LeopardSeal: Detecting Call Interception via Audio Rogue Base Stations

Christian Peeters[¶], Tyler Tucker[¶], Anushri Jain, Kevin Butler, Patrick Traynor
{cpeeters, tylertucker1, ajain1, butler, traynor}@ufl.edu
University of Florida

Abstract

Audio Rogue Base Stations (ARBSs) allow an adversary to intercept cellular calls. These devices represent a substantial escalation in the threat posed by traditional rogue base stations, which only collect device identity information. This paper presents the first technique for detecting call eavesdropping via an ARBS. Our system, which we call LeopardSeal, uses distance bounding over the call audio channel to determine whether or not extra wireless hops (and therefore increased audio delay) that are characteristic of ARBSs are present during a call. We implement a proof of concept ARBS using open-source guides and perform a measurement study across the United States. We demonstrate the ability to detect all attacks (with zero false positives) due to a statistically significant difference in round trip times between benign and attack call audio (i.e., t-test: $p \ll 0.01$) due to the large cost of additional wireless hops. Through this effort, we demonstrate the ability to robustly detect these eavesdropping devices.

CCS Concepts

• **Security and privacy → Mobile and wireless security**.

Keywords

audio rogue base stations, cellular networks, cellular security, distance bounding

ACM Reference Format:
Christian Peeters[¶], Tyler Tucker[¶], Anushri Jain, Kevin Butler, Patrick Traynor, {cpeeters, tylertucker1, ajain1, butler, traynor}@ufl.edu, University of Florida. 2023. LeopardSeal: Detecting Call Interception via Audio Rogue Base Stations. In *ACM International Conference on Mobile Systems, Applications, and Services (MobiSys '23), June 18–22, 2023, Helsinki, Finland.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3581791.3596846

1 Introduction

In cellular networks, IMSI Catchers allow an adversary to surreptitiously intercept device identifiers. These devices have allowed both law enforcement [30, 66] and unknown actors [47] to perform both presence testing and track user movement (potentially in real-time). Whether through exploiting weaknesses in older but still-deployed cellular standards, performing downgrade attacks from more secure standards, or even acquiring cryptographic keys directly from the network provider, these devices remain a substantial threat *even with modern network standards* [46].

While there has been substantial research on detecting IMSI Catchers, no attention has been given to the more capable Audio Rogue Base Stations (ARBSs). While both classes of devices are capable of capturing user identifiers known as International Mobile Subscriber Identities (IMSIs), ARBSs are also capable of intercepting and eavesdropping on-call audio. As such, this ability to wiretap arbitrarily represents a substantial increase in the potential threats to users. These extremely capable devices are already available for sale [13].

In this paper, we defend call confidentiality in the face of this more capable adversary with LeopardSeal. [1] Our approach takes advantage of a simple observation: an ARBS adds two extra wireless hops to every audio round trip, and the cost of this retransmission adds measurable delay to calls. Therefore, we create a distance-bounding solution for measuring RTT, which will be substantially increased in the presence of an ARBS. In so doing, we make the following contributions:

- **Detection of Call Interception by ARBSs:** No current security mechanisms can determine whether a call is being eavesdropped upon in real time. Creating such a detector is crucial in both unauthorized and authorized wiretapping incidents. Specifically, whereas the former is illegal, a crucial property of the latter is that it must be undetectable.
- **Identify Audio Artifact of ARBSs:** We identify a fixed increase in Mouth-to-Ear delay, caused by expensive transcoding costs related to having an additional wireless hop in each direction, as a signal of the presence of such devices. LeopardSeal was designed specifically to detect small changes in call path indicative of interceptions via an ARBS, which other applications of acoustic distance bounding have been unable to achieve [52].
- **Deploy, Bootstrap, and Measure:** We implement and deploy LeopardSeal, and discuss how our system can be bootstrapped in the real world. We then conduct testing in geographically diverse areas of the US to show a consistent attack delay of ≈ 360 ms using our constructed ARBS. We not only demonstrate the ability to detect all attacks but show that attack and benign traffic are distinct populations that are separated by ≈15 standard deviations. We further validate LeopardSeal's abilities and show that changes to our testbed do not impact detection accuracy.

MobiSys '23, June 18–22, 2023, Helsinki, Finland

ACM ISBN 979-8-4007-0110-8/23/06…$15.00
https://doi.org/10.1145/3581791.3596846

[¶]These two authors contributed equally.
[1]Leopard seals are predators of stingrays, one common name for commercial ARBSs.

Christian Peeters[¶], Tyler Tucker[¶], Anushri Jain, Kevin Butler, Patrick Traynor
{cpeeters, tylertucker1, ajain1, butler, traynor}@ufl.edu
University of Florida

The remainder of this paper is organized as follows: Section 2 provides critical background information necessary to understand these attacks and current defenses; Section 3 states our hypothesis; Section 4 details our threat model; Section 5 formally defines the audio-based distance bounding protocol that the LeopardSeal system uses; Section 6 presents the implementation of the LeopardSeal system and the experiments designed to evaluate it; Section 7 details the results of our experiments; Section 8 discusses related information; Section 9 highlights and contextualizes related research; and Section 10 offers concluding remarks.

2 Background

Cellular interceptor devices, which include IMSI Catchers and ARBSs, have been a longstanding problem in mobile telecommunications. These devices can be defined as malicious hardware capable of communicating on cellular frequencies as legitimate network entities to end devices or cellular networks. Though the ease of operating a cellular interceptor has decreased with the enhanced security practices of modern cellular standards, these devices have adapted and remain effective [18, 20, 46]. In this section, we explore the different types of cellular interceptor devices, how they can potentially exploit all generations of cellular network technologies, and discuss their impact on users.

2.1 Cellular Interceptor Devices

Cellular interceptor devices masquerade as a legitimate base station to establish a connection with end devices. The exact capabilities of these devices can vary based on the network technology and vulnerabilities they exploit. Cellular interceptors can be broken up into two separate groups: ARBSs and IMSI-catchers. [2]

2.1.1 Audio Rogue Base Stations ARBSs convince end devices that they are a legitimate cellular base station and capture all communication along a call path. While both types of cellular interceptor attempt to convince end devices they are a legitimate network entity, ARBSs possess the ability to store and forward cellular traffic between end devices and a legitimate network including call audio, hence the name Audio Rogue Base Station. In order to accomplish this, it is necessary for an ARBS to either complete or avoid the authentication process between an end device and the cellular network it wishes to communicate with. Further details on this process are provided later in this section.

The impact an ARBS has on cellular network traffic is shown in Figure 1. A traditional call path can be seen in Figure 1(a). In this scenario, an end device connects to a local base station which forwards data through the core cellular network (PSTN) to the base station nearest to the intended recipient. An ARBS attack is pictured in Figure 1(b), which begins when a target end device, instead, connects to an ARBS. The ARBS advertises itself as a part of the local network and transmits at a high power to entice nearby phones to favor it over other towers. The ARBS also establishes a connection to a legitimate base station in the role of an end device, pretending to be the target user. The ARBS then forwards all cellular

[2]IMSI-Catchers and ARBSs are sometimes called Rogue Base Stations (RBSs). To avoid confusion, we use only IMSI-Catchers and ARBSs and clearly define attack capabilities in this section.

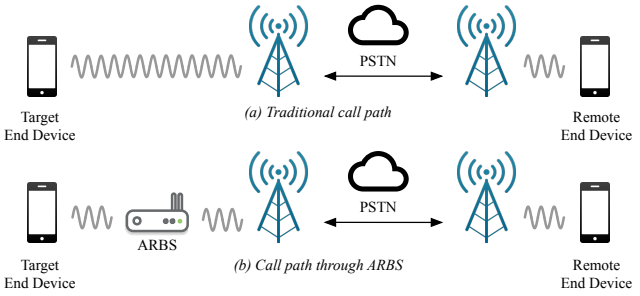

Figure 1: This network diagram compares a standard call via the telephony network to one intercepted by an ARBS.

network activity (e.g., call audio, SMS, data, location [46]) between the target end device and the legitimate network.

ARBSs were long dismissed by network providers due to the cost. However, these devices can now be constructed for as little as $1000 [6]. ARBSs are also frequently used by law enforcement and government entities [30, 34, 37]. Commercial ARBSs (e.g., Stingrays) are sold only under special agreements and are *not* available to the general public. However, information about them has surfaced over time and has shown that they are capable of operating in both 2G and modern 3G/4G networks [17].

2.1.2 IMSI-Catchers The second group of cellular interceptors, commonly referred to as "IMSI-Catchers" [61] complete their attack prior to authenticating with end devices. IMSI-Catchers obtain unique and static International Mobile Subscriber Identity (IMSI) values for cellular end devices by exploiting vulnerabilities in the various authentication protocols across all generations of cellular technologies. This has been accomplished via techniques such as forcing end devices to expose their IMSI values prior to authentication or via vulnerabilities in the handover process [55, 65]. While IMSI-Catchers are an ongoing security concern, they are not the focus of this research.

