repetition number from the DCI.

Interpret valid RB/RU from DCI Given the DCI and the parameters received from the broadcast channels, the attacker derives which resource blocks and encoding scheme to use for signaling forgery. [2] mandates that a UL grant is used x UL subframes after receiving the DCI. A consecutive N RBs/RUs are assigned to the user (Figure 2).

• **UL in CAT-M** UL grant for CAT-M is transmitted as format 6-0A/6-0B DCI in mPDCCH. The behavior is standardized in [2], Section 8.0. For FDD, $x = 4$. For TDD, the attacker can infer x from the subframe number and the cell's TDD configurations. The information is available to the attacker. In any subframe-configuration combination, $x \geq 3$. An attacker needs to skip DL frames in TDD and half-duplex FDD. Since an RB is fixed in time length, N is solely dependent on the repetition number. Therefore, an attacker can derive $N = N_{Rep}$ from the DCI.

• **DL in CAT-M** DL scheduling information grant in CAT-M is transmitted as format 6-1A/6-1B DCI in mPDCCH. Regardless of FDD or TDD, $x = 2$ (Section 7.1.11 in [2]). Similar to UL, the attacker needs to skip UL frames and derive N from the repetition number.

• **UL in NB-IoT** UL grant is transmitted as format N0 DCI in nPDCCH (Section 16.6 in [2]). Since NB-IoT has a much narrower bandwidth, a device needs to wait longer for a UL/DL transmission. A DCI carries a field called scheduling delay index (I_{delay}). It is a discrete value from $\{0, 1, 2, 3\}$, and $x = 8 \cdot 2^{I_{delay}}$. The attacker needs to skip UL frames in TDD and preserved frames in FDD.

An attacker in NB-IoT calculates N as follows. It gets the parameter (I_{Rep}, I_{Sc}, I_{RU}) in the DCI [2, 10]. From I_{Sc}, it can calculate the number of slots for each RU as N^{UL}_{slots}. From I_{RU}, the attacker learns the number of consecutive RUs assigned to the victim, as N_{RU}. The data is repeated by N_{Rep} times. The total time slots can be calculated as $N = N_{RU} N^{UL}_{slots} N_{Rep}$.

• **DL in NB-IoT** DL scheduling information grant in NB-IoT is transmitted as format N1 DCI in nPDCCH. Based on Section 16.4 in [2], $x = 5 + k_0$, where k_0 can be calculated from I_{delay} in the received DCI. Similar to our discussion in UL in NB-IoT, the attacker needs to skip the preserved subframes. We next calculate N. The DCI includes I_{SF} and I_{Rep}. From the information, the attacker can infer the consecutive subframes for each assignment (N_{SF}) and repetition counts (N_{Rep}). The total consecutive frames can be calculated as $N = N_{SF} N_{Rep}$.

5.2 Forge Messages with Correct Encoding

To craft a data-plane message that could be decoded at PHY, CDS must handle the following. First, the attacker adopts the correct common configurations for data forgery. Second, the encoding needs to be correctly used for each transmission based on DCI. They ensure that the forged data can be decoded correctly. Third, the forgery must be sent with proper power to guarantee that the forged data can overshadow the legitimate transmission while not causing saturation effect.

Configurations The attacker needs to adopt the common configurations for all UL or DL signaling messages. They can be acquired from monitoring broadcast messages. Some C-IoT specific ones are dmrs-Config-r13 (to encode UL reference signal correctly), ul-ReferenceSignalNPUSCH-r13 (to get group hopping to send UL reference signal formula), etc.

Data Encoding The encoding and modulation parameters should be consistent with the assigned value in the DCI, so that the forged signaling can be correctly decoded. In DL forging, if the original eNB uses transmission diversity, the attacker needs to apply correct precoding and layer mapping. When using RU in UL, NB-IoT splits a single RB further to enable fine-grained resource allocation. CDS derives these parameters by sub-carrier spacing (available in broadcast) and the number of sub-carriers in one RU (available in DCI).

Victim C-RNTI CDS scrambles the data-plane signaling with correct C-RNTI, which is the identity assigned to each device. Due to the vulnerability that the C-RNTI is sent in cleartext, multiple approaches are available to acquire it if its IMSI [23, 29, 31, 34, 36, 41] or traffic pattern is known [31].

Power Requirement The data forging requires capture effect, in which the attacker injects a stronger signal to force the victim to decode it instead of the legitimate signal with lower power. In order to be successfully decoded at the victim, the power of injected signal needs to reach certain levels relative to the original one. For example, with NB-IoT, where QPSK is mainly used, an SNR (signal-to-noise ratio) of 6.2dB can ensure successful decoding [40].

6 DATA-PLANE SIGNALING ATTACKS

In this section, we introduce the details for CDS data-plane signaling attacks, as shown in Table 1. Their damages range from breaking the data access (such as limiting the throughput) to privacy breach (such as localizing a victim device). They leverage the CDS forgery techniques in the last section. We present two major categories: single-protocol attacks (§6.1) and cross-layer attacks (§6.2).

6.1 Single-Protocol Attacks

Single-protocol attacks aim at breaking one protocol in either victim device or network-side eNB. An attacker can launch them by forging a single message that targets the protocol to be attacked. For each attack, we will introduce the message that an attacker can leverage and the concrete steps of the attack.

6.1.1 Radio Resource Draining. In this attack, the attacker drains the UL resource in a short period, during which no device is capable of sending UL data. The attacker achieves so by forging a Buffer Status Report (BSR). This message forces the eNB to schedule excessive resource to the victim device. Given the limited bandwidth in C-IoT, other devices will be denied from sending UL data.

Buffer Status Report A C-IoT device can send a UL BSR in MAC. It includes how much UL data is waiting in the device buffer. The BSR is an index, which maps to a range of the pending data size [11]. The eNB will process it and schedule sufficient UL grants for the buffered UL data.

Attack Details The attacker forges a BSR that indicates the victim device has much UL data to be sent. It can use any positive index (e.g. 30) in the forged BSR, which indicates the pending buffer in the BSR (could be as large as MBs), as specified in the standard [11]. The eNB subsequently grants sufficient resources to this device. The attacker can repeatedly forge BSR messages to consume more radio resources.

Attack Damages The eNB will assign grants that are sufficient for the device to send all UL data. The amount of wasted resource is thus the same as the requested amount in the BSR. Studies show that an eNB tends to assign sufficient resource for a single device before serving the others [14]. Therefore, when the network assigns all resources to the target device, other devices in the same network suffer from no access to radio resource. This is especially aggravated by the fact that C-IoT networks have limited bandwidth compared to broadband networks. The attacker can forge multiple BSR messages and disable the network for seconds.

