USENIX

THE ADVANCED COMPUTING
SYSTEMS ASSOCIATION

GLaDoS: Location-aware Denial-of-Service of Cellular Networks

Simon Erni and Martin Kotuliak, *ETH Zurich;* Richard Baker and Ivan Martinovic, *University of Oxford;* Srdjan Capkun, *ETH Zurich*

https://www.usenix.org/conference/usenixsecurity25/presentation/erni

This paper is included in the Proceedings of the 34th USENIX Security Symposium.

August 13–15, 2025 • Seattle, WA, USA

978-1-939133-52-6

Open access to the Proceedings of the 34th USENIX Security Symposium is sponsored by USENIX.

GLaDoS: Location-aware Denial-of-Service of Cellular Networks

Simon Erni
ETH Zurich

Martin Kotuliak
ETH Zurich

Richard Baker
University of Oxford

Ivan Martinovic
University of Oxford

Srdjan Capkun
ETH Zurich

Abstract

Cellular communication is ubiquitous, but must be controlled in sensitive industrial and government areas. Existing cellular jamming systems rely on high-power, wide-band transmissions, which are non-selective and can cause interference in neighboring areas, e.g., blocking emergency calls. Also, meeting both the health limits of radio emissions and installation constraints while achieving effective coverage is highly challenging and sometimes even impossible.

Recent work introduced more power-efficient uplink protocol-level DoS attacks, which can effectively neutralize a connection from anywhere in the area covered by a base station. However, these attacks still need to be made selective to block communication only within a defined area and need to be able to detect all connections for all cells in the vicinity. In practice, this detection can be difficult if the cells are far away or under adverse channel effects. In contrast, a phone might be positioned in a strong radio path, allowing it to connect to such a cell.

To address the above challenges, we propose GLaDoS, a system that improves existing uplink protocol-level overshadowing approaches and combines them with low-power wide-band noise jamming to resolve weak cell issues. GLaDoS further limits DoS to the controlled area by integrating overshadowing with an off-the-shelf localization system.

We deployed and evaluated our system in an 62500m^2 area, close to an urban area, where the use of cellular phones is not allowed, but is fully covered by over 100 commercial operator cells. Our deployment made use of 4 protocol-level DoS units, approximately 40 outdoor and 100 indoor low-power jamming units, along with corresponding antennas and front end units. We evaluated our system within this area against different phone models and measured that our system neutralizes 99.3% of all connections, while being able to track over 100 cells simultaneously. This is the first full-scale deployment of an overshadowing-based cellular communication control system.

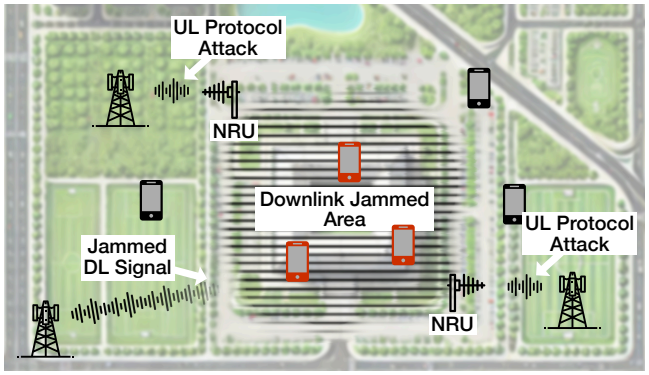

Figure 1: A high-level overview of our system. Distributed low-power downlink jammers neutralize weaker cells as shown for the base station in the bottom left corner. The two neutralization radio units (NRU) in the corners monitor and attack unwanted connections originating from within the controlled area with an uplink overshadowing protocol attack.

1 Introduction

The availability of cellular networks is crucial for modern society. However, their ubiquity also presents challenges when communication needs to be restricted, typically in highly sensitive corporate and government areas and buildings. In such scenarios, communication is typically restricted using jamming or Managed Access Systems (MAS) [5], which interfere with the downlink communication of commercial cellular networks by emitting strong signals within cellular bands (jamming) or create a stronger cellular network within the area (MAS). In order for such approaches to be successful, their signal (e.g., wideband barrage jamming) must be (significantly [19]) stronger than the downlink signal of commercial networks, which can vary widely depending on the cell in question and the location within a given area.

All downlink interference approaches present a key practical challenge: precisely controlling coverage. A system must prevent communication with both powerful, nearby cellular

towers and weak, distant ones – at all points within the restricted site – yet it must also avoid overspilling outside of the restricted area and affecting legitimate users nearby. While coverage areas of individual interference devices can be somewhat controlled by antenna direction and power output, this control is coarse-grained [5, 13]. In fact, the task is complicated further, as it is not always possible, for practical and security reasons, to install jammers in optimal locations, nor is it possible to select arbitrary power levels at sites with human occupants, as regulations limit exposure levels to avoid health concerns [23, 24]. As a result, it can be very difficult or even impossible to select appropriate installations and power levels that maintain coverage, do not interfere with communications outside the site, and provide effective coverage at the same time [5, 8]. Finally, simplistic jamming provides no means to permit special traffic, such as calling emergency services, which is the main reason such jammers are prohibited in the US [9].

Recent research has shown that far more power-efficient, protocol-level, denial-of-service attacks on cellular networks are feasible. These typically use overshadowing or signal spoofing [11, 19, 22] approaches that are much more power efficient and can target both downlink and uplink communication. However, even if such protocol-level approaches output at power that is below the regulatory health limits, unless adapted to be selective, they will still unintentionally cause denial-of-service outside of target areas. In addition, unlike classical jammers, these approaches require not only that their signals interfere with the reception at target mobile devices or their base stations but also, crucially, that they are able to correctly decode messages from *all* neighboring cells.

From the above, it is clear that designing a practical denial-of-service system that is selective within a target area, respects health limits and deployment constraints, and finally scales with the number of cells and mobile devices poses a significant challenge that goes beyond the current state-of-the-art.

In this work, we address this challenge and introduce GLaDoS, a distributed wide-area DoS system that addresses the issues listed above. GLaDoS evolves uplink protocol-level attacks and combines them with a classical *low-power* downlink jammer and a UE localization system, as shown in Figure 1. The classical jamming system is used to neutralize those cells with a weak downlink signal while using power that is below health limits and does not interfere with neighboring users. Meanwhile, a protocol-level uplink attack neutralizes connections from close-by and/or strong cells. In this case, each connection is first localized using the localization system, and if it originates from within the target area, it is blocked. The localization system that we used has high accuracy and within the evaluated target area correctly classified 98.27% of the connections.

While we will describe both downlink jamming and UE localization on a high level, they are explicitly not claimed as contributions to this paper.

We deployed and evaluated GLaDoS in a target area ($\approx 62500m^2$) that is in the vicinity of an urban environment, covered by ≈ 150 public commercial cells. In this environment, GLaDoS successfully classified up to 120 connections per second and attacked a connection every two seconds. From full-scale deployment tests we estimate that GLaDoS attacks 99.3% of all potential connections, however, 8 out of 12 tested locations achieve 100% coverage. Even in the worst location, we estimate the system to be 94.2% effective.

To summarize the contributions:

- We introduce GLaDoS, a novel attack system capable of blocking all LTE/5G NSA communication inside a defined area using a combination of low-power downlink jamming and reactive protocol-level attacks. Given the announced deprecation of 2G and 3G technologies, we focus on 4G/5G communication. GLaDoS presents key advantages over previous systems, specifically, no jamming over-spill to neighboring areas, providing effective coverage while respecting health limits and installation constraints, and allowing emergency calls. We achieve this by combining traditional low-power jamming and protocol-level attacks.
- We are the first to describe and implement a protocol-level attack system of this size showing the feasibility of a system based on state-of-the-art overshadowing attacks. We improve the reliability of existing overshadowing attacks, introduce attacks against the handover procedure, and show how protocol-level attacks can enable emergency calls even from an environment where no other cellular communication is possible.
- We detail how to implement a real-time software- and hardware-based attack system capable of monitoring all cells by all operators on all bands in parallel, significantly improving over the existing state-of-the-art which was able to handle only one cell at a time.
- We deploy and evaluate GLaDoS in a real environment. We evaluated the system simultaneously against all available cellular operators in multiple bands ranging from 700MHz to 2.7GHz. The system was evaluated against the latest generation of phones from various manufacturers. To the best of our knowledge, this is the largest deployment of such a system and it is validated across all LTE configurations in use by the operators at the time.

2 Background

2.1 LTE Basics

In LTE networks, the user equipment (UE), such as a smartphone, communicates with a base station (eNodeB) over a radio channel. A connection starts by the UE sending a random access preamble (RA preamble) to the network on the

physical random access channel (PRACH). Subsequently, the eNodeB responds with a Random Access Response (RAR), containing the radio network temporary identifier (RNTI) assigned to the UE, the next uplink grant, and the timing advance (TA) command, specifying how far in advance the UE needs to transmit to compensate for the channel delay.

In the uplink grant, the UE sends an Radio Resource Control (RRC) connection request together with any previously allocated temporary mobile subscriber identity (TMSI). If no such value is available (e.g., after switching on the phone), then the value is replaced by a random number and indicated as such.

The eNodeB replies with the RRC Connection Setup message, which contains dedicated configuration for this UE and a copy of the received RRC Connection Request message, to prevent possible contention between two UEs. Finally, the UE transmits the RRC Setup Complete message to complete the procedure.

Once the connection on the RRC connection between the UE and the eNodeB is established, the UE attempts to establish data services by connecting and authenticating itself to the mobility management entity (MME). It does so by initiating a non-access stratum (NAS) Attach or Service Request procedure. If, for any reason (e.g., failed identification or authentication), this procedure fails, the MME responds with an Attach or Service Reject message. Depending on the cause contained within that message and the number of retries, the UE will either try the procedure again or disconnect from the network entirely.

A UE does not camp on one cell forever. Once a more favorable cell is detected, the currently serving eNodeB starts a handover procedure, where the UE is transferred to another base station. In this case, the UE disconnects from the current eNodeB and sends an RA preamble to the new eNodeB with a special pre-allocated preamble index. The new eNodeB is aware of the preamble index, and the connection continues seamlessly without any NAS procedure.