2.2 Network Interception

An adversary can intercept a cellular network communication in many different ways, but the fundamental concepts of the attack remain the same. Cellular interceptor devices exploit known vulnerabilities in the authentication process between end devices and cellular networks. This idea holds true for all cellular network technology generations and has shown to be problematic regardless of the enhanced security features provided by modern networks [20, 23, 27, 35, 53].

Achieving the goal of appearing as a legitimate base station to both end devices and networks varies depending on the generation of cellular network technology for which the attack is conducted. In 2G cellular networks, this is a trivial task, as only one-way authentication is used, allowing any device capable of broadcasting on 2G frequencies to act as a base station. This vulnerability led to the introduction of the AKA protocol in 3G networks to provide mutual authentication between end devices and a serving cellular network. 4G and 5G networks make use of extensions of the 3G AKA

protocol, with the main difference being the use of a key hierarchy for improved key management within the network core [7, 59].

The introduction of the AKA protocol with modern cellular networks increased the difficulty of performing an ARBS attack. However, one of the most frequently used strategies to defeat this is to avoid the AKA protocols entirely. In what is commonly referred to as a "Step-Down Attack" [45, 64], an ARBS can force devices in an area to fall back to 2G networks by jamming the control channel frequencies of newer network protocols. End devices commonly provide backward compatibility with legacy generations in their baseband processors. This allows users to have cellular access when modern cellular service is not available. By forcing connection establishment via 2G, an ARBS can avoid the AKA protocol entirely.

Despite the eventual sunset of 2G networks, ARBSs will remain an issue. An ARBS can gain access to AKA authentication vectors through various methods depending on network technology and prerequisite abilities an adversary might have. An outside attacker compromising the network core has previously been demonstrated in the past [43, 54], and in some cases has resulted in the installation of malware within the network itself. If an adversary can acquire authentication vectors by these or similar means, they can complete AKA protocols using an ARBS. Moreover, an attack via an ARBS is likely to happen at the nearest tower, which can provide an adversary with all necessary keys. The challenge of obtaining network resources can also be overcome with the help of insider access.

In addition to compromises by malicious attackers, authentication keys may also be obtained by law enforcement and government entities in cooperation with the providers. This idea has been confirmed in the past where government groups and law enforcement will get a court-issued warrant to access cellular data to which the cellular providers need to provide access to protected network resources [39, 54, 66]. With access to authentication vectors being provided by the network, law enforcement and government entities can complete the AKA protocols. The impacts of this work on lawful use cases of cellular interceptors are discussed in Section 8.

2.3 Law Enforcement

Law enforcement and government agencies can legally obtain access to telecommunications resources to perform a lawful interception of calls and other data. The methods by which lawful interceptions are executed can vary. Some of which, referred to as "traditional wiretaps", allow interception from *within the network core* through CALEA interfaces [58] and are designed to be undetectable [3]. This approach avoids the addition of the wireless hop and consequently avoids detection by LeopardSeal. Such an approach, however, requires insider access by telecommunications providers and is *not accessible to the general public* [54].

Additionally, legal wiretapping requires judicial approval in the United States whereas laws on fake base stations are inconsistent based on location and often unclear [33, 44]. LeopardSeal is therefore capable of detecting lawful interception via the use of ARBSs (e.g., mounted on the top of an SUV with no wired connection to a network) but does not defend against traditional (i.e., no-hop) lawful interception via wiretaps.

Law enforcement would choose the former case when eavesdropping on traffic from previously unknown devices at a specific location, for example during a protest or large public event [44]. In this setting, the ARBS would be mobile, relying on an additional wireless hop to deliver victim traffic to a legitimate tower. This makes the ARBS vulnerable to LeopardSeal detection. With a reported 75 law enforcement agencies across 27 U.S. states using fake base stations.

3 Hypothesis

By adding an additional wireless hop to the path a call travels, there will be a measurable and significant increase in the round trip time (RTT) of the call audio. This increase in time is due to multiple components introduced by the addition of an ARBS, which together, result in a significant impact on RTT. Therefore, we can detect the presence of a rogue base station using prerequisite knowledge of the expected RTT for an approximate distance.

Similar to traditional IP-based Internet, cellular networks are subject to numerous forms of delay that all contribute to the overall time it takes to send and receive data packets. However, cellular networks actively manipulate packets and their contents (e.g., latency induced by codecs and digital-to-analog conversions). The concept of delay in regards to call audio is referred to as "Mouth-to-Ear Delay" [2], and more specifically, is the delay in call audio between when one party speaks and the other party hears it.

To convey the fundamental differences in latency between IP-based internet and cellular networks, we refer to the ITU G.114 standard guidelines, and the associated "E-Model" transmission rating algorithms. This shows that a one-way delay of 400ms is deemed satisfactory in regards to cellular networks, in contrast to traditional IP-based internet where 100ms would negatively impact a connection.

4 Security Model

The goal of LeopardSeal is to detect the presence of an additional wireless hop indicative of ARBS interception. We discuss attack assumptions and capabilities of an adversary using an ARBS.

4.1 Adversarial Capabilities

The adversary forwards all cellular traffic between a target device and a legitimate network. Calls that pass through the ARBS are unencrypted and thus, in addition to storing a copy of all messages, an adversary may also manipulate them. This includes fabricating, modifying, or dropping the audio used in LeopardSeal through the voice channel. This also may include injecting noise or additional sounds.

The adversary controls the ARBS, but would likely not be able to easily determine the location of the other remote end device prior to the call. However, in the event of a targeted attack on a specific call, an adversary could have acquired a remote end device's approximate location information. This could have been done via technical means such as phishing, or simply just guessing based on area code or having prerequisite knowledge. Consequently, we must assume that an adversary knows the expected latency for every call to account for targeted attacks.

Christian Peeters[¶], Tyler Tucker[¶], Anushri Jain, Kevin Butler, Patrick Traynor
{cpeeters, tylertucker1, ajain1, butler, traynor}@ufl.edu
University of Florida

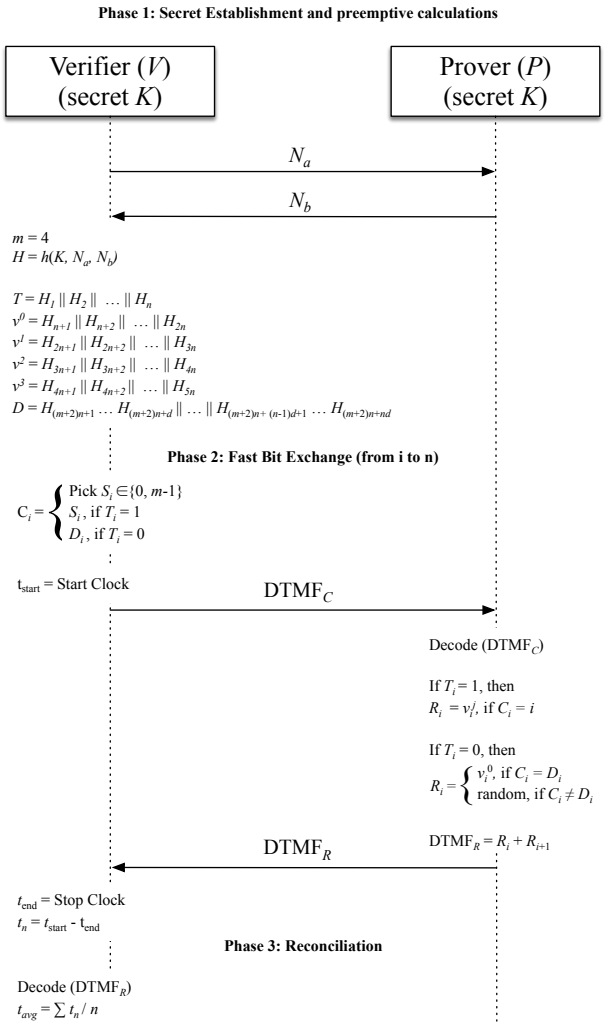

Figure 2: The LeopardSeal Protocol is a modified version of the KA2 distance bounding protocol that is applied to audio transmitted through telephony networks, encoding challenges as DTMF tones.

We assume end devices are trustworthy and not compromised with any malware that could potentially interfere with detection. Compromising an end device allows an attacker to accomplish everything possible with a rogue base station and more.

Finally, we assume that the adversary operates on the same cellular network as the local end device. By assuming this worst-case scenario, there will be minimal variations in the added call path by the ARBS and, in turn, the added RTT. In practice, this will not always hold and the latency added by the ARBS could be greater due to network delays.

5 LeopardSeal Protocol

In this section, we construct a distance bounding-inspired protocol that reflects our adversarial model. We then describe how our LeopardSeal protocol builds on prior work in distance bounding, and detail its construction. Finally, we analyze the performance and security guarantees against common distance bounding attacks.

5.1 Prior Work in Distance Bounding

Distance bounding protocols allow parties to determine a tight upper bound on the distance between one another by observing the travel time of messages. At a high level, a prover and a verifier rapidly exchange bits as challenges and responses while measuring the RTT of those messages.