6.1.2 Prolonged Packet Delivery. The attacker can leverage forged data-plane signaling to prolong a packet's delivery, without tempering the connection or forcing a DoS on the victim device. It turns the device into Discontinuous Reception (DRX) OFF state (sleep mode) by leveraging a MAC message called DRX Command. The device cannot receive data until the sleep mode ends.

C-IoT DRX C-IoT devices adopt DRX mechanism to save energy. Specifically, an eNB configures certain "ON" and "OFF" state periodically. The eNB only sends DL data during the next ON state. Since the device expects no data during DRX OFF, it turns off data reception and thus saves energy. The eNB forwards the related parameters, such as the duration of ON period and periodicity of a cycle, to C-IoT devices.

DRX Command During DRX ON, the eNB can ask a device to terminate DRX ON early, if it expects no more DL data in this period. An eNB does so by sending a MAC control element DRX command to the device. When a device receives a DRX command, it enters the DRX OFF immediately. This further saves energy for power-constraint C-IoT devices.

Attack Details The attacker first detects DRX ON state by eavesdropping on the PDCCH and decodes DCI. The device is in the ON state if a DL grant is found. The attacker thus forges a DRX command using that DL grant and the device will falsely change its state to DRX OFF.

Attack Damage When the eNB attempts to send more DL data during this DRX ON duration, the device cannot receive the data immediately. Instead, it has to wait for the next DRX ON state. In normal C-IoT networks, the latency could be hundreds of milliseconds. If the C-IoT device enables extended DRX (eDRX), the DRX

OFF period could be even longer. This further prolongs the latency to seconds.

6.1.3 Flexible Throughput Limiting. The attacker leverages the data-plane signaling, Power headroom (PHR) messages, to limit the data throughput of the device. This attack is flexible and stealthy: Instead of persistently blocking the device's access (like the attack in §6.1.1), the throughput drop results from channel fluctuation from the victim's perspective.

Power headroom PHR is an uplink control message that indicates whether the device has extra transmission power to send UL data with higher throughput. Its value equals to device's max transmission power minus the current PUSCH Power. If the value is positive, the device has extra power for higher throughput. Otherwise, the current power consumption exceeds the device's capability. Upon receiving PHR, the eNB adjusts the UL scheduling accordingly.

Attack Details The attacker forges a PHR message to eNB. In this message, the included PHR indicates negative power headroom. The attacker uses value (0..10) as the content, which means the current PUSCH power exceeds max available transmission power [11]. This convinces the eNB that the device is suffering from high power so it needs to schedule data with smaller MCS. The eNB thus reduces scheduled UL throughput to meet the (fake) power requirements.

Note that the device might later report a PHR: when it uses the limited throughput to send data, it can notify the eNB that it has sufficient power to support higher throughput. The attacker can monitor PUSCH and observe this PHR. When the attacker detects it, the attack can be launched again.

Attack Damage The attack appears as a normal channel condition fluctuation. For the eNB, it thinks that the victim device runs out of energy. From the device's perspective, network congestion is a potential reason for the throughput drop. Therefore, neither side can detect the attack. Besides, the attacker can launch the attack anytime. The 3GPP standard allows sending PHR at any frequency. When eNB increases the throughput, it can send an additional PHR.

6.2 Advanced Cross-Layer Attacks

In this section, we detail the CDS attacks that inflict cross-layer damages on C-IoT protocols. For each attack, we introduce the involved messages and the attack procedure.

6.2.1 Device Localization. This attack targets localizing the static victim C-IoT device within the same cell it resides in. The attacker can breach the privacy of a C-IoT device. To infer the device location, we leverage the cleartext field Timing Advance (TA) in the random access procedure.

Timing Advance TA is a DL MAC control element used to synchronize the UL and DL due to propagation delay from C-IoT device to eNB. A device needs to advance the timing of its UL transmission to use the assigned RB correctly. Therefore, eNB sends TA in MAC CE during random access (RACH) or whenever any correction is required during RRC connected state. TA includes a discrete timing offset T_A. Upon receiving the offset, the C-IoT device adjusts the UL timing accordingly.

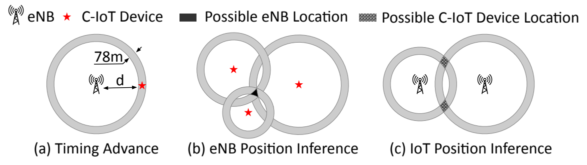

eNB
C-IoT Device
Possible eNB Location
Possible C-IoT Device Location
78m
d
(a) Timing Advance
(b) eNB Position Inference
(c) IoT Position Inference

Figure 3: Leverage TA to locate victim device/eNB.

T_A reflects the propagation delay from the eNB to the device. Since the delay is affected by the distance between the eNB and the device, it can be used to infer the location of the device. The distance $d = 3 \cdot 10^8 \cdot (T_A \cdot 8/30720000)$. Note that T_A is a positive integer from 0 to 63. Therefore, a T_A can locate the device in a circular ring. Its inner radius is d and the breadth equals to $3 \cdot 10^8 \cdot (1 \cdot 8/30720000) = 78m$. This is shown in Figure 3(a). The Non-line-of-sight (NLOS) condition might introduce a small difference between the propagation path and the real distance, but this error can rarely exceed one TA value. A device with the same distance to the eNB will be assigned with the same TA value in Line-of-sight (LOS) or NLOS settings. It will not substantially decrease the accuracy, as the device is still in the potential area inferred from TA.

Acquire the location of the eNB To locate the device, the attacker needs to get the location of the eNB so that it can use TA to infer the device's location. An attacker cannot trivially get this information over-the-air. One approach is to leverage the open cell database, such as [20]. It returns the geo-location of the eNB given a cell ID, which can be observed by the attacker. Another active approach is to inversely use TA. The attacker uses its GPS location and TA to infer the location of the eNB. This is shown in Figure 3(b). It moves around and portraits the possible location of the eNB. It then takes the intersection of the potential areas as the location of the eNB.