During an emergency call, a UE makes a connection to any cell with an establishment cause "emergency" contained in the RRC Connection Request. In such a case, special procedures take place, such that the phone call can be set up as soon as possible.

2.2 LTE/5G-NSA Protocol-Level Overshadowing Attacks

Overshadowing describes an effect where one transmission is sent at the exact same time and frequency as another, but with slightly higher power. The stronger signal will be decoded successfully at the receiver, *overshadowing* the original transmission.

In LTE, there are two types of overshadowing attacks: Downlink and Uplink overshadowing attacks. These attacks can expose the international mobile subscriber identity (IMSI) or cause a denial of service (DoS). For our use case, we focus on the overshadowing attacks that cause DoS.

2.2.1 Downlink Overshadowing

An LTE base station can transmit with as much power as 50 Watt excluding antenna gain, resulting in an Effective Isotropic Radiated Power (EIRP) of multiple kW. To overshadow such a downlink (DL) signal, the signal must be transmitted with even more power than the base station, or from an antenna closer to the victim UE than the base station. Using downlink overshadowing, one can cause indiscriminate DoS of a whole cell, e.g., using SIB overshadowing [30], or more targeted overshadowing by e.g., replying to Service / Attach Requests with Service / Attach Rejects [11]. By controlling the output power, the range of the attack can be limited.

2.2.2 Uplink Overshadowing

For uplink (UL) overshadowing, one only needs to overshadow the transmission of UEs, which transmit with only ≈ 0.2W. It is therefore much more power efficient to perform uplink overshadowing. In addition, the attacker does not need to be close to the victim UE and can attack every single UE attempting to connect to a cell from a single location in the coverage area of the cell. Therefore, just using power control mechanism, it is not possible to limit the range of an uplink overshadowing attack.

Uplink overshadowing attacks that cause DoS work by overshadowing, i.e., replacing, the original Attach Request with an Attach Request containing a blocked IMSI [11] or replacing the Service Request with one containing an invalid MAC [11]. Both will result in an identification or authentication failure and a subsequent NAS reject from the MME.

Currently, no research paper elaborates on how to attack handovers specifically, as they cannot be attacked with these conventional downlink or uplink protocol-level overshadowing attacks.

2.3 Network Planning

Most of the academic research on protocol-level attacks concentrates on showcasing the attack by attacking a single base station. However, in order to provide a proper protocol-level attack system, the system has to operate on every operator, on each frequency band allocated to all operators, and ultimately on each cell in the vicinity of the targeted area to which the UE might connect. To paint a picture, in a single country it is common to have three to four operators present on five to eight frequency bands, with three or more cells operating on the same frequency.

Low-frequency bands have good penetration through obstacles such as walls and experience less path loss. However, these bands are generally smaller in bandwidth, which means

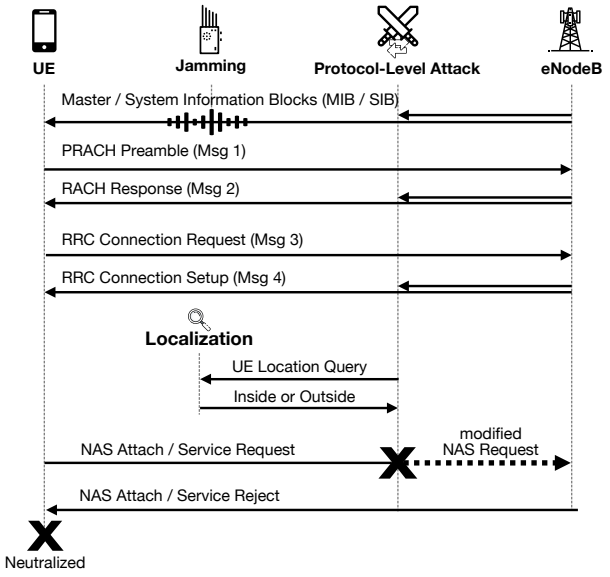

Figure 2: In **step (1)**, the system continuously jams the downlink and blocks the reception of configuration messages (MIB / SIB), at least for "weak-cells". In **step (2)**, For "strong-cells", Protocol-Level attack monitors and detects new connections, and queries the localization system if the UE is inside the area or not. If it is, in **step (3)**, the protocol-level attack is executed and the phone is blocked from the network (i.e., *neutralized*).

that they can easily become congested. Therefore, after the connection is established, the eNodeB can also transfer the connection to another eNodeB on other frequency capacity bands, where larger bandwidths are allocated to operators.

The cellular network is a very dynamic environment. For example, operators are turning off cells during the night to conserve energy, base stations are often restarted to apply new configurations, or turned off for maintenance reasons. Also, new cells are installed to increase coverage or to support events with a high number of visitors.

Finally, there are many manufacturers of base stations that are popular with operators, such as Ericsson, Nokia, or Huawei [10, 14, 16]. Each operator may use a multitude of vendors, and each vendor will configure their network slightly differently. This means that to operate in a cellular network reliably, any UE will have to support a large number of combination of features.

3 System Design

The GLaDoS system can effectively deny cellular communication in a controlled area through a combination of targeted protocol-level attacks and untargeted low-power downlink jamming. Shown in Figure 2, at a high level, the GLaDoS system performs the following steps: **(1)** GLaDoS continuously transmits low-power noise on the downlink frequencies, which increases background noise. This neutralizes cells that are far away or experience other unfavorable channel effects. The low-power noise jamming reduces the number of available cells. **(2)** GLaDoS continuously monitors all traffic on the downlink of the base stations surrounding the controlled area for new UE connections. Upon detecting one, GLaDoS notifies the distributed UE localization system, which decides if the UE is inside the controlled area or not. **(3)** If the UE is determined to be inside, a protocol-level uplink overshadowing attack is launched to prevent the connection from succeeding, i.e., neutralizing it. Due to the persistent nature of the protocol attack, the UE will not re-attempt to connect to the cellular network without manual intervention. As soon as the UE leaves the controlled area, the user thus has to toggle flightmode, restart the UE or reinsert the SIM card to regain connection. Since this system can only be operated in controlled areas without public access, it is feasible to inform all persons entering and exiting the area of the necessary steps.

The GLaDoS system consists of two main hardware units: low-power downlink jammers (DJs) and neutralization radio units (NRUs). Both operate in real-time over the entire available cellular spectrum. DJs transmit a low-power noise signal over the downlink cellular frequencies, while NRUs monitor the downlink traffic of the base stations and perform the uplink overshadowing attack if necessary. Only one NRU is required and active per cell, but multiple NRUs are placed around the controlled area such that for every nearby cell there is at least one NRU with good channel conditions towards that cell. However, in some cases, a cell is so far away, or interference and channel conditions are so poor that the downlink of a cell cannot be decoded reliably. Thus, also the uplink protocol-level attacks will be unreliable on such cells. These cells we call “weak cells”. In contrast, UEs inside the controlled area might still be able to decode a weak cell under some circumstances due to the high variability of the channel conditions. For these cases, we employ DJs. They transmit low-power noise, which is enough to make such weak cells undecodable for the UEs within the area. However, because of their low power, not all cells are jammed by this, leaving “strong cells” still available.

By itself, an uplink protocol-level attacker cannot distinguish between phones inside or outside a certain area. This is why up until now only downlink attacks (i.e., jamming or MAS) have been considered for limiting access in controlled areas, as an uplink attack would block all connections of the entire cell. However, because of the localization system, our protocol-level attacker can distinguish between connections coming from the inside or outside of the controlled area. Thus, connections from outside the controlled area can continue to work as normal.

3.1 Attacker Model

We consider users who possess mobile phones in areas where that is prohibited. In addition, we handle sophisticated and motivated users that are capable of locking phones to certain operators, bands, or cells, i.e., we do not restrict the cell selection process of the UE. However, we only consider cellular (LTE / 5G-NSA) communication with legitimate base stations of operators at their allocated frequencies and exclude other types of wireless communication (e.g. Wi-Fi).

3.2 Step 1 - Low-Power Downlink Jamming

In an urban setting, modern devices can easily decode as many as 20 cells from a single location. The UE (and, in some cases, also the user) is free to choose which one of these to connect to. Thus, a user can choose a cell that is very weak, with low resulting throughput, but that still allows communication.

Decoding these low-signal-quality cells with SDR solutions such as GLaDoS's NRU is not always possible. For these cases, we employ our low-powered DJs which drown the weak cell signal in noise, making it impossible to decode by all UEs. We target the jamming signal using directional antennas to concentrate the energy inside the area and minimize the impact outside of the controlled area.

Using these precautions, users outside the controlled area will unlikely be affected by GLaDoS as there will still be enough strong cells visible, for which the leakage of the jammers is negligible.

3.3 Step 2 - Cellular Communication Monitoring

The GLaDoS system needs to monitor the communication of all remaining cells of all operators across all bands that a UE could see from anywhere inside the controlled area. Since one can expect multiple UEs to be active in parallel or in short succession, and a UE can choose to connect to a cell at any time, the GLaDoS system cannot use techniques such as hopping or listen-and-wait to reduce the amount of processing but must monitor all traffic in parallel in real-time. To the best of our knowledge, this is the first system described in research that is capable of real-time decoding of all downlink transmissions from all base stations on all bands from all operators simultaneously.

Every new connection starts with a Random Access Procedure. The RAR message is sent by the eNodeB with a low code rate and QPSK modulation since it has to be reliably received by the UE even in challenging situations. This makes the message easy to decode by the GLaDoS system even in the presence of interference. Thus, it is an ideal message to detect new connections on a cell.

The localization system that we use takes as an input an uplink allocation contained in the RAR. Once a RAR is detected, the NRU forwards it to the localization system. The localization system then responds with the classification of the connection. It is either inside or outside of the controlled area.

3.4 Step 3 - Protocol-level Attacks

At this stage, the system has both detected a connection and classified it as either coming from outside the controlled area, in which case it is now simply ignored, or as coming from the inside. In this case, the system starts a protocol-level attack in order to prevent the successful establishment of a connection.

This decision is made at the very last moment when an uplink transmission of the UE after the RRC Connection establishment is imminent. This is done to give the system enough time for the geofencing step. There are four possible states in which the UE could be, corresponding to different protocol flows and thus attacks. The system uses the PRACH sequence index and the RRC Connection Request (both received on the downlink via the RAR, and the Contention Resolution ID, respectively) to distinguish between the states and to react with an appropriate attack.