Many distance bounding protocols offer various performance and security guarantees. Typically, these protocols are designed with the intent to be used for applications where a Prover and a Verifier are at a line-of-sight distance from one another [21, 63], and in some cases as close as physically touching [29]. Often, the messaging latency in these protocols is defined by physical constraints, such as the time needed for a wireless signal to propagate over a relatively short distance, allowing traditional distance bounding algorithms to provide centimeter-level accuracy.

Unlike traditional distance bounding, our attack scenario is on a much larger scale with a higher degree of variability in the messaging path, due to the multi-hop nature of this setting. Our approach is analogous to a secure version of *ping* designed for cellular networks. In a cellular network call, the number of hops between end devices is variable depending on network conditions. The network path information is not made available to end devices and there are no tools to obtain it. The LeopardSeal protocol that we develop must account for this variability while also providing an upper bound on RTT such that attacks are accurately identified.

We prioritize the ability to be resistant to impersonation fraud and mafia fraud over distance fraud and terrorist fraud since they do not apply to our threat model. Taking this into consideration, we look to well-known distance bounding protocols to use as a foundation for LeopardSeal. There are many possible options, and in order to find a construction that best meets our needs we specify several criteria that we look for in a protocol:

- **G1: Minimize Bits Exchanged** - Because the RTT values in LeopardSeal are much larger than those assumed by traditional distance bounding, the number of bits exchanged during the protocol greatly impacts the overall time for completion.
- **G2: Resilience to Impersonation and Mafia Fraud** - These attacks are possible under our adversarial model, so we prioritize protocols that are more effective against them.
- **G3: Minimize Computation** - It is imperative that our protocol is lightweight and could be run on an array of hardware. Many distance bounding protocols use techniques to improve security at a performance cost. This is largely due to a second slow phase after the challenge and response phase, which adds additional post-processing after exchanging data.

We looked to academic surveys on distance bounding protocols that compare security and performance [16], as well as studies on protocol verification [42] to ensure our protocol would have no known vulnerabilities.

This analysis showed that the ideal base protocol for LeopardSeal is the revised version of KA2 [38]. KA2 was designed for RFID applications, building on the Hancke and Kuhn (HK) protocol [31].

After carefully evaluating numerous distance bounding protocols, we found that the KA2 protocol best meets goals **G1**, **G2**, and **G3**. By expanding the alphabet, the revised KA2 protocol can further improve the resistance to mafia fraud.[3] Moreover, having an expanded alphabet allows us to reduce the number of bits sent. KA2 is also round-independent and does not have a second slow phase.

5.2 LeopardSeal Protocol Definition

Figure 2 provides a diagram of our LeopardSeal protocol for two participating end devices. These devices are referred to as a prover $\boldsymbol{P}$ and a verifier $\boldsymbol{V}$. The entities can be associated respectively with the target end device and the remote end device that we refer to throughout the paper. The LeopardSeal protocol we describe is a version of the KA2 protocol that is modified to work via the audio channel of calls. Additionally, our modification allows P to have more control of the protocol results.

KA2 offers variability in regards to $\boldsymbol{m}$, the size of individual challenges. The authors of KA2 suggest a value of four for m, which is what we choose as well. Analogous to the KA2 protocol, LeopardSeal consists of three stages: secret establishment and preemptive calculations, fast bit exchange, and reconciliation. The first phase of the protocol requires P and V to establish a shared resource $\boldsymbol{K}$. The method by which this happens can vary, however, we suggest that TLS should be used if available. If no IP-based communication is possible, an alternative option is to use a system of secure key establishment via the audio channel of a call, such as Authloop [56]. We leave the final decision up to an entity deploying LeopardSeal.

The LeopardSeal protocol implements the KA2 algorithm over an audio channel, so data must be exchanged in-band. To achieve this, we employ DTMF tones, which are standard in-band signals that operate as base 16 values. Note that in Figure 2, all messages that are exchanged between V and P after key and nonce establishment are DTMF messages. We provide a more in-depth discussion of the KA2 protocol in our appendix.

Once both P and V have K, they each select a nonce ($\boldsymbol{N_P}$ and $\boldsymbol{N_V}$ respectively) and exchange them via the same channel used to establish K. Once N_P and N_V have been exchanged, P and V generate a sequence of bits $\boldsymbol{H}$ of length $(2+m)n$, where $\boldsymbol{n}$ is the number of challenges to be sent, using a pseudorandom function (a MAC or hash algorithm) given K, N_P, and N_V as inputs. For our case of $m = 4$, H is then divided into six separate sequences of length n each: pre-defined m-ary challenges $\boldsymbol{D}$, random binary values $\boldsymbol{T}$, and four binary sequences $\boldsymbol{v_0}$, $\boldsymbol{v_1}$, $\boldsymbol{v_2}$, and $\boldsymbol{v_3}$. Once these sequences are generated, the first stage of the protocol is complete.

5.3 Protocol Performance

In addition to the selection of the m value, there are other design considerations for LeopardSeal that need to be determined for real-world deployment. The n value is the number of challenges and responses used in the fast bit exchange phase. The number of challenges and responses is a trade-off between performance and security, so the authors of KA2 provide the success probability of fraud given n and m [38]. The value of n can be adjusted to meet the security and performance needs of a deploying system, and the following describes how n impacts LeopardSeal.

[3]Though KA2 does have slightly weaker security than HK in regards to distance fraud in some cases, this is a beneficial trade-off due to the irrelevance of distance fraud in our threat model.

Number of Rounds	Number of Challenges (n)	Impersonation Fraud Success Rate	Mafia Fraud Success Rate	Execution Time (s)
1	2	6.25%	19.14%	2.82
2	4	0.39%	3.66%	5.64
4	8	1.53E-03%	0.13%	11.28
8	16	2.33E-08%	1.80E-04%	22.56
16	32	5.42E-18%	3.25E-10%	45.12

Table 1: This table provides the measures of performance and security for the LeopardSeal protocol when the size in bits of each individual challenge $m = 4$.

The security offered by LeoardSeal is identical to that of KA2. In regards to an adversary attempting Impersonation Fraud on the LeopardSeal protocol, they would need to correctly guess all challenges. This yields a success probability of $(1/4)^n$ when $m = 4$. If an adversary instead attempts mafia fraud, the success probability is subject to the balance of bit values in the random sequence T calculated during the second phase of the protocol. As a result, the probability ranges between the best-case scenario of $(7/16)^n$ and the worst-case of $(3/4)^n$.

We model the completion time of LeopardSeal with respect to the number of rounds with the equation:

$$t_c = 2n(t_{nd} + t_{dtmf} + t_p)$$

where t_{nd} is the one-way network delay, t_{dtmf} is standard DTMF length of 90ms, and t_p is the standard pause time following a DTMF tone of 65ms [1]. *In our atypical distance bounding setting, we will observe a significant decrease in performance as n increases.* Though one-way network time can vary, especially in the case of an ARBS attack, for performance analysis we assume a worst-case scenario according to the E-Model [2] of a 550ms one-way network delay. Under these assumptions, we provide Table 1 that contains performance and adversarial success values for varying n.

We acknowledge that the times provided in the table can seem long when compared to network protocols, but it is in line with other published work on telephony security (e.g., PinDr0p [15]). Additionally, implementors of LeopardSeal can set specific times and volume levels to maximize the usability of our system. LeopardSeal is intended to be run once nearby a location before arrival and once after arriving at that location to compare results, which we further explain in the next section.

6 Implementation

To test our hypothesis, we observe this attack and compare measurements of mouth-to-ear delay to that of legitimate calls in various scenarios. This section provides an overview of the implementation of the three components necessary to perform our experiments: the *target* end device (i.e., the Prover), the *remote* end device (i.e., the Verifier), and the ARBS. Figure 3 provides an overview of the construction of each device. We then detail our testing methodology.

Christian Peeters[¶], Tyler Tucker[¶], Anushri Jain, Kevin Butler, Patrick Traynor
{cpeeters, tylertucker1, ajain1, butler, traynor}@ufl.edu
University of Florida

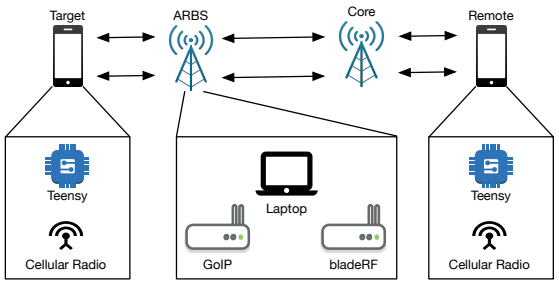

Figure 3: High-level diagram of our ARBS and loopback device implementation.

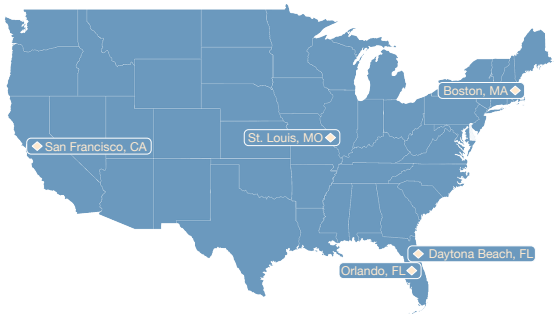

Figure 4: Geographical locations of remote end devices while our ARBS remained at a fixed location in the Southeastern United States.