Infer the device's distance to multiple eNBs If the attacker knows the distances of a C-IoT device to multiple cells, the area of its potential location can be further reduced by taking the intersection. Figure 3(c) depicts the idea. Consequently, an attacker forces the C-IoT device to connect to multiple cells and eavesdrop on TA under each eNB. According to standard [9], when the UL data is not synchronized, the device will re-initiate RACH procedure. If the number of attempts exceeds the limit, it initiates RACH and RRC connection to another cell. Based on this behavior, the attacker repeatedly sends noises on the reference signals. This only needs transmission power comparable to the victim device [33]. The victim device will be forced to connect to another cell. If the attacker is in the coverage area of this new cell, it camps on the new cell for eavesdropping TA and sending noises. If the new cell is from a different physical eNB, the attacker eavesdrops on the TA value. The attacker can repeat the procedure until it cannot connect to the new cell. It gets the victim's TA in different BSes to enhance the localization accuracy.

Attack Damage The device localization attack is effective against a device in the coverage of multiple cells. It is suitable for localizing a static or quasi-stationary device. The attacker can position the victim device without compromising any internal data, such as the victim's GPS information. In addition to a specific victim, a malicious attacker can also use this technique to locate all active local devices (e.g. alarms) in a cell.

6.2.2 Packet Delivery Loop. In this attack, the attacker forces the C-IoT device to repeatedly send the same packets instead of the new ones. The C-IoT device still has access to the radio resource but transmits repeated data. This forces the energy-constraint C-IoT devices to waste a large amount of power, quickly draining their battery. The attacker achieves so by sending RLC control messages.

RLC Control RLC Control is used by RLC protocol to ensure reliable transfer. In RLC AM (Acknowledged Mode, the most common mode used in C-IoT), C-IoT device and eNB will initiate RLC control messages to notify the other side whether the data reception is successful. This is triggered when a timer expires or the other side sends a polling bit. An ACK in RLC Control specifies the sequence numbers (SN) of the packets successfully transmitted. When some packet SNs are missing, RLC control includes an NACK that explicitly specifies the corrupted data. Note that RLC control is different from MAC ACK/NACK which is an unreliable indicator for quick recovery (<10ms), while RLC control is used to ensure reliable transfer. The receiver of an RLC control checks the content and acts accordingly. If an ACK is present, the receiver will clear the specified data in the buffer. If an NACK is present, the receiver of the NACK will read the content within the message and initiate the retransmission. The corrupted data packets will be forwarded to MAC.

Attack Details The attacker first eavesdrops on the RLC control exchange over-the-air by eavesdropping on PDCCH for DCI and decoding data channels. This allows the attacker to get the highest acknowledged SN (SN_m). It subsequently forges an NACK with a sent but unacknowledged SN (e.g., $SN_m + 2$). Upon reception of this message, eNB/device retransmits the packets with the SN included in the forged RLC Control.

Attack Damage The RLC retransmission in MAC has a *higher priority* compared to the normal data transmission. The retransmission thus blocks the new data. Moreover, the attacker can repeatedly forge this message and cause the loop. The device thus loses UL access, while still logically connected to the eNB. Meanwhile, the constant retransmissions keep the device in DRX ON, quickly draining its limited power. In the extreme case where the attacker constantly forges messages, the device never enters DRX OFF.

6.2.3 Connection Reset. The attacker can reset the RRC connection by forging an AS RAI. This turns the device to IDLE state and might terminate any stateful connection (e.g. TCP).

Access Stratum Release Assistance Indication (AS RAI) AS RAI is a C-IoT-specific feature introduced in Release 16 [11]. A C-IoT device uses this message to notify the eNB whether it expects any subsequent data transfer with an indicator. The purpose is to save energy when the device has no more data transmission.

Attack Details The attacker forges an AS RAI control element with AS RAI value as 01. The attacker embeds an RAI in a UL MAC control element called DCQR and AS RAI. In the message, the DCQR part can be any arbitrary content. The eNB falsely regards that no following data is expected from the C-IoT device.

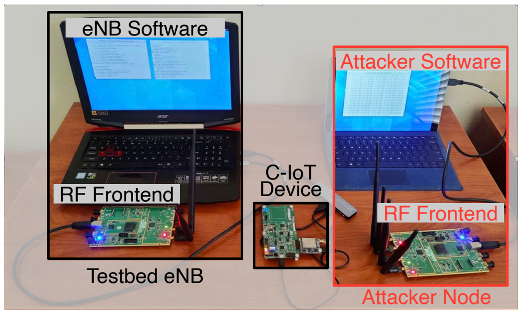


Figure 4: The testbed for attack validation.

Attack Damages The eNB falsely regards that no data is expected from the C-IoT device. Although AS RAI is a MAC layer message, it impacts RRC connection between the victim device and the eNB. A forged AS RAI indicating no data terminates the RRC connection with the victim device. After the attack, the victim might initiate a re-connection. The attacker can monitor the channel and repeatedly forge AS RAI attacks for consistent connection reset.

7 ATTACK VALIDATION

7.1 Experimental Setup

Testbed Our testbed is shown in Figure 4. A Surface Pro 3 with Intel i7-4650U processor and 8G RAM runs the attacker software. It implements the attack logic and controls the forged signaling on Ubuntu 20.04. The attacker software connects to a USRP B210 as its RF frontend.

The IoT device is an STM32L496 LTE IoT Cellular-to-Cloud Discovery Pack [44]. To connect to an operational network, it uses a SIM card from operator X and registers on X's NB-IoT networks. To connect to our testbed network, we insert a sysmoUSIM SIM card [45] and register its info on our server.

The attacker node targets both operational and testbed NB-IoT. We do not have access to a Cat-M testbed, but the implementation could be adapted with similar components. For DL attacks, since the damage will only inflict damages on the victim, we validate the attacks with an operational network. As some UL attacks affect operational networks and users, we customize a private NB-IoT network. To set up an NB-IoT eNB, an Acer laptop with i7-7700HQ CPU and 16G RAM runs OpenAirInterface [35] for eNB processing logic. It is connected to another USRP B210 for sending and receiving wireless signals.

Attacker's Location Due to the limitation of the SDR-based attacker and testbed, we place the attacker node close to the victim to ensure sufficient relative power. We test our data forgery in §7.3 with two different locations. In location A, the attacker is 5 meters away from the victim in the same room. In location B, the attacker is separated by a wall in a different room from the victim. They are 15m away from each other. The testbed is placed in an indoor building on the third floor with NLOS condition. The settings are adapted from the previous work as in [50]. The location used for each data-plane signaling attack is introduced in §7.4.

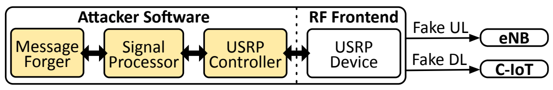


Figure 5: The implementation of the attacker node.