In all attack scenarios, GLaDoS only transmits exactly on the time and frequency resources granted to the UE that is under attack, that is, GLaDoS *overshadows* only the victim UE transmissions. This affects thus only the UE connection under attack and no other connections, as they are scheduled on different time and frequency resources. Tests undertaken in network-operator test labs have also confirmed that there is no impact on other connections or service degradation.

In the case that the RRC Connection Request contains `emergency` as establishment cause, the connection is immediately allowed since this connection represents an emergency call. This procedure is always possible for any UE, even without a (valid) USIM, thus also after being subjected to a persistent DoS attack executed by GLaDoS [3,4].

3.4.1 Initial Attach

In this state, the UE does not have an existing connection context. Thus, the UE first transmits an RRC connection request with a random identifier, which is how this state is identified. The subsequent NAS Attach Request message transmitted by the UE gets replaced (overshadowed) by the GLaDoS system with an NAS Attach Request message containing an invalid IMSI. This leads to the network responding with a NAS Attach Reject message and to the DoS of the UE attempting to connect. As shown in [11], this uplink overshadowing DoS attack is persistent (depending on the UE model, up to 3 retries are done), which means that an illegal UE will not cause any additional signaling traffic and thus also no further load on the network.

3.4.2 Service Request

Here, the UE has an established connection with the core network, but the radio connection has been suspended to preserve power. Upon receiving a paging message or request to transmit some data from the user, the UE reconnects to the radio network (i.e., eNodeB). It first transmits an RRC connection request containing its previously assigned temporary identifier (S-TMSI) and an establishment cause of `mt-Access` or `mo-Data`. This triggers the protocol-level attacker to overshadow the subsequent NAS Service Request message with one containing an invalid message authentication code, leading to a NAS Service Reject message from the network. The UE then drops its context and tries to attach again, resulting in the procedure shown in Section 3.4.1.

3.4.3 Tracking Area Update / Detach Request

Both sub-states are quite rare to encounter. A detach request originates from a phone that turns off, so we do not need to specifically handle this case and can arbitrarily overshadow it. For a tracking area update, the cause value is the same as in an attach request (`mo-Signalling`), and we thus cannot reliably distinguish it. Thus, we also overshadow it with an invalid Attach Request, which is not expected by the UE and eventually triggers the timeout of T3430 [4] and an increment of the failed tracking area update counter. After 5 of such attempts, the UE falls back to the Attach Request procedure, which is successfully attacked.

3.4.4 Handover/Reestablishment Procedure

During the Handover/Reestablishment procedure, the phone is already attached to the network, and a security context is setup between the UE and eNodeB. The previously mentioned attacks cannot be launched in this case. Instead, our protocol-level attacker sends random data in every uplink grant given by the base station. Effectively, every uplink transmission by the victim UE is overshadowed by buffer status reports of the MAC layer, reporting empty buffers of the UE. First, this makes the delivery of user data from the UE impossible, and second, informs the eNodeB that it can release the connection. Eventually, after 10-30 seconds, eNodeB will send an RRC Connection Release, terminating the connection.

4 Protocol-Level Attack Software Implementation

An NRU contains multiple sub-NRUs, each of which uses a custom FPGA-based SDR for radio sample RX/TX, along with general-purpose processors performing the protocol-level attack. Each sub-NRU is capable of RX/TX for the entire bandwidth of a given cellular band and accepts a pluggable band-specific frontend to tune it appropriately. As such, the system can scale across required bands by replicating the sub-NRUs with appropriate frontends. The general-purpose processing is provided by commodity server hardware. Scaling to cover multiple cells within a band can thus be handled in software via parallel processing.

In general, the software is compatible with a range of SDR platforms, providing flexibility in selecting or custom-designing hardware to meet the specific requirements of a given deployment. We discuss the software implementation in detail here, with the hardware being described in Section 5.

4.1 Multi-Cell Processing

In existing literature, most of the cellular protocol-level attacks and analysis systems can monitor and attack only one base station at a time. In the best case, Ross and Reaves [25] created a system that can switch between cells in less than 100 ms and could therefore monitor multiple cells – but also not in parallel.

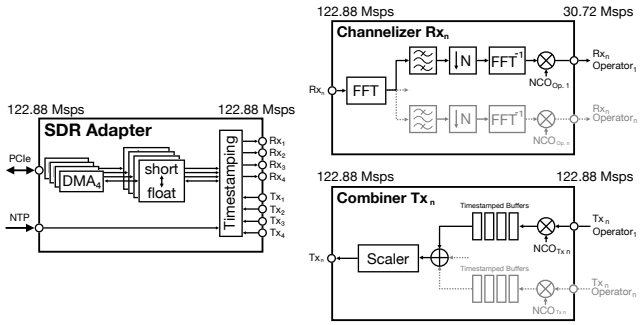

Figure 3: Multi-Cell Processing. Each received IQ stream is channelized and downsampled for the respective operator allocation and distributed to individual cell instances. Generated samples are shifted in frequency, summed, and scaled before transmitting.

We are the first to propose and implement a soft- and hardware system which can operate on multiple base stations in parallel on a single SDR. The solution consists of three parts. **(1)** Our SDR is designed such that we receive and transmit on four ports at 122.88 Msps each with enough bandwidth to receive and transmit on each port a whole band with all its operators and cells in it. The SDR adapter converts between int16 and float32 and performs the necessary timestamping operations. **(2)** On the receive side, a channelization and distribution step turns the wideband signal into several 30.72 Msps streams, one for every cell to be decoded. The signal is also shifted in frequency such that the cell center frequency is at 0 Hz, simplifying subsequent decoding. **(3)** On the transmit side, we generate the uplink signal at 122.88 Msps (with the cell signal in the center). Then we shift it in frequency and combine it with other concurrent transmissions from other cells before transferring the signal to the SDR.

Further details on how these three parts (channelization, distribution, combination) are implemented together with the SDR adapter can be found in the appendix in Appendix A.

4.2 Cell Discovery and Management

The cellular network is highly dynamic, as cells sometimes change their PCI or turn on and off at different times during the day. Our system needs to react quickly and autonomously to these changes; thus a dynamic cell search and management system had to be developed.

The system consists of two parts. First, we run a cell search continuously in a dedicated thread in order to detect new cells as soon as possible. The thread looks for the primary and secondary synchronization signals (PSS/SSS) of the cells by cross-correlating both PSS and SSS signals on the channelized signal. Once a cell has been found, it outputs its EARFCN and PCI tuple. Second, the cell management spawns the new cell instance according to the EARFCN and PCI on a new thread and starts the decoding process. It then checks the decoding quality and broadcasts metrics such as the average SNR and successful decoding rate to other NRUs listening on the same band. If two or more NRUs can decode the same cell, they mutually agree on which should be handling this cell based on these metrics. If the cell does not need to be decoded anymore, its instance is destroyed. This step is done periodically in a peer-to-peer fashion over a custom gRPC protocol.

4.3 Cell Downlink Decoding

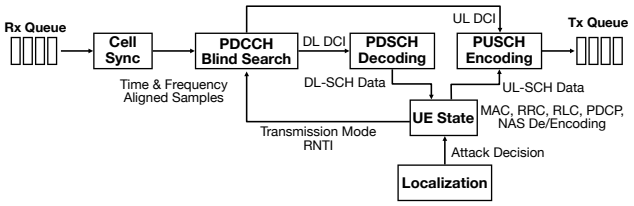

Figure 4: Processing Steps for an Individual Cell

For each cell found by the Cell Search algorithm, our system creates a cell instance running on a dedicated core. Since we can decode a downlink subframe in less than 1*ms* on a single core (see Section 6.1), we do not need to parallelize the decoding, simplifying the implementation. To monitor the downlink of a cell for new connections or transmissions, the cell first needs to have a sync; that is, it needs to determine its time and frequency offset to the cell's downlink. Since the SDR has a high-quality clock (see Section 5.2.5), we found that we surprisingly do not need to perform frequency offset correction at all. After synchronization, each subframe is processed sequentially. First, the physical downlink control channel (PDCCH) is monitored for downlink control information (DCI) messages in a blind decoding step. These DCIs carry information for either a message in the downlink in the current subframe, or a time and frequency allocation (grant) to transmit a message in the uplink in +4 subframes. Next, if we found any DCIs for downlink messages, we decode the physical data shared channel (PDSCH) for these messages and forward the bits to the relevant UE instance. Finally, we also forward the uplink grants to the UE instance, such that it can decide if it needs to send something on the physical uplink shared channel (PUSCH) or not, as discussed in Section 4.6. Next, we will describe our implementation of the optimized DCI blind search, while more details about the data decoding from PDSCH up to the NAS messages can be found in the appendix in Appendix B.

4.3.1 DCI Blind Search

DCIs have a CRC checksum appended to them, which is XORed with the RNTI to which the DCI is addressed. A UE is constantly monitoring the PDCCH for DCIs destined to its allocated C-RNTI. In this expensive process, called *blind search*, all possible formats and locations (candidates) in the PDCCH are decoded and checked.

We monitor the PDCCH for DCIs destined at UEs that are either under attack or whose location (in/out) has not been decided yet. Furthermore, we need to detect all RAR messages since they contain the C-RNTI of new UE connections. Hence we need to also monitor the RA-RNTIs, a special RNTI that does not belong to any other user. Any DCI Format 0 messages (indicating an uplink transmission of the phone in 4 subframes after the current one) are passed to the UE instance, which will encode a PUSCH message and send it to the radio to overshadow the original UE transmission. Any decoded downlink DCIs are passed on to the PDSCH decoding step, which will ultimately forward the contained message bits to the UE instance as well.

A simple implementation of a blind search would iterate through all RNTIs, decode every location and format, and check if the CRC matches the RNTI. In our software, we optimized the blind decoding to be able to monitor up to 20 concurrent UEs in less than 1ms, as shown in benchmarks in Section 6.1.4. First, we decode the common search space locations for DCIs with Format 1A/0 for all RNTIs, and Format 1C for non-user RNTIs (i.e., RA-RNTIs, or SIB). If the CRC of the decoded DCI matches any of the RNTIs we are looking for, we store the found DCI. Next, we generate a map of all UE specific locations for the C-RNTIs and decode all locations for DCIs Format1A/0 and Format 1/2/2A/... - depending on the transmission format that was indicated in the RRC Connection Setup message for that UE. A key insight is that user-specific locations that overlap are decoded only once per format.