6.1 ARBS Implementation

The first step to conducting our experiments was to acquire an ARBS. However, these devices are only sold to law enforcement and government agencies under nondisclosure agreements. To the best of our knowledge, no published work in this space has been able to acquire such a device [11].

Because of the difficulty of obtaining these devices, there are many tutorials online that describe how to build a GSM ARBS using commodity hardware. We determined that the most practical and fair way to do this was to assemble our own GSM ARBS using publicly available guides, while also taking into account the typical RTT associated with cellular network hardware outlined in the standards.

We took careful steps to assure that we did not unfairly bias device performance when constructing our ARBS. Specifically, we followed open-source guides for creating ARBSs, which include suggestions for hardware and software [4, 8, 32, 40]. Based on these guides and our experience in this space, we configured these devices to minimize processing and latency overhead as much as possible.

Finally, we validated our construction against the literature on telecommunication latency. Standards and studies on the mouth-to-ear delay suggest that call latency is primarily introduced by wireless uplinks and downlinks [2, 5, 36], which is doubled when connected to an ARBS. These standards anticipate approximately 190ms of delay for both uplink and downlink in one direction (doubled when considering RTT). As such, *approximately 380ms of additional round trip delay is to be expected based on these references.* An ARBS with this performance, regardless of whether it is law enforcement-grade or an open-source project would therefore be representative of what the standards expect and allow.

Our ARBS consists of three main components: a software-defined radio (SDR) to act as our GSM base station, a cellular gateway to connect to a legitimate base station, and a host device to bridge them. We chose the bladeRF x40 [9] as our SDR because it is the only SDR currently supported by YateBTS [14], the most popular open-source GSM software. We only permitted connections from our test sysmocom SIM cards [12] to prevent any unintended device connections.

To bridge calls from the bladeRF into a legitimate provider network, we created a local SIP connection from YateBTS to a SIP server hosted directly on our cellular gateway, a GOIP-1 device [10]. The GOIP-1 is advertised online [8] as an "IMSI-Catcher".[4] The host/bridge runs on a Dell Precision laptop running Ubuntu 18.04 LTS to support YateBTS.

6.2 End Device Construction

We chose to assemble our own cellular end devices to measure RTT for call audio. This allowed us to directly interface with call audio, which is not possible with standard mobile devices. Additionally, building our own hardware provided us with the ability to verify the connection to our ARBS and allowed us to automate the process of collecting RTT data.

Each device consists of a Teensy development board interfaced with 3G cellular modules and Mint Mobile (MVNO of T-Mobile) SIM cards. We validated that our results are not dependent on this network by performing cross-MNO tests, which can be found in Section 7.

Once a remote end device receives a call, it automatically begins sampling the audio in line and retransmitting that audio through the microphone line. This process is consistent for all tests, making the small sampling delay equivalent for every sample we read. We also note that the internal hardware delays do not impact our ability to accurately measure RTT. The only requirement is that the delay is consistent and in the order of several GSM audio frames.

Our target end device also uses a Teensy ARM M7 development board with the same 3G cellular module. In short, the target device initiates the call to a remote end device. Once the call is established, the target transmits an audio tone during a call and measures the time taken from the beginning of that tone to the moment it receives it back. To detect these audio tones, we use an analog-to-digital converter which looks for a deviation in the voltage on the audio input, indicative of audio on the line. We also use techniques such as noise filtering and manual inspection to avoid false-positive audio.

6.3 Test Methodology

6.3.1 Initial Tests and Local Distance Consistency We performed initial experiments to confirm that our target and remote end devices can obtain a consistent measure of RTT. We then performed

[4]As stated in Section 2, the term "IMSI Catcher" is an ambiguous term that here represents an ARBS because it intercepts call audio.

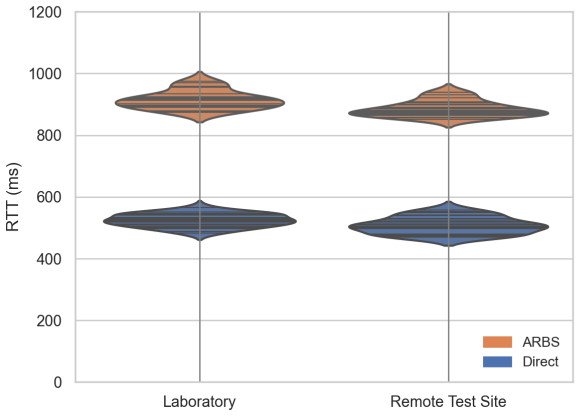

Figure 5: Violin plot comparing RTT measurements of calls within our lab to calls made to a location five miles away. We observe distinct groupings for each path with strong similarities between the two nearby locations.

the same tests, but this time with the target end device connected to our ARBS. We measured the RTT for 20 calls along each call path and compared the two sets to see if the RTT had increased and if it was consistent across multiple calls.

Our next step was to see how small changes in distance impact the RTT for legitimate calls and calls connected to our ARBS. To do this, we moved one of the remote end devices to another location five miles from our lab. We measured 30 phone calls over 2 days for both call path cases. We then compared the legitimate and ARBS-connected calls for each location. We compared both benign and attack call RTTs, and tested for a statistically significant difference.

6.3.2 Long Distance Detection For the next set of experiments, we sent remote end devices across The United States. At each location, we conducted a series of tests over two days. Each RTT sampling test consisted of 100 phone calls, in an effort to capture network behavior at a certain time of day. During each call, our target end device in our lab sends 10 tones to a remote end device and records the RTT of each tone. If the target correctly measures 10 samples, it will take the average and save the result. If any issues arise, the program will discard the incomplete test and immediately redial the remote end device.

We then sent remote devices to multiple locations around the United States via the US Postal Service. Figure 4 shows the locations of these devices. We strategically these locations as they represent a diverse set of points throughout the continental United States.

7 Experimental Results

Across all of our experiments, we frequently perform a standard two-sided t-test which results in two metrics: a t-score and a p-score. These allow us to evaluate the distinction between two RTT data sets. Before performing these calculations, we ensured that our data was normally distributed using the Shapiro-Wilk test.

7.1 Initial Tests & Local Distance Consistency

We begin by evaluating the results of our initial RTT measurements when the end devices and the ARBS are all within our lab. The average time of the 20 legitimate call RTT samples was 507 ms, while the average for calls connected to our ARBS was 885 ms. These results suggest a clear distinction in RTT when the target end device is connected to the ARBS. We then evaluated these measurements with a standard two-sided t-test, which produced $t = 50$ and $p \ll 0.01$. These values strongly support the idea that the two groups of RTTs from each call path are highly self-correlated and negligibly correlated with each other.

After determining that the ARBS had a significant impact on RTT, we moved the location of the remote end device to a test site location five miles away and again collected 40 RTT samples. The results of this experiment are shown in Figure 5 alongside the measurements taken within our lab. In this case, the average times for both paths were within 30 ms of those of the original set of tests. The results of both tests exhibit a statistically significant difference between the legitimate and ARBS-intercepted calls.

7.2 Long Distance Detection

After establishing consistency within a local area, we decided to vary the location of the remote end device throughout the United States. For each location, we collected 200 RTT samples over two days, where half were legitimate calls and half were ARBS-intercepted.

Overall, we observe similar results for RTT and difference of means at each location. All of our calls go through the same ARBS, so the difference of means, or amount of delay added by the ARBS path, is similar for all locations: St. Louis (366 ms), San Francisco (374 ms), and Boston (381 ms). These values align well with our target value of 380 ms mentioned in Section 6. Due to these similarities, we achieve $p \ll 0.01$ for all test locations. These results allow us to reject the null hypothesis associated with the t-test. Finally, we calculate that the average RTT for ARBS paths is 15 standard deviations away from the average RTT for direct paths for these three locations. We offer a visual representation of our statistical analysis as a probability density function in Figure 6. This plot shows us the distribution of the data at each location, with two clear groups of curves which represent the direct and ARBS paths. The peaks of each curve appear in the same order for both paths, suggesting again that the ARBS adds a near-constant delay to the RTT of audio signals. Moreover, the difference between attack and non-attack traffic is exactly as predicted by the standards documents' characterization of additional wireless hops [2, 5, 36].