Power Requirement To evaluate power requirement for successful data forgery, we adjust the USRP's transmission power at the attacker with different levels of relative power. We have tested with relative power levels of 3dB, 5dB, and 7dB.

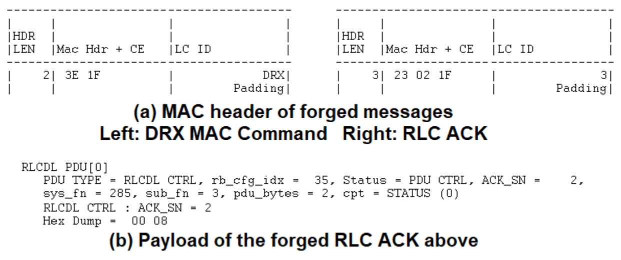
Ethical Considerations Our attacker node is carefully managed during experiments. The DL signals only reach a few meters; we ensure no device other than our victim NB-IoT board is affected. We validate the UL attacks at our testbed, without affecting the operational networks. The attacker node cannot access any other C-IoT device. Mobile operators are also notified of our findings.

7.2 Attacker Node Implementation

The overall implementation is presented in Figure 5. USRP acts as an RF frontend. The software part contains three components: a USRP Controller, a Signal Processor, and a Message Forger. The USRP Controller communicates with the USRP. It provides interfaces for the Signal Processor to send and receive wireless C-IoT signals. Meanwhile, it does correction on signal according to hardware (detailed below). The Signal Processor carries out PHY processing, decoding messages from the eavesdropped signal while encoding forged messages into wireless signals. We also allow the Signal Processor to send noises to corrupt a specific channel. The Message Forger implements a simplified NB-IoT protocol stack (Up to RLC). It receives demodulated data from Signal Processor and crafts the fake signaling. To forge a signaling message, it passes the data to the Signal Processor.

USRP Controller The USRP Controller uses the API from the RF frontend to assist Signal Processor in eavesdropping on and sending wireless signals. Besides, it ensures correct timing and frequency synchronizations. For timing, USRP Controller aligns itself with the synchronization signals (NPSS and NSSS) from eNB. For frequency, it tackles two critical roadblocks: *Carrier frequency offset* (CFO) and *Sampling frequency offset* (SFO). The frequency offsets can cause failure in attack if not properly handled. While a GPS-Disciplined Oscillator can be applied to minimize the difference, we adopt a more cost-effective approach. The USRP Controller estimates the CFO and SFO between itself and the eNB with synchronization signals. Then it carries out the correction on transmitting (after it gets signals from the Signal Processor) and receiving (before passing signals to the Signal Processor). For CFO, the attacker applies a phase rotation on the signals. While for SFO, the attacker inserts or removes samples to compensate when the difference in sampling accumulates to a certain degree.

Signal Processor The Signal Processor is mainly responsible for PHY layer processing. Our implementation extends the signal processing logic from srsLTE [25]. Concretely, we realize the modulation of NB-IoT data (in both UL and DL) and noise signals. The processing follows 3GPP standards [2, 7] and supports different modulation schemes. It decodes relevant messages of interest (e.g. DCI or data-plane signaling messages intended for the victim)



```
---------------------------------------        ---------------------------------------
|      |                |              |       |      |               |               |
|HDR   |                |              |       |HDR   |               |               |
|LEN   |Mac Hdr + CE    |LC ID         |       |LEN   |Mac Hdr + CE   |LC ID          |
---------------------------------------        ---------------------------------------
|     2| 3E 1F          |           DRX|       |     3| 23 02 1F      |              3|
|      |                |       Padding|       |      |               |        Padding|
```

(a) MAC header of forged messages
Left: DRX MAC Command Right: RLC ACK

```
RLCDL PDU[0]
   PDU TYPE = RLCDL CTRL, rb_cfg_idx =  35, Status = PDU CTRL, ACK_SN =    2,
   sys_fn = 285, sub_fn = 3, pdu_bytes = 2, cpt = STATUS (0)
   RLCDL CTRL : ACK_SN = 2
   Hex Dump =  00 08
```

(b) Payload of the forged RLC ACK above

Figure 6: MobileInsight log of forged messages.

and forwards them to Message Forger. When the Message Forger generates the attack traffic (forged signaling or noise) the Signal Processor generates the corresponding signals according to the request.

Message Forger Based on the decoding results from the Signal Processor, the Message Forger crafts fake data-plane signaling messages. The victim will receive the forged message instead of the original authentic one because of the stronger signal strength of the attacker. The fake messages follow MAC/RLC standards [11, 12]. They appear to be legitimate messages but are specially designed to inflict certain damages on the device or network. Forged messages are then forwarded messages to Signal Processor for modulation.

7.3 Validate Data-plane Signaling Forgery

We first validate a fake message can be successfully decoded and processed for both UL and DL (§5). For DL, the attacker forges a MAC CE message and an RLC message. We demonstrate the victim device can receive both. The content of the forged MAC CE is 3E 1F 00 00 00, where 3E indicates a DRX command and 1F 00 00 00 are 4-byte paddings. On the NB-IoT board, we use the NB-IoT support in MobileInsight [32] to collect fine-grained cellular logs. A MobileInsight log only includes messages that are *correctly decoded and processed.* As shown in Figure 6(a), the victim receives and accepts the forged MAC CE. The victim device correctly recognizes the DRX command in the forged message. We also test with message 23 02 1F 00 08, where 23 02 means there is a 2-byte RLC message, 1F is the padding, and 08 is the RLC control. As shown in Figure 6(b), the victim device accepts the RLC control message. It successfully recognizes the message 08 as an RLC ACK for packet SN 02. Similarly for UL, we send two messages to verify UL forge data: 3D 23 02 1F 00 00 (uplink BSR with padding) and 23 02 1F 00 08 for RLC control. In the eNB logs, both messages are decoded correctly.

Table 2: Success rate of data forging.

Relative Power	3dB	5dB	7dB	Location A	Location B
DL	40.3%	75.8%	99.9%	99.1%	92.7%
UL	41.2%	70.3%	99.8%	96.3%	94.1%

Message forging success rate We measure the success rate that the forged signaling is accepted. We present the obtained success rates in operational networks for DL forgery and at the testbed for UL forgery. For each setting, we forge 1000 data-plane signaling messages and count them in MobileInsight logs. The results are shown in Table 2. The legitimate data are fully blocked in all power levels greater than 3dB. Meanwhile, the success rate of decoding forged data is 40.3% at 3dB, 75.8% at 5 dB, and 99.9% at 7 dB for DL and 41.2% at 3dB, 70.3% at 5 dB, and 99.8% at 7 dB for UL. For location-based testing, UL forgery has a high success rate with 96.3% in location A and 94.1% in location B. For DL, 99.1% and 92.7% of the signaling messages can be successfully decoded for location A and B, respectively.