The individual locations are only decoded if sufficient en-

ergy is present in these regions. This means in practice many locations can be skipped entirely. To make this energy calculation as fast as possible, we first store the cumulative sum of all energy values in a separate array A. We can then query that array for the sum of the energy in the PDCCH region from i to j as $A(j) - A(i-1)$, which is a constant-time operation, instead of calculating the entire sum every time.

4.4 Localization

For the localization we use a system based on a distributed set of uplink receivers around and inside the controlled area. GLaDoS informs the localization system about uplink transmissions, and the localization system measures the signal power and signal quality indicators of these transmissions. There are three types of uplink signals the system measures: a PRACH preamble, the reference signal inside the PUSCH, and the reference signal in the PUCCH transmission. The physical layer measurements of the signals from all the receivers are collected on a server where a pre-trained and deployment-specific localization model infers if the transmission originates inside or outside of the controlled area.

4.5 Timing Advance

For the eNodeB to be able to reliably decode many UE transmissions in parallel, they must all arrive at the same time at the eNodeB. Thus, UEs must measure and compensate the flight time of their transmissions. To this end, after receiving a PRACH, the eNodeB measures the difference from the expected arrival time and reports that in the RAR message. Therefore, our protocol-level attacker also periodically sends PRACHs and monitors the response from the eNodeB to learn what timing advance it should apply to its transmissions. This is then the same for all UEs for this base station.

This procedure also serves for a continuous health check of the system, since the whole receive and transmit chain needs to be operational. In essence, if none of our RARs can be decoded, something must be wrong with the transmitter or receiver part.

4.6 Uplink Transmission

After it has been decided that the connection needs to be attacked, our system continuously listens for uplink grants for the connection and reacts and transmits on each to reliably block all communication from the UE. If the grant is not of sufficient size to transmit an attack message, a buffer status report is sent on the MAC layer, indicating that there is still data to send. The same attack message is repeated on every suitable uplink allocation until a matching RLC acknowledgment message is received on the downlink. After that, empty buffer status reports are sent until a reject or release message is received, or there was no uplink allocation received for more than one minute (which indicates that we failed to decode the reject or release message).

After the data is encoded up to the MAC layer, it is passed on to the PUSCH encoding. As shown in Section 6.1.3, this will take too long if > 1 UE are to be attacked in a single subframe. Therefore, the PUSCH encoding is parallelized, and a task is issued to a workerpool per UE to encode and send a PUSCH transmission, such that the cell instance may continue to decode the downlink. How this is implemented is shown in the appendix in Section C.1.

After the uplink transmission by the radio, the system looks for the (negative-)acknowledgment on the Physical HARQ Indicator Channel (PHICH) and re-tries the transmission another if there was a NACK. However, even if the transmission was acknowledged on the PHICH, a subsequent uplink grant could specify to re-transmit it again (on a potentially different uplink frequency location), which our system then also does.

In summary, we have implemented the attacks to be reliable on both lower (HARQ) and higher (RLC) layers, retransmitting the message whenever necessary. Note that we are not transmitting any downlink acknowledgments on the physical uplink control channel (PUCCH) at all, but leave this job to the UE, such that it is able to receive the downlink reject message reliably.

4.7 Latency Requirements

We are running the attack software on a multicore x86 host platform. Our system must operate in real-time and comply with a specific set of time constraints. First, there is a strict time between the reception of the uplink grant in subframe n and the corresponding uplink transmission in subframe $n+4$, leaving the system with exactly 3 ms to process any downlink data and generate the uplink response. Second, we have limited time for the localization system to make the decision about the connection's origin on the order of tens of milliseconds. These constraints make the system a *firm* real-time system [17]. We show how this is achieved in the appendix in Appendix C.

4.8 Observability

To debug a system of this complexity, metrics, logs, and also traces are very helpful, as long as they do not impact the system performance when collected. A metric can be the temperature of the CPU, FPGA or ADC, an ever-increasing IQ sample, attack, or connection counter, or even a histogram such as the distribution of modulation and coding schemes. To expose these, we used an open-source polling-based metric collection and storage system. Similarly, to store and query logs, we used an open source log aggregation system.

To investigate sporadic performance and latency issues, we integrated an open source tracing tool into our system. Using that, we instrumented all key functions in the code,

from SDR DMA transfer loops, over channelization, all the way to downlink sync, decoding, and uplink encoding and transmission. This way, performance regressions are visible and can be debugged by requesting a trace.

4.9 Emergency Calls Filtering

If the `establishmentCause` value in the RRC Connection Request is set to `emergency`, our system can determine if the connection belongs to an emergency call. If so, we immediately ignore the connection and allow the emergency call connection.

To verify that a phone sets this value, we tested this hypothesis in our lab setup with an Amarisoft base station. We put our test phone in a Faraday cage, verify that no other cells are visible, and call 9-1-1. We saw that the emergency call flag was present in the RRC Connection Request.

5 Real-World Deployment

We implemented our system and deployed it in a controlled area where cellular communication is prohibited. The area is $\approx 62500m^2$ large. The area is on the outskirts of a city with many neighbors in the vicinity. Around 140 downlink jammers (DJs) were installed in total inside and outside of the buildings in the area. Around the perimeter of the area, 4 NRUs were installed. The locations of the NRUs were carefully chosen to optimize for direct line of sight to many base stations.

Surveys conducted inside the area with a commercial network analyzer device showed that more than 150 LTE cells of different operators on 5 different bands located around the area, of which 70 could have their SIB decoded. Once low-power jamming is on, the number of cells drops to around 30. These numbers are just indicative, as the number of cells fluctuates because the operators are turning them off and on, but still showing the impact of the low-power downlink jammers.

This deployment covers five FDD LTE frequency bands – all those available in the country. These bands span as little as 30MHz and go up to 80MHz. On all these bands, our downlink jammers transmit noise on the downlink frequencies and NRUs are able to receive the downlink and transmit on the uplink frequencies.

5.1 Low-Power Downlink Jammers (DJ)

Inside the controlled area, we deploy two types of downlink jammers. For outdoor coverage, we use ≈ 40 jammers configured with an output power of 16dBm per band, mounted on poles between 2-3m high and using a directional antenna oriented towards the inside of the controlled area. Indoors, we use ≈ 100 jammers configured with 10dBm output level per band with omnidirectional antennas mounted in the ceiling. The output power of all the jammers are further adjustable in the range of $\pm$ 10-20 dB, such that we can fine-tune the system to keep the balance between overspill and neutralization of "weak cells".

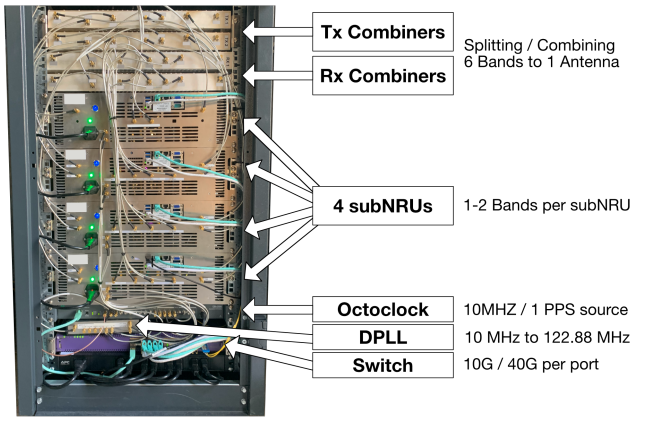

Figure 5: Picture showing a deployed NRU. It consists of an assembly of four subNRUs, combiners, clocking infrastructure, and networking. We implemented the protocol-level uplink overshadowing attacks on top of this specialized hardware platform.

We estimate the jamming power received outside the area on a single cell to judge the impact of the jamming on the outside of the area. Starting with a signal strength of 16dBm, cable loss (-3dB), opposing antenna directionality (i.e., directional antenna facing away from the neighboring area, -10dBi), free-space path loss at the nearest distance of $\approx$30m at 800MHz (-60dB), and the power of the jammer spread over one band (-6dB, assuming four operators), we end up with around $<$-63dBm per cell. From experiments, we established that we need 20dB of J/S to neutralize a cell. From measurements carried out with a commercial cellular scanner, we estimate that at least 80-90% of the 70 cells around the immediate area would still be available, as they have a maximum power of > -83dBm. This amount is sufficient for normal operation of the radio network, since users will be able to connect to their respective operators on the remaining cells.

5.2 Neutralization Radio Unit (NRU)

We build our protocol-level attack software on top of an SDR hardware platform called a Neutralization Radio Unit (NRU). In the following, we detail how this platform works.

5.2.1 Software-Defined Radio (SDR)

A x86 host-based processing approach gives us full flexibility in terms of how the attack software is implemented. To this end, we first evaluated commercially available software-defined radios (SDR), but found that based on the number of devices necessary to cover the full spectrum with the amount

of devices required for the area, no COTS device would be commercially viable. Thus, we opted to have an SDR designed for our needs.

We selected an architecture that provides us with a 16bit sample resolution with a sampling rate of 122.88Msps. Each SDR is paired with a x86 computer and exchanges IQ samples over PCIe, forming a *subNRU*. Using a multitude of these *subNRUs*, we have the capability to receive and transmit on the full spectrum necessary.

5.2.2 Host Computer Specifications

We use a modern multicore CPU paired with 128GB RAM and a 1TB SSD. We choose this CPU because it is the CPU on the market with excellent single- and multi-core performance while being very cost-efficient.

5.2.3 Transmit RF Front End

We know that the power necessary to overshadow an UE on the uplink can be up to 26dBm in the absolute worst case for a cell received on the cell edge. This is because 23dBm is the maximum output power [2] for a UE of standard power class 3, with an additional 3dB margin to reliably overshadow the connection [11, 22].