7.3 LTE Experiments

While the majority of our experiments use 2G and 3G networks, we ran additional experiments in LTE for completeness. We constructed a new loopback device that uses an LTE module, then ran tests against it within our local area. We did not observe significant changes in RTT when comparing these results to existing local results that we previously collected. Our direct results were 515 ms and 466 ms while our ARBS results were 898 ms and 840 ms for 3G and LTE, respectively. These results are about 50 ms apart for each path, which is significantly less than our measured ARBS

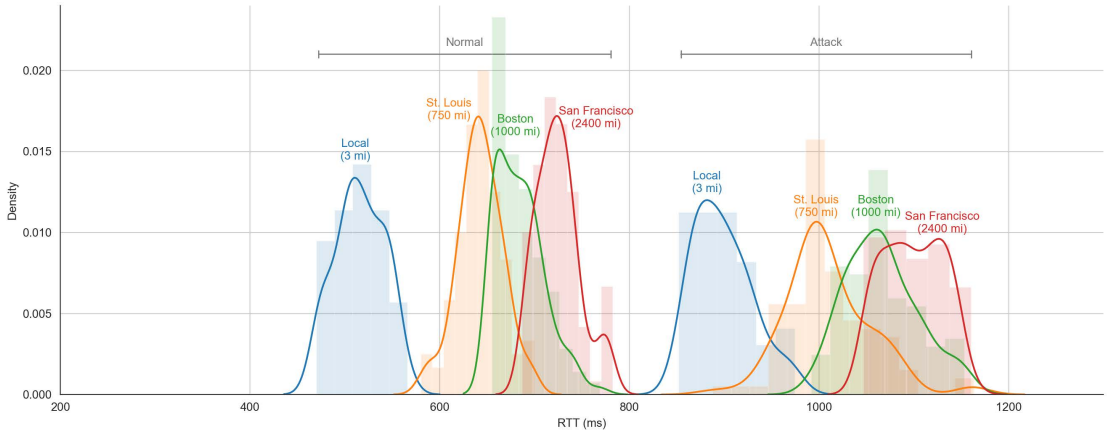

Figure 6: Probability Density Function for tests in our local area, St. Louis, San Francisco, and Boston. The left cluster represents the direct calls for each location while the right cluster represents the ARBS-intercepted calls. We observe a consistent delay of ~380 ms between normal and attack results for each location (e.g., San Francisco Normal: 1093 ms; San Francisco Attack: 705 ms). While we only show these 4 locations for clarity, we have performed tests in a total of 15 cities throughout the United States, all of which produce this consistent delay.

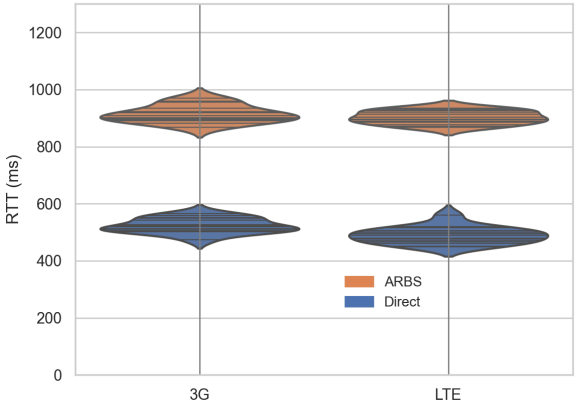

Figure 7: Plots of our local experiments using 3G and LTE in our loopback devices. We observe that results for direct and ARBS paths are very similar.

delay. This means that LTE RTT is generally indistinguishable from 2G/3G RTT. We share our results in Figure 7.

7.4 Time Consistency

Analogous to IP networks, cellular networks are also subject to changes due to diurnal patterns. Specifically, delays in traditional IP networks can vary greatly depending on the time of day and other various network conditions. This phenomenon could have a significant impact on the RTT of call audio and requires additional experimentation. While this effect should not be true in voice calls (variation occurs in call establishment time, but should not vary once the call is connected), we analyzed 200 measurements from our results in Boston. We then looked at the standard deviation of the measured RTT for both legitimate and ARBS-connected calls over the entirety of the time we collected data.

We split the entirety of the data from our Boston RTT measurements into three categories: *Before 10:00 AM*, *10:00 AM - 4:00 PM*, and *After 4:00 PM*. Figure 8(a) shows the distribution of RTT values from each of these categories as a Violin plot. The peaks of each plot stay very consistent for each category representing the different times of day, suggesting that results remain consistent regardless of when calls were made. We chose these times of day to represent the typical morning, core workday, and evening timeframes.

This small difference in RTT allows us to retain the clear distinction between direct and ARBS paths for each time. We can conclude from these results that LeopardSeal will be effective at identifying calls intercepted by an ARBS, regardless of the time of day.

We ran an additional experiment in San Francisco 10 months after our first experiment. We observed no discernable difference between the two sets of measurements, as the average direct and ARBS delays were 706 ms and 1093 ms for 2021, and 700 ms and 1029 ms for 2022. This shows us that our detection technique is robust across long timeframes, in this case around a year apart. Figure 8(b) contains the associated plot for San Francisco.

7.5 Cellular Gateway Evaluation

As mentioned in Section 6, part of the reason for constructing our own ARBS was the legal difficulty of accessing and using law enforcement-grade technology. However, despite our inability to obtain and use such a device, we wanted to verify the ability of LeopardSeal to work against other hardware. To verify that higher-end gateways would not impact our ability to detect ARBSs, we acquired a Dinstar UC2000-VG ($2000), which is considerably more expensive than the GoIP ($100) used in our other experiments.

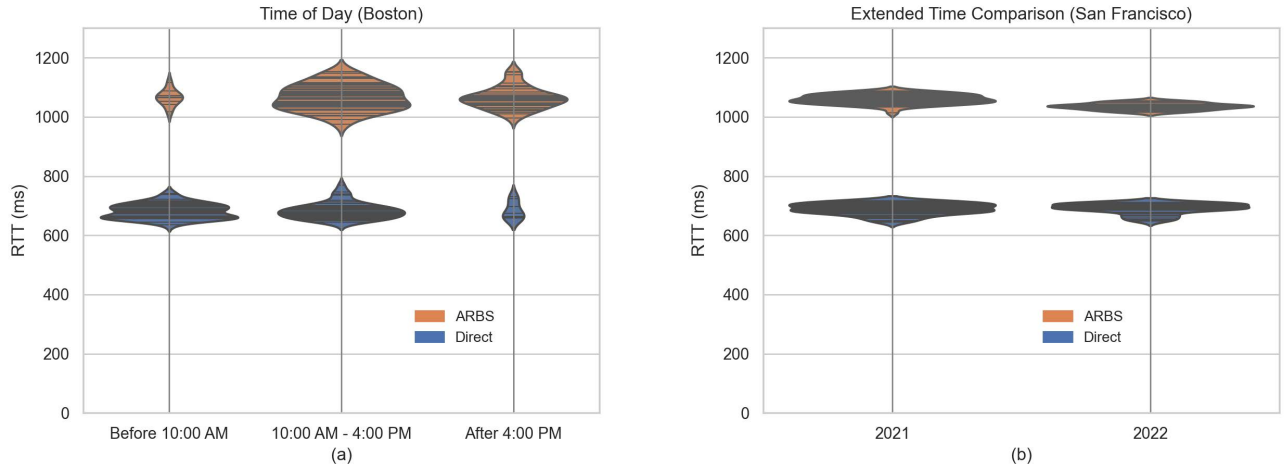

Figure 8: Violin plots showing the effect that different times of day and different years can have on RTT measurements. We observe that neither factor significantly impacts our results, suggesting that LeopardSeal can provide consistent measurements.

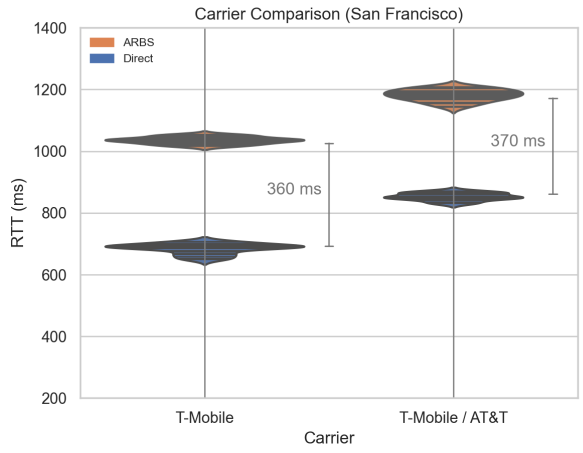

Figure 9: Violin plot showing the difference between experiments using one MNO (T-Mobile) and two different MNOs (T-Mobile and AT&T). We observe cross-carrier delay is significantly less than ARBS delay.

Using the UC2000-VG, we collected RTT measurements for 20 calls between the two end devices located in our lab over 24 hours. We then compared those RTT measurements with the original measurements we took in our lab using the GoIP. We found the average RTT differed by only approximately 25ms, the size of a single audio frame. This difference likely can be contributed to variations in network conditions. As such, this additional experiment further confirms our hypothesis that the wireless hops are the source of delay.

7.6 Cross-MNO Experiments

We acquired a TracFone SIM using the AT&T network to run experiments between different MNOs, as crossing an SS7 node does incur an additional delay to the audio channel. We ran tests on adjacent days with the same loopback device, running Mint Mobile (T-Mobile), then TracFone (AT&T). Figure 9 contains the results of our experiments. We observe an increase in RTT in this scenario, although the added delay is also significantly below that of an ARBS path. Our average cross-MNO delay is 168 ms, which is less than half of the average 380 ms delta that exists in the presence of an ARBS. Most critically, the difference between direct and ARBS paths in both setups remains consistent at approximately 360 ms, which is highlighted in Figure 9. Direct path data for the cross-MNO case remains disjoint from the ARBS data for the single carrier case, so we can still assign a minimum threshold for ARBS detection despite the increase in RTT from communication between separate carriers.