7.4 Validate Data-plane Signaling Attacks

Next, we evaluate the damage and impact for each attack introduced in §6. The radio resource draining is validated in the testbed environment to avoid disrupting other devices in the cell. The flexible throughput limiting and connection reset attacks are also validated in our testbed because we cannot otherwise confirm the attack damages. The other attacks are validated in the operational network X. We place the attacker 5 meters away from the victim for each attack. For the device localization attack, the device is placed at three different, LOS outdoor locations. For all other attacks, the device is placed at an NLOS location in an indoor building on the third floor. Since we have validated the possibility of forging a data-plane signaling message with other settings, the attacks are also applicable in those scenarios.

Validate radio resource draining attack We demonstrate that a forged BSR can drain the radio resources. The fake BSR has a value of 31, which indicates 1KB UL data is pending. We measure the percentage of RU assigned to the fake BSR, divided by the total RU. Figure 7(a) illustrates how this ratio changes over time in response to a forged message. When the attack starts, the eNB schedules for this forged BSR for 200ms. During this period, the attacker completely occupies the channel for 32ms and uses 50% of the channel for 18ms. We also test how the eNB responds to two forged messages, shown in Figure 7(b). Each BSR claims there is 1000B pending data. The attacker completely occupies the channel for 64ms during the next 180ms.

The forged BSR messages drain the resource for the other devices in the same cell. For a time period, the attacker occupies 100% UL resources. The damage could be even more severe if the attacker uses a larger BSR index or initiate frequent BSR messages. The attack applies to operational networks, as the eNBs that follow 3GPP standards should respond to BSR with sufficient grant. 4G Broadband can easily transfer 1KB UL data in one subframe [46], however, it requires NB-IoT to consume most resources over hundreds of milliseconds.

Validate prolonged packet delivery attack We present an attack trace that demonstrates a DRX command can delay the DL data reception. As shown in Figure 7(c), the DRX ON state is supposed to finish at time 4540ms. However, a DRX command signaling message prematurely turns the device into DRX OFF state at time 4520ms. At time 4530ms, the eNB sends a DL packet. Since the victim is in the sleep mode (DRX OFF), it fails to receive the packet. The transmission cannot go through until the next DRX ON period at 4820ms. The latency is prolonged by more than 300 milliseconds in this experiment. This extra latency caused by the attacker is dominated by the periodicity of a DRX cycle time. In this operational network, the periodicity is 320ms. This value could be seconds in some operational networks.

Table 3: Evaluation of Device Localization. Inferred area and error are average numbers in an area.

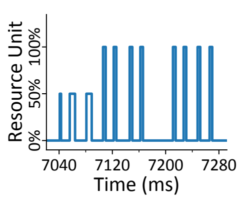
Area	Locations	# cells	Inferred area (m^2)	Error (m)
1	15	3	4475	119.4
2	10	3	9775	197.6
3	12	3	6900	99.1

Validate flexible throughput limiting attack We first validate the PHR attack messages are successfully accepted and decoded. Since existing codes for handling PHR in NB-IoT is incomplete, we use its broadband processing logic to derive the damage. When a PHR is received, the eNB adjusts the MCS for UL transfer. We draw the relationship between the MCS assignment and PHR in Figure 7(d). Assume the eNB assigns 3 resource units to the victim device, we also plot how PHR impacts the UL throughput. A forged PHR can reduce the throughput by 3.27x.

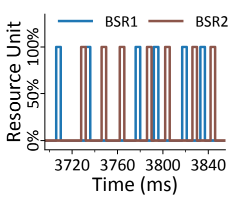
Validate device localization attack In this experiment, we demonstrate: 1) the attacker can locate the victim C-IoT device with TA, and 2) the inference is accurate as the real position of the device is close to the inferred area. We test three different areas in City A. The attacker first finds 3 local eNBs. It then forces the device to connect to them and eavesdrops on TA. The attacker succeeds in each attack within a minute. We record the user location with GPS as ground truth and compare it with the attack results, shown in Table 3. In each area, we localize the device as it moves to at least 10 different locations. The average area by inference is 4475-9775 m^2, and the device is 100-200m away from that result on average. To gauge the error, we sample the points in the inferred area and calculate the distance between the sampled spots and the device. The error stems from large TA coverage area, inaccurate TA incurred by interference, and imprecise eNB location, etc. The localization is less accurate compared to positioning techniques used for emergency calls in [16, 24] (error of 50-100m). However, the attacker achieves so without actually connecting to or controlling the device.

Validate packet delivery loop We forge RLC control to force the device to repeatedly send the already accepted data. We show a trace from our experiment in Figure 7(e). At time 4740ms, the device sends packets with SN 62. Packets with SN 60-62 are not acknowledged. The attacker learns the SN and forges an RLC control with NACK at time 4754ms. The forged control message indicates packets 60-62 are not received at the eNB. The device thus retransmits them upon receiving the RLC control. The attacker repeats this message at time 4891ms. The packet sending forms a loop. Starting from 4500ms, the C-IoT device is kept in DRX ON period. The power-constraint C-IoT device thus consumes excessive energy for >500ms. The attacker can repeat the message to persistently drain the device's energy.

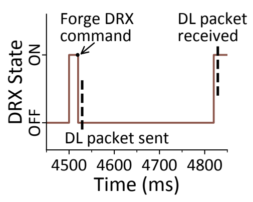
Validate connection reset attack We partially validate the attack. We first validate the eNB successfully decodes a forged RAI and the content is compliant with the standard. Unfortunately, we cannot directly observe the connection reset effect in our experiments, as both the operational network and our testbed run on release 14 and do not support release 16 message AS RAI. However,



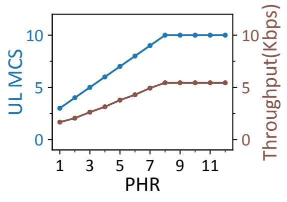
(a) The forged BSR wastes UL resource.



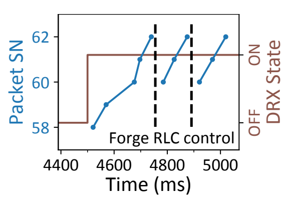
(b) Two forged BSRs drain UL resource.