To achieve this, an uplink bandpass filter and a power amplifier (PA) are added after the transmission port to amplify the signal in order to reach the required power to overshadow the UE uplink transmissions.

Before going towards the antenna, the transmit channels from multiple subNRUs are combined using RF combiners.

5.2.4 Antennas

The protocol layer attack system must attack all frequencies simultaneously. Thus, the antennas need a wide frequency range from 700MHz all the way to 2.7GHz. Next, the channel towards the base station should experience as little interference from other base stations on the same frequencies as possible. Thus, we selected antennas with a narrow horizontal beamwidth of around 30 degrees.

We mounted the antennas on tall structures inside the controlled area, ensuring a direct line-of-sight to the eNodeBs where possible. However, since there are way more base station sites (around 15 larger groups consisting of 3-12 cells each) than we have NRUs (or antennas), we cannot ensure perfect direction to each cell. Each tall structure we equip with two NRUs, with four antennas on top of each structure, and point them north, west, south, and east. We fine-tune the directionality such that we cover the closest cells to the controlled area as much as possible.

Each antenna has two ports with different polarizations. We have found that it depends on the polarization which cells the system can decode. Therefore, we have connected an NRU to each antenna-polarization pair at both locations. We have then brute-forced the optimal solution, to determine which NRU to connect to which antenna port such that an optimal coverage is achieved.

5.2.5 Clock Management

Both the NRUs and the localization system need to be time-synchronized with a deviation ± 5 ns between each of them. If all devices are co-located together, simple solutions to distribute the analog clock distribution (i.e., 1PPS and 10MHz) signals over a coaxial cables could suffice. However, as we have large distances between the devices, we cannot use coaxial cables. We selected a high-precision timing product to distribute both clocking and timing, achieving a PPS signal with <1ns deviation between any 2 endpoints at all times. We also considered GNSS-based solutions, but those can be easily jammed by any non-technical adversary. However, it is still used as a backup solution to the precision timing solution in case of equipment failure.

6 Evaluation

To evaluate the system, we first show synthetic but realistic benchmark results of the protocol-level attack. We then describe an extensive full-system test in the real world with our own phones.

6.1 Benchmarks

We benchmarked key components of our system, to show that attacking on all of the cells in real-time can be achieved. We see that the full subframe (1 ms) processing, that is, channelization, DL decoding, and final UL encoding, takes, in the worst case, $354us + 224us + 466us = 1044us$. The uplink encoding does not need to be done in all subframes and the channelization step is done in parallel on a different core. Therefore, our system meets the latency and performance requirements set out in Section 4.7.

6.1.1 Multi-Cell Channelizer performance

# Operators	6	5	4	3	2	1
Runtime [us]	354	337	297	298	257	250

Table 1: Benchmark of channelization for various amounts of operators.

6.1.2 Physical Downlink Shared Channel Decoding

We first benchmarked the channelization step and show the results in Table 1. The input to the channelization is 1ms worth of IQ samples at 122.88MHz. The channelization outputs an

IQ streams at 30.72MHz sampling rate, centered at the operators frequency allocation. The number of streams depends on the number of operators that have a frequency allocated in a band. We see that the time required for channelization grows with the number of output streams. In the most common case, there are four operators per band.

SNR [dB]	0	5	10	15	20
Runtime [ms]	22.4	18.5	14.8	14.7	14.7

Table 2: Benchmark of decoding a PDSCH message in 100 subframes in a row.

Next, we benchmark the decoding of PDSCH messages for 100 subframes in a row for a typical 10MHz cell, EPA channel model, using a R.10 reference downlink waveform generated using MATLAB [29]. This consists of first doing a PDDCH blind search for one RNTI and then PDSCH decoding. We can see the results in Table 2. For low SNR values, it takes longer to decode, however, for SNR above 10dB, the decoding time is almost equal regardless of the SNR.

6.1.3 Physical Uplink Shared Channel Encoding

PUSCH PRBs	5	10	20	30	40
Runtime [us]	435	443	453	466	452

Table 3: Benchmark of encoding a PUSCH message and producing the TX waveform.

We benchmarked the PUSCH message encoding from bits to IQ samples and present the results in Table 3. The cell size is set to 10 MHz (50 PRB) and we measure the time to encode one subframe with a 122.88Msps sampling rate. We use a very low coding rate and QPSK modulation for this message (mcs_idx = 2).

6.1.4 DCI Blind Search

# RNTIs	2	4	8	16	20
Naive [us]	119	341	778	1656	2086
Optimized [us]	121	216	268	281	288

Table 4: Benchmark and comparison of different DCI blind search methods.

Finally, we benchmarked our improved DCI blind search described in Section 4.3 against naive sequential DCI decoding. In the naive DCI decoding, blind search is performed for each RNTI separately. This used to be one of the main bottlenecks of our system. However, we can see that for the naive blind search, the runtime scales rapidly above 1 ms with the number of RNTIs, whereas for our new blind search algorithm, the decoding of additional RNTIs increment the runtime only minimally and stays well below 1 ms. This is important, as this blind search step has to be done every single subframe (1 ms).

6.2 Full-Scale Deployment Evaluation

We created a testing setup that we can fit in a backpack. The setup consists of four rooted phones connected to a laptop running a commercial software that allows to control and monitor the cellular connection of the phones. The software allows us to connect to the debug port of the modems inside the chips and to read the RRC and NAS messages sent and received by the phone. The four phones we used are: Samsung Galaxy S12, two Samsung Galaxy A24s, and Oppo Reno 8 Pro. This gives us a selection of three different modem manufacturers: Qualcomm, Exynos, and Mediatek. Each phone had a SIM card from one of the four operators available in the country, such that we can test the system against every operator. Every 7 seconds, we turn the airplane mode on and off again to force a new connection. We found that 7 seconds is a time period in which a phone completes a connection in a normal setting without any kind of jamming solution. Toggling the airplane mode is done to ensure that the phone will attempt to re-connect to the network even after having received a NAS Attach Reject message.

We conducted an over 2h long evaluation campaign, during which our system was able to observe connections from up to 106 cells. For all of those cells, we received their SIBs and received responses to our own PRACH transmissions. These RARs contained timing advance values between 7 and 90, translating to a distance of 500m to 7km between our system and the cells. For this evaluation, we walked around inside and outside of the facility buildings. We visited 12 different indoor and outdoor locations. At each location, we waited between 3 and 15 minutes. In between locations, we kept the testing setup running. We call this time between locations "Walk", and it is mostly outdoors between different buildings. During the test, we had permission to turn on the entire system, low-power DJs were turned on to eliminate connections to weak cells and NRUs were blocking the remaining connections that were classified to be inside the controlled area.

First, we individually evaluate the performance of the low-power DJs and protocol-level attacks and then evaluate the system as a whole.

To evaluate the performance of the DJ, we first split up the measurement logs into 7 second epochs. If, in that time, a connection attempt was made (regardless if it was attacked or not), we determine that the downlink jammer failed to neutralize all cells, if there is none, we can conclude that it did. Thus, the performance of the DJ, $Perf_{DJ}$, is defined as:

	Low-Power Downlink Jammer					Protocol-Level Attack					Overall System				
	Op 1	Op 2	Op 3	Op 4	Tot.	Op 1	Op 2	Op 3	Op 4	Tot.	Op 1	Op 2	Op 3	Op 4	Tot.
Loc 1	88.5	100	84.6	100	93.3	100	-	100	-	100	100	100	100	100	100
Loc 2	100	100	100	100	100	-	-	-	-	-	100	100	100	100	100
Loc 3	100	100	100	100	100	-	-	-	-	-	100	100	100	100	100
Loc 4	97.4	100	100	94.9	98.1	100	-	-	100	100	100	100	100	100	100
Loc 5	100	100	96.2	100	99.0	-	-	100	-	100	100	100	100	100	100
Loc 6	68.3	100	95.8	97.5	90.4	100	-	100	100	100	100	100	100	100	100
Loc 7	43.3	68.3	50.0	98.3	65.0	100	94.7	100	100	98.8	100	98.3	100	100	99.6
Loc 8	50.0	100	19.2	69.2	59.6	100	-	100	100	100	100	100	100	100	100
Loc 9	69.6	100	100	100	92.4	23.8	-	-	-	23.8	76.8	100	100	100	94.2
Loc 10	100	76.7	71.7	100	87.1	-	78.6	100	-	90.3	100	95.0	100	100	98.8
Loc 11	23.1	100	46.2	100	67.3	100	-	100	-	100	100	100	100	100	100
Loc 12	84.6	80.8	30.8	53.8	62.5	100	80.0	100	58.3	84.6	100	96.2	100	80.8	94.2
Walk	83.8	97.0	90.5	97.8	92.3	94.3	100	100	100	97.0	99.1	100	100	100	99.8
TOTAL	**79.9**	**95.1**	**85.4**	**96.4**	**89.2**	**90.5**	**90.7**	**100**	**87.5**	**93.5**	**98.1**	**99.5**	**100**	**99.5**	**99.3**

Table 5: Performance of the DJ, protocol-level attack, and the GLaDoS system as whole. Performance is shown in percentages. We evaluated each component in 12 different locations and for 4 different operators. Finally, we output the overall success.