7.7 Detection Threshold

Throughout our experiments, we observe a significant difference in RTT between direct and ARBS paths. Therefore, we can use a threshold as a detection mechanism for an ARBS given a pair of locations for the caller and callee. This value can be arbitrarily set during implementation of our system. Nevertheless, we suggest a threshold of 300 ms greater than the ground truth RTT for a legitimate call based on our results. This value accounts for delays incurred by cross-MNO call paths while ensuring that all results near the expected 380 ms ARBS delay are appropriately flagged.

8 Discussion

LeopardSeal requires additional steps to achieve real-world deployment, as do artifacts in all research papers. The most important step in reaching widespread deployment of LeopardSeal is the support of OEMs. LeopardSeal is intended to be run as part of the default calling application. This is largely due to third-party apps being restricted from having access to call audio for both iOS and Android devices. This also enables user adoption of LeopardSeal, which is

Christian Peeters[¶], Tyler Tucker[¶], Anushri Jain, Kevin Butler, Patrick Traynor
{cpeeters, tylertucker1, ajain1, butler, traynor}@ufl.edu
University of Florida

commonly a hurdle for any new technology. Aside from integration into mobile operating systems, other real-world deployment considerations need to be made as well.

8.1 Impacts on Law Enforcement

The implications of this research have the potential to have an impact on the lawful use of ARBSs by law enforcement and government groups with a warrant. Though the intent of this work is to allow users to detect if they are being unlawfully spied on by unsanctioned ARBSs, the system could be used by criminal suspects to determine if they are under surveillance by law enforcement. This is an unintended consequence of LeopardSeal, however, we believe that this offers a silver lining regarding the lawful use of rogue base stations.

We first note that traditional wiretapping typically occurs in phone switches in compliance with CALEA [58]. The advantage of this approach is that it is undetectable, and can be used to capture call audio and metadata (depending on the warrant) regardless of the location of the surveilled target. That said, incidents such as the January 6th attack on the US Capital may make it necessary to intercept threats as they emerge. What is critical is that standards and laws dictate that such interception should still be undetectable [3, 57].

Without legal access to law enforcement-grade ARBSs, we can not characterize such devices. However, the techniques and hardware they use are not dissimilar to ours, so we have no reason to believe that they would be undetectable. Such devices should therefore be evaluated by appropriate authorities using LeopardSeal.

8.2 Real World Deployment

While deploying LeopardSeal on current mobile phones has challenges, a more challenging issue is in determining how to deploy the remote end device. A caller may not know the location of a potential callee (e.g., another mobile phone user), or may not wish to call them until they believe that their call is not being surveilled.

Many options would work in this space. A user could potentially use another number under their control (e.g., work or home phones) and implement the remote/echo service there. Alternatively, phone numbers mapping to web services/automated voice response (AVR) systems could also be employed. In a relatively straightforward fashion, traditional services such as Time-by-Telephone [49][5] could easily serve in this role.

Our previous experiments demonstrate that attack and benign traffic are easily differentiable based on the delay added by an ARBS. However, this information alone is insufficient to instruct users on how they might actually use LeopardSeal to determine whether or not their calls are currently being surveilled. That is, given the timing output from LeopardSeal, a user needs more than just a single RTT in order to determine the presence of an ARBS.

The answer to this problem is bootstrapping or having a LeopardSeal measurement from a location believed to be safe. One may select such a location by context: for instance, before attending a protest, a potential target may collect a bootstrap value at their home a few miles away from their future location. A natural question then arises, "What range is appropriate for collecting a bootstrap value?"

Our experiments in Section 7 offer a starting point. Specifically, as shown in Figure 5, RTT times between physically close locations in the same town are extremely similar. A user would therefore need only to move one base station away from the ARBS (i.e., out of range). Unfortunately, the problem remains that a user would already need to know that they are under surveillance in order to move away and take their bootstrap measurement. Accordingly, it would be useful to understand how far away from a location where monitoring is suspected a target can bootstrap.

We perform one final experiment to attempt to quantify this answer. We mailed our remote device to Daytona Beach, Florida. This location is approximately 55 miles away from our previous measurements in Orlando, Florida. Twenty samples from each location, half benign and half attack, show that RTT values for both locations are highly similar. A t-test of the benign and attack datasets further confirms the visual intuition, with $p = 0.9$ and $p = 0.8$, respectively, thereby supporting the null hypothesis that these samples are drawn from the same population. As ARBS attacks are highly localized (single microcell range of ≈ 1 mile), an area of approximately $9,500\text{mi}^2$ ($\pi \times 55\text{mi}^2$) provides sufficient space to bootstrap safely. We believe that results begin to degrade much beyond this point. As shown by Peeters et al. [52], who use a different acoustic distance bounding technique to detect SS7 rerouting attacks, distances of approximately 200 miles yield significantly different RTT times. As such, we do not recommend using bootstrapping distances beyond the above experiment.

Finally, we note that once the bootstrapping process is complete, the user will experience only minor fluctuations in RTT values. Cellular networks prioritize call quality, therefore users do not experience significant time fluctuations within a call itself. Rather, these fluctuations occur in call setup and *do not affect* measurements performed by LeopardSeal. Our *Time Consistency* experiments in Section 7 support this observation.

9 Related Work

Cellular network security research has produced several approaches to defending against cellular interceptor devices. While varied, we can categorize all of these approaches as app-based, sensor-based, or network-based according to a recent study by Park et al. [51].

App-based solutions perform detection directly from the end device. Brenninkmeijer [19] and Park et al. [50] conducted studies on popular app-based solutions for Android phones in 2016 and 2017, respectively. Both studies concluded that all applications were easily circumvented and suggested that app developers seek direct access to the phone's baseband processor to strengthen their defense, which is not easily implemented. Li et al. [41] created an application called FBSRadar which used crowdsourced information to detect cellular interceptor devices. While their system is highly effective, it only works against rogue base stations that send out spam SMS messages.

Sensor-based solutions use dedicated hardware to learn about the local network and detect anomalies. Dabrowski et al. [26] identified metrics such as cell ID, base station capabilities, evidence of

[5]"At the tone, the time will be:"

jamming, lack of proper encryption, and more as useful in determining if a base station is a malicious device. A few years later, Ney et al. [48] created a crowdsourced network of cellular interceptor device detectors called Seaglass by installing cellular transceivers on ride-sharing cars to map network topology. Seaglass detects anomalies by looking at broadcast information, location inconsistency, and deployment lifetime of each active base station. Zhuang et al. [67] propose an alternative approach using RF fingerprinting techniques to identify base stations. This approach alone, however, is not sufficient in detecting an ARBS due to the principle of relying on a short deployment lifetime being indicative of ARBSs. Many legitimate circumstances may result in a base station having a short deployment lifetime, such as the deployment of mobile microcells at sporting events. As a result, many of these sensor-based solutions produce false positives.

Network-based solutions rely on existing infrastructure to detect cellular interception devices, thus placing detection on network operators. In 2015, Do et al. [28] proposed a machine learning solution that would detect anomalies through network behavior and returned as Steig et al. [60] with a non-ML solution that relies on measurement reports from end devices. This provides a list of all nearby base stations to the connected base station, in an attempt to identify any unfamiliar entities. Dabrowski et al. [25] conducted a similar study, concluding that invalid LAC codes transmitted by phones, a database of ciphers used by each phone, and transmission delays in control messages are strong indicators of detecting an ARBS. These solutions, however, would introduce significant changes to the network core.

At the core of this work is the application of distance-bounding protocols that determine deviations in call paths by measuring the RTT of exchanged messages. These have been applied to the telephony network core, as well as wireless communication such as UWB and GPS [22, 24, 52, 62].

LeopardSeal is most closely related to Sonar by Peeters et al. [52], which addressed call rerouting via SS7 with the use of distance bounding. However, there are numerous aspects of the two systems that differ, including the impact on RTT being fundamentally different between the two attack scenarios. The delay introduced by ARBSs is less reliant on call path distance than SS7 redirections and focuses on the addition of a wireless hop nearby end devices. Unlike Sonar, LeopardSeal is able to detect small deviations in a call path. Sonar explicitly notes this scenario as a limitation.

In addition to the differences in the fundamentals of the attacks, the capabilities of the two systems also differ. Sonar implements a modified version of the Hancke and Kuhn protocol [31], which is a fundamental distance bounding protocol. This protocol selection addresses only distance fraud and disregards other common distance bounding attacks. In Section 5, we discussed the selection of the distance bounding protocol proposed by Kim and Avoine [38], which provides additional security benefits. This allows LeopardSeal to defend against a more active attacker, unlike Sonar.

10 Conclusion

Audio rogue base stations allow adversaries to eavesdrop on calls. However, the additional wireless hops that incur delays from encryption, transcoding and retransmission create measurable differences in the delay experienced by surveilled calls. In this paper, we develop the LeopardSeal protocol to detect this delay. LeopardSeal uses a secure distance bounding algorithm that we tailor for use in the voice channel. We test our protocol on devices located throughout the continental United States. Our experiments not only validate our hypothesis but also provide guidance on how such a system can be practically deployed. Whether an adversary exploits 2G connectivity, downgrade attacks or the compromise of network credentials, we demonstrate that such attacks can be robustly detected.