(c) DRX command prolongs packet latency.



(d) PHR affects MCS and throughput.



(e) Forged RLC controls cause packet delivery loop.

Figure 7: Impact quantification of data-plane signaling attacks through experimental validation.

a standard-compliant eNB should terminate the connection based on the attack message.

8 DEFENSE

We present a novel time-based scheme that ciphers and integrity protects data-plane signaling for C-IoT. We first review the issues to address for protecting data-plane signaling. We then elaborate on our suggested design to address the issues.

8.1 Design Issue and Solution Space

Encrypting and integrity protecting data-plane signaling is a straightforward solution to protect against the proposed attacks. However, C-IoT key hierarchy terminates at the PDCP protocol. Any lower-layer protocol (i.e., RLC and MAC) is unaware of the security keys established from the mutual authentication procedure. Moving data-plane signaling messages to PDCP/RRC for security is not a good option. This will incur unacceptable cross-layer overhead, since such data-plane signaling messages require prompt actions in their own protocols. This is inappropriate for power-constrained C-IoT devices. Therefore, MAC/RLC need to derive new keys to protect data-plane signaling.

In addition, the device and eNB must update the keys for every message to prevent keystream reuse. If the constant keys are used, this can allow an attacker to launch bit flipping attack [38]. One naive option for key update is to derive keystream based on sequence number (SN), similar to what PDCP does for data-plane packets. However, data-plane signaling messages do not necessarily have SN. Adding extra sequence number vastly changes the current RLC/MAC design and increases overhead for transmission.

We thus have three requirements for securing data-plane signaling: 1) The keys must differ from those used in PDCP; 2) The generation of keystream must use parameters other than PDCP SN number; and 3) The processing must be efficient for C-IoT devices.

8.2 Proposed Solution

We propose an energy-efficient solution for protecting data-plane signaling messages in C-IoT. It is efficient as it can be performed at the MAC layer with limited cross-layer operations. The solution is also readily available as it extends the existing security mechanism in C-IoT. In this mechanism, MAC encrypts and integrity protects each MAC control element and RLC control. It generates stream keys for each data-plane signaling to ensure no key-reuse.

$$\begin{tikzcd}[column sep=small] & & K_{eNB} \arrow[dll] \arrow[dl] \arrow[d] \arrow[dr] \arrow[drr] & & \\ K_{UPenc} & K_{RRCenc} & K_{RRCint} & K_{DSenc} & K_{DSint} \end{tikzcd}$$

Figure 8: New security keys for data-plane signaling.

First, protecting data-plane signaling needs separate keys (Requirement 1). In the current 4G/5G key hierarchy, a K_{eNB} is established during RRC connection setup. Both the device and the eNB use this key to derive K_{UPenc} to encrypt data packets and K_{RRCenc}/K_{RRCint} to protect control plane messages. We generate two new keys, K_{DSenc} and K_{DSint}, as shown in Figure 8. They are derived from the existing key hierarchy using the same one-way function SHA256 but with different parameters. The keys are exclusively used for protecting data-plane signaling messages at RLC and MAC.

To satisfy Requirement 2 and 3, we propose a novel time-based stream key generation at MAC. We take subframe (SFN), frame (FN), and hyperframe (HFN) numbers as parameters. They are readily available in MAC without cross-layer operations. Both eNB and C-IoT device have consistent time frames after connection setup. This saves overhead for exchanging the parameters. With the encryption key K_{DSenc}, we encrypt each data-plane signaling message with a function $F(\text{ds}, K_{DSenc}, \text{SFN}, \text{FN}, \text{HFN}, \text{dirc}, \text{len})$, where ds is the data-plane signaling message, len is its size, and dirc is the direction of the message (DL/UL). F is an EEA encryption algorithm. Operational networks have adopted and improved EEA algorithms over years. We re-use them for fast roll-out, only with slight adaptation based on the different numbers of inputs as counters. Similarly, we integrity protect the data-plane signaling messages with K_{DSint} and the same set of parameters.

We address a potential key-reuse issue: As SFN, FN, and HFN are clock arm ticking values in 4G/5G, the combination of (SFN, FN, HFN) will return to the same value every 2.91 hours. Therefore, we further suggest the network resets the keys if an RRC connection lasts longer than this.

Security Analysis First, our solution prevents the attacker from eavesdropping on the data-plane signaling. The messages are encrypted with stream cipher keys. The attacker cannot infer the data without access to K_{DSenc} [8]. Second, forging data-plane signaling is prevented by integrity protection. An attacker cannot calculate the correct Message Authentication Code without K_{DSint}. Hence, any attack that forges data-plane signaling will fail.

```
//PDCP
   Keys = RRCConnectionSetup()
   DS.keySetup(Keys)
//MAC send
   ds = generateDataSignaling()
   ds = DS.protect(ds, SFN, FN,HFN, dirc, len)
//MAC receive
   ds = getDSfromPHY()
   ds = DS.process(ds, SFN, FN,HFN, dirc, len)
```

Figure 9: Solution pseudocode.

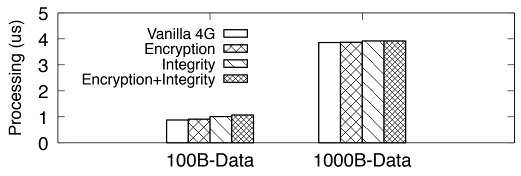


Figure 10: Processing overhead.

8.3 Prototype and Evaluation

Prototyping solution We implement our solutions in a C library. When RRC connection is established, both sides call the library to set up K_{DSint} and K_{DSenc}. We use EEA2 for encryption and EIA2 for integrity protection. To send an RLC control or MAC CE, the sending side MAC makes an API call protect(ds, SFN, FN, HFN, dirc, len) provided by the library. The library generates the stream cipher and integrity keys, which are used to protect the messages. The receiving side calls a library function to verify the integrity and decrypt the packet. The pseudocode is shown in Figure 9.

Evaluation We first validate the correctness of the encryption and decryption: The decryption and integrity check achieve a 100% success rate, regardless of retransmission, channel condition, or message length. We then evaluate the overhead of our solution. We compare the processing overhead and extra data with the legacy C-IoT. The defense scheme can thus protect against all attacks proposed in this paper.

For processing overhead, we compare the time for processing the data packets with and without our data-plane signaling message protection mechanism. For extra data overhead, we use the data-plane signaling message size of 10B, which is the max message that we use in our attacks. They are sent with two types of data packet sizes, 100B and 1000B.