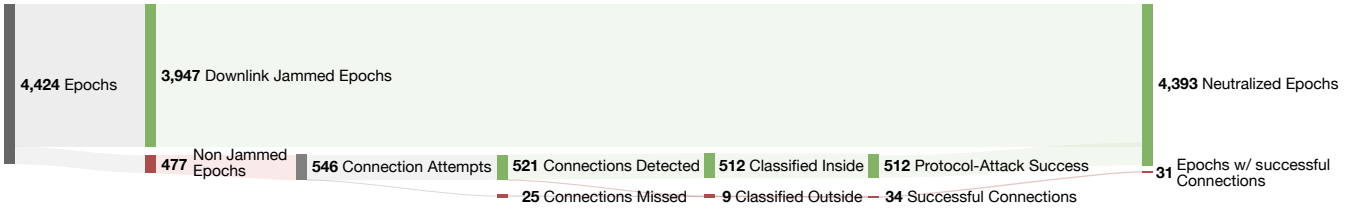

Figure 6: We had 4,424 epochs in which the phones had the airplane mode toggled on/off. In 3,947 epochs, the phones did not find a cell because of the downlink jamming. In the other 477 epochs the phones started at least one connection attempt to a cell since after getting a reject, it might re-try immediately for a different cell. In total, we had 546 connection attempts that were handled by the protocol-level attack. 25 of those connections were missed and 9 were classified wrongly as being outside the controlled area, leading to 34 successful connections belonging to 31 epochs.

$$Perf_{DJ} = 1 - \frac{\text{\# Connection Attempts}}{\text{\# 7s Epochs with Successful Connections}}$$

For the protocol-level attack performance test, we define that a connection has been established successfully if it has not received a Service, Attach, or Authentication Reject during its establishment. As in $Perf_{DJ}$, we split the time into 7s epochs. If there is an established connection in the 7s epoch, the attack fails. On the other hand, if there are only rejected connections in the epoch, we define this 7s time period as neutralized. In our tests, we do not consider Handovers/Reestablishments separately but they are classified as failed attacks (since a connection would be established first using a proper Attach procedure). The performance of the protocol-level attack $Perf_{Attack}$ is defined as:

$$Perf_{Attack} = 1 - \frac{\text{\# 7s Epoch with Successful Connections}}{\text{\# 7s Epochs with} \geq 1 \text{ Connection Attempt}}$$

In the end, the performance $Perf_{GLaDoS}$ tells us what percentage of connections made every 7 seconds were successfully blocked by either the DJs or the protocol-level attacks.

During our walk, the test phones tried to connect to 25 different cells and tried to make 546 connections. Figure 6 summarizes the results for each operator for the protocol-level attacks. To summarize the performance of the three steps of the protocol-layer attack, we achieve: 95.42% for DL detection subsystem, 98.27% for localization system used, and 100% for the UL attack subsystem. In Table 5, these are broken down further by location. Overall, our low-power jammers neutralize 89.2% of possible connections, and our protocol-level attacks neutralize 93.5% of the attempted connections. We conclude that the overall performance of GLaDoS is 99.3%. This performance depends heavily on location. We can see that in some cases the DJs are effectively neutralizing all connection attempts, making further adjustments (reductions) in power level necessary to still allow emergency calls.

6.2.1 Failed Protocol-Level Attacks Investigation

In this subsection, we investigate the failed attacks by examining both the logs of the rooted test phones and our system. The software to control the rooted test phones reports the contention resolution ID, which is in our case the RRC Connection Request. We can use this message to find this particular connection attempt in the GLaDoS system logs based on the random value sent during the RRC connection establishment. We can then investigate every successful connection and determine why it was not neutralized. Of the 34 successful connections:

- **9 connections** succeeded due to a misclassification by the localization system. This problem is restricted to one location, Loc 9 as reported in Table 5. We have not seen any other misclassifications in other locations.
- **25 connections** were successful because the protocol-aware attack missed (i.e., did not decode) the connection attempt. Out of those:
  - **5** were due to the operator activating a new cell. The system takes a short time to find the new cell and assign it to an NRU, which lead to missed connections.
  - **3** occurred because a cell was moved from one NRU to another, due to better decoding quality there. During this time period, the connection were missed.
  - **9** were due to overflows / loss of synchronization.
  - For the **8** remaining failures, the system was unable to decode the RRC Connection Setup, likely due to low SNR / bad channel conditions.
- **0 connections** succeeded during the protocol-level attack stage. That is, once a connection was detected and correctly classified as inside, we have successfully launched an attack on it, and the phone received the intended reject message.

After the experiments were completed, we have found an inefficient part of the code which was causing Rx overflows in the system the longer the system was running. Since then, this issue has been resolved and we can expect fewer such Rx overflows. There was one connection to a "weak cell", which should have been removed by downlink jammers. Finally, we have found an issue in the power boards that supply the transceiver board that resulted in very high phase noise. This bug was since resolved, and decoding issues should occur more sporadically in the future.

Put together, this means that we can expect an even better performance from a more mature version of our current prototype system.

6.3 Outside Impact

We evaluated whether users outside of the controlled area are affected by low-power jamming or the protocol-level attack.

6.3.1 Low-Power Jamming

To evaluate the impact of low-power jamming, we performed two extensive drive-tests with a commercial network scanner and a set of four phones (each with a SIM from a different operator). First, we turned low-power jamming on, and afterwards turned it off as the baseline. In both tests, we have disabled the protocol-level attack. During the drive tests, we have regularly stopped, especially at locations closest to the controlled area. During these stops, we have run a speed test and also observed the latency by observing the RTT between the phone and a server reachable over the internet. At the same time, we have measured the signal strength (RSRP) and quality (RSRQ) of the cells with the scanner.

In none of the locations where we stopped (or during the drive test), none of the phones lost service, and, moreover, none of the results showed any impact on the speed test or the latency measurement. For the RSRQ, a more detailed impact analysis was performed. First, the measurements were grouped by GPS location by hexagonal shapes using H3 [1] with a mean edge length of around 28m. From each hexagon, the maximum RSRQ per cell was calculated and compared between DJ on and off. Across all hexagons, there is a median reduction in RSRQ of -0.13dB (SD 3.06dB). Limiting ourselves to four hexagons closest to the area (where we also performed speed tests), we saw that in two locations, the median RSRQ *improved* by 0.25dB (SD 1.5dB) and 0.34dB (SD 1.25dB), and in the other two it worsened by -0.46dB (SD 1.54dB) and -0.3dB (SD 1.30dB).

Due to the low median reduction in RSRQ and their high flucations indicating that the reduction might also be due to slight measurement errors, as well as no impact on connectivity, latency, or speed, we can conclude that in our deployment, there is no impact on outside users by the downlink jamming.

6.3.2 Protocol-Level Attack

The only case where an outside user is impacted by the protocol-level attack is when the localization system misclassifies the origin of the connection as an inside connection. In our deployment, the localization has a false positive rate of 0.28% measured over more than 10,000 connections from outside of the controlled area. These connections were done using a testing rig which can produce large number of new connections deployed in a backpack.

Moreover, most modern phones retry a connection even if the attack is successful multiple times. Therefore, the impact for outside users is even lower than the reported 0.28%.

6.4 Comparison to Classical Jamming

Previously, the controlled area in our deployment utilized a classical jamming system. This area was selected for deploying the GLaDoS prototype due to the advantages of this novel system, as outlined in the Introduction. Some observations

highlight the benefits of GLaDoS, even in its current prototype stage. Experience from our industry partner for the classical jamming showed that in most locations we are no worse than the classical jamming approach. However, in Location 11 (see Table 5), we achieved perfect coverage with GLaDoS, where the classical jamming experienced coverage issues.

We must note that our system has significant potential for further improvement as the system matures, thus widening the gap between classical jamming and a combined approach even further. Although the classical jamming approach served its purpose in the past, the GLaDoS prototype has shown promising results, achieving a 99.3% effectiveness rate while improving coverage at some locations. These findings demonstrate that even this early version performs well and suggest there is significant potential for further improvement as the system matures.

7 Related Work

Beyond simplistic noise jamming, efficient approaches to achieve DoS have been identified in [19]. Meanwhile, protocol-level attacks against LTE/5G NSA networks have been widely studied in the context of fake base stations and relays [26–28]. These attacks were later improved by using overshadowing techniques [20, 21, 30], yet still suffer from the same coverage/overspill issues as traditional jamming.

The first paper to introduce uplink overshadowing attacks in LTE networks was AdaptOver [11]. The authors target UEs to attack based on the TMSI sent in the contention resolution ID. This is sufficient to show the feasibility of attacks on availability by attacking the researcher's phone; however, such a technique does not address restricting communication in a specific area, as we do here.

LTrack [18] is a system that combines localization with downlink overshadowing attacks. This work conceptually explored how to track a large number of users using IMSI-extracting messages and a localization system. However, the targeting is based on an IMSI-TMSI database, not a predefined area. It also does not prevent communication. Furthermore, while both components (localization and overshadowing) were tested, the system as a whole was not implemented. The work of Oh et al. [21] also explored the combination of localization and downlink and uplink overshadowing attacks to find Uncooperative Cellular Devices. The RNTI of an Uncooperative Cellular Device is first found with silent SMSs, after which overshadowing attacks are used to induce the eNodeB to schedule extensive uplink traffic of the UE.

Although these papers use some combination of localization and overshadowing attacks, the similarity with our system is minimal. In our work, we use localization to protect a defined area and only attack connections that are inside it. The performance requirement for our system is therefore much greater. Moreover, compared to previous works, we show the system running in a real world deployment handling all nearby cells across multiple operators.

Monitoring of base stations has been done in multiple academic systems [7, 12, 15, 18]. However, to the best of our knowledge, the only academic work that attempted to monitor more than a single base station is that of Ross and Reaves [25]. This work showed a cell-hopping system that can switch between cells while keeping sync – however, only one cell is monitored at a time, with any other traffic being missed. In contrast, we describe a system capable of fully tracking multiple cells and operators simultaneously.

8 Conclusion

In today's world, there is still a large need for jamming systems. In this paper, we demonstrate that traditional jamming systems can be replaced by a new generation of systems based on protocol-level attacks. Our system minimizes the perturbation of the jamming signal in neighboring areas, significantly reduces exposure to radio signals, and filters out emergency calls, ensuring that access to these services is maintained. This is the first protocol-level system of its kind at this scale. It monitors and performs attacks on all the cells in the vicinity of the controlled area across all frequency bands for all operators. We evaluated its performance at 99.3% in a real-world deployment. To further validate our system, we investigated every connection that our system failed to attack.

Although GLaDoS is limited to LTE, in many countries the rollout of 5G is currently limited to the non-standalone variant, that means that all initial connection establishment is running via LTE and thus covered by GLaDoS. We expect that GLaDoS will be even more relevant for 5G-SA, as traditional jamming would require even more output power to sufficiently block communication due to beamforming and similar techniques. However, porting GLaDoS to 5G-SA will pose many challenges, such as handling higher bandwidths, a greatly reduced reaction time to uplink allocations, as well as an increased variance of possible configurations on all layers that need to be implemented and tested.

In conclusion, GLaDoS is the first real-world-ready cellular communication control system based on overshadowing attacks.