Acknowledgments

We would like to express our gratitude to our sponsors for enabling this research. This work was supported by the US Department of Homeland Security Award ID 70RSAT20CB0000017.

A Appendix

The research artifact accompanying this paper is available via https://doi.org/10.5281/zenodo.7933110.

A.1 Extended Protocol Discussion

The fast bit exchange phase of the protocol requires the prover and the verifier to exchange specific m-ary values based on the previous sequences. In the LeopardSeal protocol, P and V exchange these values over the audio channel of a call. The most intuitive way to exchange data over this channel is to use the long-existing system of DTMF tones, which in turn, requires us to convert the base four challenges and base two responses into base 16. Doing this allows us to combine two challenges into a single base 16 value which can be mapped to a DTMF tone. This is another reason why we needed to base LeopardSeal on a round-independent distance bounding protocol, the ability to send multiple challenges and responses at once. Moreover, it is possible for us to have simply used a m value of 16 instead of four, eliminating the need to convert to base 16. We chose not to do this at this time, for performance reasons and the lack of existing security evaluations of the KA2 protocol for a m value of 16.

The fast bit exchange phase executes for n iterations. Each iteration i begins with V selecting a random m-ary value $S_i \in 0 : m-1$. V then creates a challenge $\boldsymbol{C_i}$ whose value is determined by the corresponding $\boldsymbol{T_i}$. If $T_i = 1$, C_i is set to the randomly generated S_i. If $T_i = 0$, C_i is instead set to the pre-defined m-ary challenge D_i. Because we choose four as the value for m, two challenges C_i and C_{i+1} are calculated at once and then combined into a base 16 value which is converted into a DTMF tone and sent to P. Upon sending the DTMF tone to P, V will start a timer.

Upon receiving the DTMF tone, P converts the tone back into the two base four challenges, C_i and C_{i+1}. Starting with the first challenge C_i, P observes the corresponding T_i to determine a response $\boldsymbol{R_i}$. If $T_i = 1$, P sets R_i to the binary value within the sequence v_i^j where j is the random value S_i sent in the challenge C_i. If $T_i = 0$, P observes if C_i matches D_i. If the values match, P assumes the

Christian Peeters[¶], Tyler Tucker[¶], Anushri Jain, Kevin Butler, Patrick Traynor
{cpeeters, tylertucker1, ajain1, butler, traynor}@ufl.edu
University of Florida

received challenge C_i was not randomly selected and sets R_i to v_i^0. If C_i and D_i do not match, P assumes the presence of a malicious party and may continue the protocol using all random responses for the remainder of the protocol or terminate the call. If the presence of a malicious party is not assumed, P will then repeat the process for C_{i+1} to obtain R_{i+1}. P then combines R_i and R_{i+1} into a single DTMF tone [6], which is then sent to V.

P and V are working together in the LeopardSeal protocol, so P may also independently measure time by starting a timer upon sending the response to V. This timer would then be stopped upon receiving the next challenge from V and recorded. By allowing P to measure the RTT, it further decreases the likelihood of successful impersonation fraud. This is because the ARBS will have to accurately produce the correct RTT for both P and V, as opposed to just V.

Once V receives the DTMF tone, it stops the timer and records the received responses R_i and R_{i+1} and the time between sending the challenge and receiving the response from P. This process is repeated until all n challenges are sent from V to P and all n responses are sent from P to V. Once this is done, the fast bit exchange phase is complete.

The final reconciliation phase of the LeopardSeal protocol allows P and V to determine if an adversary was present during the fast bit exchange. To come to this conclusion, P and V observe the series of challenges and responses exchanged. If all n challenges and responses matched what was expected from sequences calculated during the first phase, P and V conclude that there were no adversarial DTMF tones played during the fast bit exchange. P and V then take the average of all the RTT measurements previously recorded. If the average RTT is within a predefined acceptable range, approximately what is expected for a legitimate call, the end devices inform their users that the call is not currently under attack by an ARBS. Conversely, if the RTT was not within the predefined RTT range, the call is terminated and the users are informed.

[6]Because only the challenges are m-ary, the 2 combined responses will only ever result in eight possible DTMF tones.

References

[1] 2000. *Specification of Dual Tone Multi-Frequency (DTMF) Transmitters and Receivers; Part 2: Transmitters.* Technical Report ES 201 235-2. European Telecommunications Standards Institute (ETSI).

[2] 2003. *One-Way Transmission Time.* Technical Report G.114. International Telecommunications Union (ITU).

[3] 2006. *Lawful Interception (LI); Concepts of Interception in a Generic Network Architecture.* Technical Report ETSI TR 101 943 V2.2.1. European Telecommunications Standards Institute (ETSI).

[4] 2016. Building Your Own Rouge GSM Basestation with a BladeRF. (2016). https://www.rtl-sdr.com/building-your-own-rogue-gsm-basestation-with-a-bladerf/

[5] 2016. *End-to-End Quality of Service for Voice over 4G Mobile Networks.* Technical Report ITU-T G.1028. Telecommunication Standardization Sector of ITU.

[6] 2016. How to Build Your Own Rogue GSM BTS for Fun and Profit. (2016). https://www.evilsocket.net/2016/03/31/How-To-Build-Your-Own-Rogue-GSM-BTS-For-Fun-And-Profit/

[7] 2019. A Comparative Introduction to 4G and 5G Authentication. *Informed Insights by CableLabs* (2019). https://www.cablelabs.com/insights/a-comparative-introduction-to-4g-and-5g-authentication

[8] 2021. *Best Price GoIP 1 Port IMSI Catcher, SIM Server GSM Gateway with IMEI Change Low Cost IMSI Catcher.* https://www.alibaba.com/product-detail/best-price-goip-1-port-imsi_60559890007.html

[9] 2021. *bladeRF x40.* https://www.nuand.com/product/bladerf-x40/

[10] 2021. *Shenzhen Hybertone Technology Co., Ltd.* http://www.hybertone.com/en/pro_detail.asp?proid=10

[11] 2021. *Stingray Phone Tracker.* https://en.wikipedia.org/w/index.php?title=Stingray_phone_tracker&oldid=1011185824

[12] 2021. *Sysmocom Site - sysmoUSIM-SJS1 SIM + USIM Card (10-Pack).* http://shop.sysmocom.de/products/sysmousim-sjs1

[13] 2021. Xcell. 4G (LTE) Direct Interception System (Without Downgrading to 3G or 2G). https://www.discoverytelecom.eu/catalog/5438.html

[14] 2021. *YateBTS - LTE & GSM Mobile Network Components for MNO & MVNO.* https://yatebts.com/

[15] Chaitrali Amrutkar, Young Seuk Kim, and Patrick Traynor. 2016. Detecting Mobile Malicious Webpages in Real Time. *IEEE Transactions on Mobile Computing* 16, 8 (2016), 2184–2197.

[16] Gildas Avoine, Muhammed Ali Bingöl, Ioana Boureanu, Srdjan Čapkun, Gerhard Hancke, Süleyman Kardaş, Chong Hee Kim, Cédric Lauradoux, Benjamin Martin, Jorge Munilla, et al. 2018. Security of Distance-Bounding: A Survey. *ACM Computing Surveys (CSUR)* 51, 5 (2018), 1–33.

[17] Sam Biddle. 2016. Long-Secret Stingray Manuals Detail How Police Can Spy on Phones. *The Intercept* (2016). https://theintercept.com/2016/09/12/long-secret-stingray-manuals-detail-how-police-can-spy-on-phones/

[18] Ravishankar Borgaonkar, Lucca Hirschi, Shinjo Park, and Altaf Shaik. 2019. New Privacy Threat on 3G, 4G, and Upcoming 5G AKA Protocols. *Proceedings on Privacy Enhancing Technologies* 2019, 3 (2019), 108–127.

[19] Bauke Brenninkmeijer. 2016. Catching IMSI-Catcher-Catchers: An Effectiveness Review of IMSI-Catcher-Catcher Applications. *Bachelor Thesis, Radboud University (Nijmegen, The Netherlands)* (2016).