We show the results of the overhead from our experiments in Figure 10. We measure the processing overhead for data packets and data-plane signaling messages. We calculate the amortized overhead with the frequency of data-plane signaling from real traces, and the overhead from encryption is 0.4-3.6%. Similarly, the amortized processing overhead for applying integrity protection is 14.8% and 1.6% for two data packet sizes, respectively. Moreover, encrypting data-plane signaling does not incur extra data overhead, while each data-plane signaling incurs 4B extra data when integrity protected. Hence, both processing and data overheads for our proposed method are small.

9 RELATED WORK AND DISCUSSION

Although operational operators have deployed C-IoT networks for years, their security is little studied. Existing work unveils the vulnerability of C-IoT in OS [22], IoT cloud [30], authentication [17, 18, 48], etc. Our work takes a fresh perspective from the unique features of C-IoT in the data plane. Instead of security, researchers mainly focus on C-IoT's other properties, such as energy [49], efficiency [19, 27], etc. The assumptions are made that C-IoT networks are secure as they inherit security measures from 4G/5G broadband with enhancements. However, we invalidate them in this paper.

Research on general security of cellular networks (4G/5G) has become an active area in recent years. Authors in [37–39, 43] propose several attacks to manipulate data packets and impersonate the victim. However, these attacks rely on the use of false base stations (FBS), while 5G targets to eliminate FBS [4, 5]. In contrast, our proposed attacks do not require FBS, as the attacker node can forge data-plane signaling messages without connecting with eNB or victim C-IoT device. [28, 41, 42, 50] exploit unprotected control-plane signaling messages. All these works target data in 4G/5G broadband before mutual authentication. To the best of our knowledge, we conduct the first work to study C-IoT vulnerabilities and data-plane signaling after mutual authentication.

10 CONCLUSION

In this paper, we study a relatively unexplored security topic on cellular IoT: data-plane signaling induced attacks. We show that, despite all existing security mechanisms on both control-plane signaling messages and data packet forwarding inherited from the legacy 4G/5G networks, new attacks exploiting the unprotected data-plane signaling are still feasible in C-IoT networks. Our attack, CDS, can forge data-plane signaling messages that are successfully decoded and accepted by the receiver at both UL and DL. This is achieved by leveraging the cleartext data-plane signaling and other vulnerabilities in RLC/MAC/PHY protocols. Moreover, an attacker can leverage the forged data to inflict damages way beyond the commonsense denial-of-service related threats. Adversaries may breach location privacy, limit the C-IoT device throughput, create local forwarding loops, reset the C-IoT connection at will, prolong the packet delivery, and drain the C-IoT radio resources. Such attacks will expose unattended cellular IoT devices to bigger damages and risks.

We implement the data-plane signaling forgery and the attacks that leverage these forged data-plane signaling messages. Our testbed evaluation and validation over operational networks have confirmed the feasibility of such new attacks. To protect the data-plane signaling messages with low overhead, we propose a time-based defense solution at MAC. It leverages the synchronized timers that are readily available at MAC to derive keystream for integrity protection and encryption. We implement the solution and our evaluation confirms its effectiveness and cost-efficiency.

ACKNOWLEDGMENTS

We greatly appreciate the anonymous reviewers for their constructive comments during the review process. The advice has helped improved the quality of this work substantially. We also thank the members of UCLA WiNG for their early feedback. This work is partly supported by NSF CNS-1910150 and NSF CNS-2008026.