Acknowledgements

The authors thank Marc Röschlin and Patrick Leu for the original idea to combine low-power downlink jamming with a protocol-level approach. We also appreciate the practical RF interface and propagation related insights, general support and critical review provided by Oliver Bosshard.

Ethics Considerations

Running a system that blocks cell-phone communication in controlled areas is essential for public safety. This work improves upon classical noise jamming and makes them more efficient and selective, such that neither neighbors nor people in the controlled area suffer from the negative consequences of classical noise jamming. Especially since emergency calls are always allowed both from within and around the controlled area while still denying uncontrolled access to the cell network, our system increases the safety of the general public.

Our work is the result of a collaboration between university and an industry partner, who in turn had all required permissions from the necessary authorities. Before deployment, individual components of the system were rigorously tested in labs, also in collaboration with operators. We showed that there is no impact on connections not attacked by our system, that there is an adequate performance of the localization system used by this system and that emergency calls are always allowed. Based on these results, the government approved to operate our prototype system in the controlled area where the use of a phone is prohibited for a limited duration. Following that, we performed an evaluation of the system with our own phones.

Data Collection

Our system does not rely on the collection of mobile phone identifiers such as IMSI / TMSI, and we were careful to make sure that no such Personally Identifiable Information (PII) identifiers are recorded/stored in our system, neither within or outside of the controlled area:

- On the paging channel, operators transmit paging messages containing TMSIs, and in some cases IMSIs. We do not decode the paging channel and therefore our system does not record these PIIs.
- In RRC Connection Request, during Service Request procedure, the UE sends its previously allocated TMSI, but these are not collected. If there is no such previously allocated TMSI, as is the case during an Attach Request procedure, it sends a random 40 bit identifier. We designed our experiments such that the phones always use the Attach Request procedure. Thus, we only need to collect, store, and filter these random identifiers to match the connections to our own phones. This random identifier is generally not considered as PII, and is recorded purely for the use case of debugging of our system with our UEs.
- In certain conditions, the UE sends its IMSI or TMSI in an Attach Request on the uplink. Our system does not decode any messages on the uplink; therefore these PIIs are not going to be recorded by our system.
- The localization solution that we use does not collect IMSI or TMSI but relies on RNTIs to identify the connection. RNTIs are randomly assigned per connection and are not considered PII.

Compliance with Open Research

In the jurisdiction of this work, the system as a whole is classified as dual-use and is thus subject to export control laws, prohibiting the distribution of the code to the public. Additionally, the project is currently the subject of a pending intellectual property application. Unfortunately, these constraints prevent us from making the code publicly available. Nevertheless, we hope that the detailed descriptions and results provided in the paper will aid researchers in advancing this field further.

References

[1] uber/h3, January 2025. original-date: 2017-12-21T01:38:35Z. URL: https://github.com/uber/h3.

[2] 3GPP TS 36.101. 3GPP TS 36.101 V16.20.0, May 2024. URL: https://www.etsi.org/deliver/etsi_ts/136100_136199/136101/16.20.00_60/ts_136101v162000p.pdf.

[3] 3GPP TS 36.331. 3GPP TS 36.331 V18.3.1, October 2024. URL: https://www.etsi.org/deliver/etsi_ts/136300_136399/136331/18.03.01_60/ts_136331v180301p.pdf.

[4] 3GPP TS 24.301. 3GPP TS 24.301 V16.9.0, September 2023. URL: https://www.etsi.org/deliver/etsi_ts/124300_124399/124301/16.09.00_60/ts_124301v160900p.pdf.

[5] B Alberth, M Birchler, N Natarajan, and D Roberson. Contraband Phone Solution Assessments: Comparison of the Use of Managed Access v. Precision Jamming Systems in Correctional Facilities. page 25, February 2018. URL: https://s3.documentcloud.org/documents/5204322/2-2-18-T-Mobile-Contraband-UEs-RAA-Final.pdf.

[6] M. Borgerding. Turning overlap-save into a multiband mixing, downsampling filter bank. *IEEE Signal Processing Magazine*, 23(2):158–161, March 2006. Conference Name: IEEE Signal Processing Magazine. URL: https://ieeexplore.ieee.org/abstract/document/1598092, doi:10.1109/MSP.2006.1598092.

[7] 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, pages 25–30, New York, NY, USA, 2016. Association for Computing Machinery. URL: https://dl.acm.org/doi/10.1145/2980055.2980057, doi:10.1145/2980055.2980057.

[8] Caconde. Nearby jail's signal jammer affecting cellphone and devices using 2.4GHz band, November 2020. URL: https://electronics.stackexchange.com/q/531972.

[9] Federal Communications Commission. Jammer Enforcement, April 2020. URL: https://www.fcc.gov/general/jammer-enforcement.

[10] Nokia Corporation. Radio access networks | Nokia, 2025. URL: https://www.nokia.com/networks/radio-access-networks/.

[11] Simon Erni, Martin Kotuliak, Patrick Leu, Marc Roeschlin, and Srdjan Capkun. AdaptOver: adaptive overshadowing attacks in cellular networks. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, pages 743–755, Sydney NSW Australia, October 2022. ACM. URL: https://dl.acm.org/doi/10.1145/3495243.3560525, doi:10.1145/3495243.3560525.

[12] 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)*, pages 1–7, December 2019. ISSN: 2576-6813. URL: https://ieeexplore.ieee.org/document/9014096, doi:10.1109/GLOBECOM38437.2019.9014096.

[13] Eric Grommon, Jeremy G Carter, Fred Frantz, and Phil Harris. A Case Study of Mississippi State Penitentiary's Managed Access Technology. August 2015. URL: https://www.ojp.gov/pdffiles1/nij/grants/250262.pdf.

[14] Ericsson Group. Ericsson Radio System, 2025. URL: https://www.ericsson.com/en/portfolio/networks/ericsson-radio-system.

[15] Tuan Dinh Hoang, CheolJun Park, Mincheol Son, Taekkyung Oh, Sangwook Bae, Junho Ahn, BeomSeok Oh, and Yongdae Kim. LTESniffer: An Open-source LTE Downlink/Uplink Eavesdropper. In *Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '23, pages 43–48, New York, NY, USA, June 2023. Association for Computing Machinery. doi:10.1145/3558482.3590196.

[16] Ltd. Huawei Technologies Co. Super RAN, Top Gear, 2025. URL: https://carrier.huawei.com/en/spotlight/5g-ran/.

[17] Hermann Kopetz. *Real-Time Systems*. Real-Time Systems Series. Springer US, Boston, MA, 2011. URL: http://link.springer.com/10.1007/978-1-4419-8237-7, doi:10.1007/978-1-4419-8237-7.

[18] Martin Kotuliak, Simon Erni, Patrick Leu, Marc Röschlin, and Srdjan Čapkun. {LTrack}: Stealthy Tracking of Mobile Phones in {LTE}. pages 1291–1306, 2022. URL: https://www.usenix.org/conference/usenixsecurity22/presentation/kotuliak.

[19] 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*, 54(4):54–61, April 2016. Conference Name: IEEE Communications Magazine. doi:10.1109/MCOM.2016.7452266.

[20] Norbert Ludant and Guevara Noubir. SigUnder: a stealthy 5G low power attack and defenses. In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '21, pages 250–260, New York, NY, USA, June 2021. Association for Computing Machinery. doi:10.1145/3448300.3467817.

[21] Taekkyung Oh, Sangwook Bae, Junho Ahn, Yonghwa Lee, Tuan Dinh Hoang, Min Suk Kang, Nils Ole Tippenhauer, and Yongdae Kim. Enabling Physical Localization of Uncooperative Cellular Devices. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, ACM MobiCom '24, pages 1530–1544, New York, NY, USA, 2024. Association for Computing Machinery. URL: https://dl.acm.org/doi/10.1145/3636534.3690709, doi:10.1145/3636534.3690709.

[22] Cheol Jun Park and Mincheol Son. SigOver + alpha, 2019. Series: 36C3: Resource Exhaustion Published: Chaos Computer Club e.V. doi:10.5446/53158.

[23] M. E. Parsanezhad, S. M. J. Mortazavi, T. Doohandeh, B. Namavar Jahromi, H. Mozdarani, A. Zarei, M. Davari, S. Amjadi, A. Soleimani, and M. Haghani. Exposure to radiofrequency radiation emitted from mobile phone jammers adversely affects the quality of human sperm. *International Journal of Radiation Research*, 15(1):63–70, January 2017. Publisher: International Journal of Radiation Research. URL: http://ijrr.com/article-1-1887-en.html.

[24] A. Rafati, S. Rahimi, A. Talebi, A. Soleimani, M. Haghani, and S. M. J. Mortazavi. Exposure to Radiofrequency Radiation Emitted from Common Mobile Phone Jammers Alters the Pattern of Muscle Contractions: an Animal Model Study. *Journal of Biomedical Physics & Engineering*, 5(3):133–142, September 2015. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC4576874/.

[25] Alexander J. Ross and Bradley Reaves. Towards Simultaneous Attacks on Multiple Cellular Networks. In *2023 IEEE Security and Privacy Workshops (SPW)*, pages 394–405, May 2023. ISSN: 2770-8411. URL: https://ieeexplore.ieee.org/document/10188622, doi:10.1109/SPW59333.2023.00040.

[26] David Rupprecht, Katharina Kohls, Thorsten Holz, and Christina Poepper. IMP4GT: IMPersonation Attacks in 4G NeTworks. In *Proceedings 2020 Network and Distributed System Security Symposium*, San Diego, CA, 2020. Internet Society. URL: https://www.ndss-symposium.org/wp-content/uploads/2020/02/24283.pdf, doi:10.14722/ndss.2020.24283.

[27] 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 *Proceedings 2016 Network and Distributed System Security Symposium*, page 15, San Diego, CA, 2016. Internet Society. doi:10.14722/ndss.2016.23236.

[28] Altaf Shaik, Ravishankar Borgaonkar, Shinjo Park, and Jean-Pierre Seifert. 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*, pages 221–231, Miami Florida, May 2019. ACM. URL: https://dl.acm.org/doi/10.1145/3317549.3319728, doi:10.1145/3317549.3319728.