REFERENCES

[1] 3GPP. Proposed way forward on IMT-2020 submission, Sep. 2017.
[2] 3GPP. TS36.213: Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures, Sep. 2019.
[3] 3GPP. TS33.401: 3GPP System Architecture Evolution (SAE); Security architecture, Jul. 2020.
[4] 3GPP. TS33.501: Security architecture and procedures for 5G System, Dec. 2020.
[5] 3GPP. TS33.809: Study on 5G security enhancements against False Base Stations (FBS), Dec. 2020.
[6] 3GPP. TS33.853: Key issues and potential solutions for integrity protection of the User Plane (UP), Dec. 2020.
[7] 3GPP. TS36.211: Evolved Universal Terrestrial Radio Access (E-UTRA); Physical channels and modulation, Sep. 2020.
[8] 3GPP. TS33.401:3GPP System Architecture Evolution (SAE); Security architecture, Sep. 2021.
[9] 3GPP. TS36.201: Evolved Universal Terrestrial Radio Access (E-UTRA); LTE physical layer; General description , Jan. 2021.
[10] 3GPP. TS36.212: Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding, Jan. 2021.
[11] 3GPP. TS36.321: Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC) protocol specification, Jan. 2021.
[12] 3GPP. TS36.322: Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Link Control (RLC) protocol specification, Jan. 2021.
[13] 3GPP. TS38.211: NR; Physical channels and modulation, Jan. 2021.
[14] Balasingam, A., Bansal, M., Misra, R., Nagaraj, K., Tandra, R., Katti, S., and Schulman, A. Detecting if lte is the bottleneck with bursttracker. In *The 25th Annual International Conference on Mobile Computing and Networking* (2019), pp. 1–15.
[15] Blackman, J. A 2019 surge, a 2020 wobble, a 2025 boom - nb-iot and lte-m on track, says ericsson. https://enterpriseiotinsights.com/20200616/channels/news/nb-iot-and-lte-m-on-track-says-ericsson, Jun 2020.
[16] Bressner, T. A. H. Development and Evaluation of UTDoA as a Positioning Method in LTE. https://www.diva-portal.org/smash/get/diva2:867797/FULLTEXT01.pdf, Jun 2015.
[17] Cao, J., Yu, P., Ma, M., and Gao, W. Fast authentication and data transfer scheme for massive nb-iot devices in 3gpp 5g network. *IEEE Internet of Things Journal 6*, 2 (2018), 1561–1575.
[18] Cao, J., Yu, P., Xiang, X., Ma, M., and Li, H. Anti-quantum fast authentication and data transmission scheme for massive devices in 5g nb-iot system. *IEEE Internet of Things Journal 6*, 6 (2019), 9794–9805.
[19] Cavo, L., Fuhrmann, S., and Liu, L. Design of an area efficient crypto processor for 3gpp-lte nb-iot devices. *Microprocessors and Microsystems 72* (2020), 102899.
[20] CellMapper. CellMapper. cellmapper.net, Mar 2021.
[21] Chakrapani, A. Nb-iot uplink receiver design and performance study. *IEEE Internet of Things Journal 7*, 3 (2019), 2469–2482.
[22] Coman, F. L., Malarski, K. M., Petersen, M. N., and Ruepp, S. Security issues in internet of things: Vulnerability analysis of lorawan, sigfox and nb-iot. In *2019 Global IoT Summit (GIoTS)* (2019), IEEE, pp. 1–6.
[23] Dabrowski, A., Petzl, G., and Weippl, E. R. The messenger shoots back: Network operator based IMSI catcher detection. In *International Symposium on Research in Attacks, Intrusions, and Defenses* (2016), Springer, pp. 279–302.
[24] Fischer, S. Observed Time Difference Of Arrival (OTDOA) Positioning in 3GPP LTE. https://www.qualcomm.com/media/documents/files/otdoa-positioning-in-3gpp-lte.pdf, Jun 2014.
[25] Gomez-Miguelez, I., Garcia-Saavedra, A., Sutton, P. D., Serrano, P., Cano, C., and Leith, D. J. srsLTE: An open-source platform for LTE evolution and experimentation. In *Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization* (2016), pp. 25–32.
[26] GSMA. Mobile iot lpwa - lte-m & nb-iot commercial launches: Gsma. https://www.gsma.com/iot/mobile-iot-commercial-launches, Mar 2021.
[27] Harwahyu, R., Cheng, R.-G., Wei, C.-H., and Sari, R. F. Optimization of random access channel in nb-iot. *IEEE Internet of Things Journal 5*, 1 (2017), 391–402.
[28] Hussain, S., Chowdhury, O., Mehnaz, S., and Bertino, E. LTEInspector: A systematic approach for adversarial testing of 4G LTE. In *Network and Distributed Systems Security (NDSS) Symposium 2018* (2018).
[29] Hussain, S. R., Echeverria, M., Chowdhury, O., Li, N., and Bertino, E. Privacy attacks to the 4g and 5g cellular paging protocols using side channel information. In *NDSS* (2019), vol. 19, pp. 24–27.
[30] Jia, Y., Xing, L., Mao, Y., Zhao, D., Wang, X., Zhao, S., and Zhang, Y. Burglars' iot paradise: Understanding and mitigating security risks of general messaging protocols on iot clouds. In *IEEE S&P* (2020), IEEE, pp. 465–481.
[31] Kohls, K., Rupprecht, D., Holz, T., and Pöpper, C. 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* (2019), pp. 249–260.
[32] Li, Y., Peng, C., Yuan, Z., Li, J., Deng, H., and Wang, T. Mobileinsight: Extracting and analyzing cellular network information on smartphones. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking* (2016), pp. 202–215.
[33] Lichtman, M., Jover, R. P., Labib, M., Rao, R., Marojevic, V., and Reed, J. H. Lte/lte-a jamming, spoofing, and sniffing: threat assessment and mitigation. *IEEE Communications Magazine 54*, 4 (2016), 54–61.
[34] Mjølsnes, S. F., and Olimid, R. F. Easy 4G/LTE IMSI catchers for non-programmers. In *International Conference on Mathematical Methods, Models, and Architectures for Computer Network Security* (2017), Springer, pp. 235–246.
[35] OpenAirInterface. OpenAirInterface Official Website, April 2018. http://www.openairinterface.org.
[36] Positive Technologies. Diameter Vulnerabilities Exposure Report, 2018.
[37] Rupprecht, D., Jansen, K., and Pöpper, C. Putting LTE security functions to the test: A framework to evaluate implementation correctness. In *10th USENIX Workshop on Offensive Technologies (WOOT 16)* (2016).
[38] Rupprecht, D., Kohls, K., Holz, T., and Pöpper, C. Breaking LTE on layer two. In *2019 IEEE Symposium on Security and Privacy (SP)* (2019), IEEE, pp. 1121–1136.
[39] Rupprecht, D., Kohls, K., Holz, T., and Pöpper, C. IMP4GT: Impersonation attacks in 4G networks. In *Symposium on Network and Distributed System Security (NDSS). ISOC* (2020).
[40] Schwarz, R. . LTE: System Specifications and Their Impact on RF & Base Band Circuits. https://cdn.rohde-schwarz.com/pws/dl_downloads/dl_application/application_notes/1ma221/1MA221_0e.pdf, Apr 2013.
[41] Shaik, A., Borgaonkar, R., Asokan, N., Niemi, V., and Seifert, J.-P. Practical attacks against privacy and availability in 4G/LTE mobile communication systems. *NDSS* (2015).
[42] Shaik, A., Borgaonkar, R., Park, S., and Seifert, J.-P. On the impact of rogue base stations in 4G/LTE self organizing networks. In *Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks* (2018), pp. 75–86.
[43] Shaik, A., Borgaonkar, R., Park, S., and Seifert, J.-P. New vulnerabilities in 4g and 5g cellular access network protocols: exposing device capabilities. In *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks* (2019), pp. 221–231.
[44] STMicroelectronics. LTE Cellular to Cloud Pack with STM32L496AG MCU. https://www.st.com/en/evaluation-tools/p-l496g-cell02.html, Jan 2021.
[45] sysmocom. sysmoUSIM-SJS1 SIM + USIM Card (10-pack). http://shop.sysmocom.de/products/sysmousim-sjs1, 2016.
[46] Tan, Z., Zhao, J., Li, Y., Xu, Y., and Lu, S. Device-Based LTE Latency Reduction at the Application Layer. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)* (2021).
[47] ublox. Even with 5G on the horizon, 4G LTE-M and NB-IoT devices show no signs of going obsolete. https://www.u-blox.com/en/blogs/insights/even-5g-horizon-4g-lte-m-and-nb-iot-devices-show-no-signs-going-obsolete, Apr 2020.
[48] Xie, T., Tu, G.-H., Li, C.-Y., and Peng, C. How can iot services pose new security threats in operational cellular networks? *IEEE Transactions on Mobile Computing* (2020).
[49] Yang, D., Zhang, X., Huang, X., Shen, L., Huang, J., Chang, X., and Xing, G. Understanding power consumption of nb-iot in the wild: tool and large-scale measurement. In *ACM Mobicom* (2020), pp. 1–13.
[50] Yang, H., Bae, S., Son, M., Kim, H., Kim, S. M., and Kim, Y. Hiding in plain signal: Physical signal overshadowing attack on LTE. In *28th USENIX Security Symposium (USENIX Security 19)* (2019), pp. 55–72.