[29] Inc. The MathWorks. Downlink reference measurement channel configuration - MATLAB lteRMCDL - MathWorks Schweiz. URL: https://ch.mathworks.com/help/lte/ref/ltermcdl.html.

[30] 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)*, pages 55–72, Santa Clara, CA, August 2019. USENIX Association. URL: https://www.usenix.org/conference/usenixsecurity19/presentation/yang-hojoon.

A Multi-Cell Processing Implementation

A.1 Software Defined Radio (SDR) Adapter

Our SDR can handle two distinct frequency bands at the same time. Ports 1 and 2 are tuned to one frequency band, whereas ports 3 and 4 receive a different band. The SDR adapter transports these four bidirectional IQ streams (one per port) to and from the SDR via four direct memory access (DMA) processes over PCIe from / to the host memory. The samples are int16 IQ samples, as also the transceiver of the SDR operates with a 16 bit ADC/DAC. Before/After the DMA, they are converted between int16 for the SDR and float32, which the rest of the processing uses.

The current timestamp for both Rx and Tx is calculated as follows. First, upon startup, the SDR is not yet exchanging any samples with the host. The DMA transfers only start once the 1 PPS signal connected to the SDR (see Section 5.2.5 for details) turns high, i.e., precisely at the start of a full second. Other than that, the SDR has no notion of time. To align the sample timestamp globally across the whole system, upon reception of the first samples, we get the current time from the host, which is synchronized with NTP, and round to the next full second. This represents the absolute time of the first sample. A counter on the host then counts how many DMA buffers (of fixed size) have been exchanged with the SDR and can thus determine the exact amount (and thus timestamp) of samples since the first PPS pulse.

The four DMA processes are executed in a loop in a separate radio thread each. The loop executes the following three steps. First, it determines the timestamp of the received samples and the samples to transmit next. Next, it executes the channelizer and distribution step explained in Section A.2 and Section A.3. Finally, it collects the amount of samples to be sent from the Tx combiner and prepares them for sending. Put together, these steps allow the individual cell instances to operate independently from each other and as if they were using dedicated separate SDRs each.

A.2 Downlink Reception Channelization

To channelize the downlink, we implemented the channelizer from [6], which uses the overlap-save method. First, we convert the input signal into frequency domain using an FFT. Note that we can reuse the output of this rather expensive FFT operation for every operator channel. Next, we apply an operator-specific bandpass filter such that other operator signals are attenuated before applying a coarse frequency shift using rotation of the FFT bins and downsampling the signal to 30.72Msps. Finally, we apply the inverse FFT operation again and apply a fine-grained frequency offset correction. We optimize the two parameters of the channelizer, filter size P, and number of input samples in the overlap-save loop L, for the lowest processing time using a grid search. The best values,

$P = 256$ and $L = 1792$, take 303 microseconds to channelize 1 ms worth of samples. This is enough time to meet the latency and performance requirements.

A.3 Distribution

After channelization, the samples are distributed to each cell instance. This works by placing a reference to the memory location (a `shared_ptr`) of the channelized IQ samples in a lockfree receive queue, which exists per cell. The receive queue keeps track of the amount of samples in the queue, and the last requested sample time of the cell instance. When the cell instance requests a batch of samples (most of the time, 1ms, except during initial synchronization), the queue copies all corresponding samples to the sample buffer of the cell and removes entries which were consumed fully.

Since the samples are shared between multiple cells operating on the same frequency, when the last of the cells releases the `shared_ptr`, the memory is released into a memory pool used by the channelizer again, such that no memory allocations have to be done during runtime.

Also, care must be taken here to handle overflow conditions, that is, when the cell instance is not fast enough to keep up with the rate of the IQ stream. We detect such a condition by calculating the amount of pending samples in each receiver queue. Whenever there are more than 2ms worth of samples in the queue, the distribution step clears the queue and marks it as "overflowed" until the cell instance is able to catch up again. Extra care is taken such that in this case, the gap between received samples is in 1ms increments, otherwise the time synchronization of the cell is disturbed unnecessarily.

A.4 Uplink Transmission Combination

If the cell instance (or one of its UEs) wants to send something, for example, a PUSCH transmission to overshadow a UE or a PRACH preamble, it generates uplink samples with a 122.88Msps sampling rate, but places the signals at the middle of the frequency band. To align the transmission with the frequency position of the cell within the band, the Tx combiner applies a frequency offset correction and shifts the samples to the correct operator uplink frequency. Then it places them in a queue with the specified transmission time as shown in Figure 3.

The radio thread operates in a loop. At every iteration, the combiner is asked for samples that are ready to transmit in a given window. The combiner then checks every transmit queue for samples that have a Tx timestamp that is within that transmission window. If there are no samples within that window, zeros are returned; if there are samples from exactly one buffer, they are simply copied over. If there are samples from more than one buffers to be sent, the overlapping samples are summed up in the time domain before being returned. During the experiment conducted in Section 6.2 we had 0.24% of all transmissions overlapping. If the maximum absolute value of any sample after addition exceeds 1, all the sample values are scaled such that the maximum absolute value is less than 1 to avoid saturation of the transmitter.

B Downlink Data Decoding

All DCIs that were found with downlink grants are then passed to the PDSCH Decoder, which decodes the PDSCH symbols to bits. Then, the message bits are decoded through 3GPP standard compliant decoders of the various layers. If the message is a broadcast message (e.g., a RAR, or SIB), it is decoded using the radio resource control (RRC) decoder. If the message is sent to a UE, the system passes the bits through the medium access control (MAC), radio link control (RLC), packet data convergence protocol (PDCP), RRC, and optionally non-access stratum (NAS) decoders to determine the relevant higher-layer protocol messages.

B.1 Attack Success Determination

In the following, we will show both an example of a PDSCH message being decoded up to the NAS layer and how the system determines whether or not an attack was successful. In general, the processing follows the 3GPP standards.

For example, a `RRC Connection Setup` message for a UE is first passed through the MAC layer. There, it is determined that the message belongs to a logical channel ID (LCID) 0, corresponding to the common control channel (CCCH). That means it is not further encoded and can be decoded with the CCCH RRC decoder. A `NAS Attach Reject` message would carry an LCID of, e.g., 1, indicating the specific RLC processing necessary that was specified in the `RRC Connection Setup` message where the data *bearer* corresponding to the LCID was setup and specified. After reassembly of all RLC segments, it is passed on to the PDCP layer, which just removes its header and integrity field. There, the RRC decoder decodes the dedicated control channel (DCCH) and finds a downlink information transfer message, containing an embedded NAS message. Finally, the NAS message can be decoded by the ASN.1 decoder and the message handled appropriately by the UE - in this case, mark the attack as successful since the `NAS Attach Reject` indicates to the UE a termination of the connection. If the message instead would be a `NAS Attach Accept` or `NAS Security Command`, etc. message, it would indicate that the original `NAS Attach Request` or `NAS Service Request` message sent by the UE has passed through to the eNodeB and that the attack has failed.

C Low-Latency Considerations

To comply with the latency constraints set out in Section 4.7, we implemented the following measures. We use a Linux operating system (OS) running on a host with activated simultaneous multithreading (SMT) and a few configuration changes. First, we fixed each thread to a specific CPU core using *CPU pinning* in order to minimize overhead in relocating threads between executions and to have a reasonable guarantee that the thread will be able to execute immediately once the data to process has arrived. Next, we used *CPU Isolation* to assign almost all CPU cores to running only our software and nothing else, leaving one physical core (i.e., two logical cores) for the OS and applications such as logging, SSH, etc. Next, we do not want the CPU frequency to scale down in order to conserve power, thus we apply the *performance* mode of the CPU governor. Finally, we schedule all interrupts to be handled by non-isolated cores by disabling the `irqbalance` daemon and setting the `irqaffinity` kernel parameter. Using an open-source tool which measures the number of times a thread gets interrupted and how long, we verified the impact of our configuration changes. On isolated cores, we get a maximum latency between iterations of around $15\mu s$, while on non-isolated cores this rises to more than $2.5ms$, which is clearly unsuitable for our real-time needs, but enough for any housekeeping tasks. We also evaluated the use of a real-time kernel, but this did not yield further improvements.

C.1 Application Layer Parallel Processing

At the application layer, we extensively optimized and benchmarked all parts of the system (e.g., the channelizer, downlink decoder, and uplink encoder) and are also measuring for every uplink transmission how far in advance we managed to generate and schedule it to meet the firm real-time constraints, as shown in Section 6.1. Still, this alone is not enough because we are attacking many UEs on many cells and operators at the same time and thus need to use parallel processing.

In our approach, we are splitting up the tasks across available cores as follows. First, we dedicate four logical cores for the radio management to copy samples back and forth and to perform channelization. Next, we dedicate two logical cores for the message passing over ZMQ, one each for the receiver and transmitter loop. Furthermore, we assign one logical core to perform cell searching continuously and two to the OS and all remaining non-latency-critical tasks (e.g., logging, observability).

Finally, all remaining logical cores are assigned exactly one cell instance each. This stems from the fact that in our implementation one core is fast enough to process the downlink of one subframe of a cell for several UEs in much less than 1 ms (the duration of a subframe) in most cases. This improvement allowed us to vastly simplify our previously explored approaches, e.g., round-robin assigning the decoding of a subframe to a thread pool (since one was taking longer than 1ms) and the inherent locking complexity arising from such approaches.

Also, we opted against scheduling more than one cell on a core, since we observed that across several operators, subframes are aligned in time for all their cells. That means during a 1ms interval, most cell-cores are busy at the same time decoding the downlink. Any prolonged interruption or waiting for the core to become available would then directly hurt the latency and thus stability of the system. Still, since all these cores remain idle after that and wait for the samples of the next subframe to arrive, we can still use them during that time for the final task - the encoding of the uplink. To this end, we create a thread pool, and set the CPU affinity of each thread in the pool to all cores assigned to cell decoding. Thus, whenever an PUSCH message needs to be encoded to IQ samples, which happens always *after* the downlink decoding is finished, we submit a task to the thread pool to encode the uplink. At the very least, the core / cell that just submitted the task is also immediately available to encode the uplink, incurring a slight overhead. However, in the case that many uplink transmissions need to be encoded, one core alone would be too slow. In this case, the encoding task is taken up by another free core (as decided by the Linux scheduler).