[20] Richard Chirgwin. 2018. Now You, Too, Can Snoop on Mobe Users from 3G to 5G with a Raspberry Pi and €1,100 of Gizmos. (2018). https://www.theregister.com/2018/12/05/mobile_users_can_be_tracked_with_cheap_kit_aka_protocol/

[21] Tom Chothia, Joeri De Ruiter, and Ben Smyth. 2018. Modelling and Analysis of a Hierarchy of Distance Bounding Attacks. In *Proceedings of the USENIX Security Symposium.*

[22] Jolyon Clulow, Gerhard P Hancke, Markus G Kuhn, and Tyler Moore. 2006. So Near and Yet So Far: Distance-Bounding Attacks in Wireless Networks. In *Proceedings of the European Workshop on Security in Ad-hoc and Sensor Networks (ESAS).*

[23] Cas Cremers and Martin Dehnel-Wild. 2019. Component-Based Formal Analysis of 5G-AKA: Channel Assumptions and Session Confusion. In *Proceedings of the ISOC Network and Distributed System Security (NDSS) Symposium.*

[24] Cas Cremers, Kasper B. Rasmussen, Benedikt Schmidt, and Srdjan Capkun. 2012. Distance Hijacking Attacks on Distance Bounding Protocols. In *Proceedings of the IEEE Symposium on Security and Privacy.*

[25] Adrian Dabrowski, Georg Petzl, and Edgar R. Weippl. 2016. The Messenger Shoots Back: Network Operator Based IMSI Catcher Detection. In *Proceedings of the International Symposium on Research in Attacks, Intrusions, and Defenses (RAID).*

[26] Adrian Dabrowski, Nicola Pianta, Thomas Klepp, Martin Mulazzani, and Edgar R. Weippl. 2014. IMSI-Catch Me If You Can: IMSI-Catcher-Catchers. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC).*

[27] Martin Dehnel-Wild and Cas Cremers. 2018. Security Vulnerability in 5G-AKA Draft. *Department of Computer Science, University of Oxford, Tech. Rep* (2018), 14–37.

[28] Thanh Van Do, Hai Thanh Nguyen, Nikolov Momchil, and Van Thuan Do. 2015. Detecting IMSI-Catcher Using Soft Computing. In *Proceedings of the International Conference on Soft Computing in Data Science (SCDS).*

[29] Saar Drimer and Steven J. Murdoch. 2007. Keep Your Enemies Close: Distance Bounding Against Smartcard Relay Attacks. In *Proceedings of the USENIX Security Symposium.*

[30] Justin Fenton. 2016. Key Evidence in City Murder Case Tossed Due to Stingray Use. *The Baltimore Sun* (2016). https://www.baltimoresun.com/news/crime/bs-md-ci-stingray-murder-evidence-suppressed-20160425-story.html

[31] Gerhard P Hancke and Markus G Kuhn. 2005. An RFID Distance Bounding Protocol. In *Proceedings of the International Conference on Security and Privacy for Emerging Areas in Communications Networks (SecureComm).*

[32] Grant Hernandez and Tyler Tucker. 2020. Creating a Cellular Testbed with YateBTS and srsLTE. (2020). https://hernan.de/blog/creating-a-cellular-testbed-with-yatebts-and-srslte/

[33] Aleksander Hougen. 2022. How to Block StingRay Surveillance in 2022 in 2G, 3G, 4G & 5G Networks. *Cloudwards* (2022). https://www.cloudwards.net/how-to-block-stingray-surveillance/

[34] Tom Jackman. 2017. Police Use of 'StingRay' Cellphone Tracker Requires Search Warrant, Appeals Court Rules. (2017). https://www.washingtonpost.com/news/true-crime/wp/2017/09/21/police-use-of-stingray-cellphone-tracker-requires-search-warrant-appeals-court-rules/

[35] M Awais Javed and Sohaib khan Niazi. 2019. 5G Security Artifacts (DoS/DDoS and Authentication). In *Proceedings of the International Conference on Communication Technologies (ComTech).*

[36] Wenyu Jiang, Kazuumi Koguchi, and Henning Schulzrinne. 2003. QoS Evaluation of VoIP End-Points. In *Proceedings of the IEEE International Conference on*

Communications.
[37] Erin Kelly. 2017. Bipartisan Bill Seeks Warrants for Police Use of 'Stingray' Cell Trackers. (2017). https://www.usatoday.com/story/news/politics/onpolitics/2017/02/15/bipartisan-bill-seeks-warrants-police-use-stingray-cell-trackers/97954214/
[38] Young-Sik Kim and Sang-Hyo Kim. 2011. RFID Distance Bounding Protocol Using m-ary Challenges. In *Proceedings of the International Conference on Information and Communication Technology Convergence (ICTC)*.
[39] Keith Kirkpatrick. 2020. Who Has Access to Your Smartphone Data? *Commun. ACM* 63, 10 (2020), 15–17.
[40] Alissa Valentina Knight. 2020. Hacking GSM: Building a Rogue Base Station to Hack Cellular Devices. https://www.linkedin.com/pulse/hacking-gsm-building-rogue-base-station-hack-cellular-alissa/.
[41] Zhenhua Li, Weiwei Wang, Christo Wilson, Jian Jhen Chen, Chen Qian, Taeho Jung, L. C. Zhang, Kebin Liu, Xiangyang Li, and Yunhao Liu. 2017. FBS-Radar: Uncovering Fake Base Stations at Scale in the Wild. In *Proceedings of the ISOC Network and Distributed System Security (NDSS) Symposium*.
[42] Sjouke Mauw, Zach Smith, Jorge Toro-Pozo, and Rolando Trujillo-Rasua. 2018. Distance-Bounding Protocols: Verification without Time and Location. In *Proceedings of the IEEE Symposium on Security and Privacy*.
[43] Silvere Mavoungou, Georges Kaddoum, Mostafa Taha, and Georges Matar. 2016. Survey on Threats and Attacks on Mobile Networks. *IEEE Access* 4 (2016), 4543–4572.
[44] Tim McMillan. 2021. A Secretive Technology Could Be Bad News For Capitol Rioters. https://thedebrief.org/a-secretive-technology-could-be-bad-news-for-capitol-rioters/.
[45] Ulrike Meyer and Susanne Wetzel. 2004. A Man-in-the-Middle Attack on UMTS. In *Proceedings of the ACM Workshop on Wireless Security (WiSec)*.
[46] Yomna Nasser. 2019. Gotta Catch 'Em All: Understanding How IMSI-Catchers Exploit Cell Networks. *Electronic Frontier Foundation* (2019). https://www.eff.org/wp/gotta-catch-em-all-understanding-how-imsi-catchers-exploit-cell-networks
[47] Lily Hay Newman. 2018. DC's Stingray Mess Won't Get Cleaned Up. *Wired* (2018). https://www.wired.com/story/dcs-stingray-dhs-surveillance/
[48] Peter Ney, Ian Smith, Gabriel Cadamuro, and Tadayoshi Kohno. 2017. SeaGlass: Enabling City-Wide IMSI-Catcher Detection. (2017), 39–56.
[49] United States Naval Observatory. 2021. *Telephone Time*. https://www.usno.navy.mil/USNO/time/telephone-time
[50] Shinjo Park, Altaf Shaik, Ravishankar Borgaonkar, Andrew Martin, and Jean-Pierre Seifert. 2017. White-Stingray: Evaluating IMSI Catchers Detection Applications. In *Proceedings of the USENIX Workshop on Offensive Technologies (WOOT)*.
[51] Shinjo Park, Altaf Shaik, Ravishankar Borgaonkar, and Jean-Pierre Seifert. 2019. Anatomy of Commercial IMSI Catchers and Detectors. In *Proceedings of the ACM Workshop on Privacy in the Electronic Society (WPES)*.
[52] Christian Peeters, Hadi Abdullah, Nolen Scaife, Jasmine Bowers, Patrick Traynor, Bradley Reaves, and Kevin Butler. 2018. Sonar: Detecting SS7 Redirection Attacks with Audio-Based Distance Bounding. In *Proceedings of the IEEE Symposium on Security and Privacy*.
[53] Roger Piqueras Jover and Vuk Marojevic. 2019. Security and Protocol Exploit Analysis of the 5G Specifications. *IEEE Access* 7 (2019), 24956–24963.
[54] Vassilis Prevelakis and Diomidis Spinellis. 2007. The Athens Affair. *IEEE Spectrum* (2007). https://spectrum.ieee.org/telecom/security/the-athens-affair
[55] Cooper Quintin. 2019. The 5G Protocol May Still Be Vulnerable to IMSI Catchers. *Electronic Frontier Foundation* (2019). https://www.eff.org/deeplinks/2019/01/5g-protocol-may-still-be-vulnerable-imsi-catchers
[56] Bradley Reaves, Logan Blue, and Patrick Traynor. 2016. Authloop: End-to-End Cryptographic Authentication for Telephony Over Voice Channels. In *Proceedings of the USENIX Security Symposium*.
[57] Stefan Rommer, Peter Hedman, Magnus Olsson, Lars Frid, Shabnam Sultana, and Catherine Mulligan. 2019. *5G Core Networks: Powering Digitalization*.
[58] Micah Sherr, Eric Cronin, Sandy Clark, and Matt Blaze. 2005. Signaling Vulnerabilities in Wiretapping Systems. In *Proceedings of the IEEE Symposium on Security and Privacy*.
[59] Eric Southern, Abdelkader Ouda, and Abdallah Shami. 2013. Securing USIM-Based Mobile Communications from Interoperation of SIM-Based Communications. *The International Journal for Information Security Research (IJISR)* 3 (2013), 313–324.
[60] Simen Steig, Andre Aarnes, Thanh Van Do, and Hai Thanh Nguyen. 2016. A Network Based IMSI Catcher Detection. In *Proceedings of the International Conference on IT Convergence and Security (ICITCS)*.
[61] Daehyun Strobel. 2007. IMSI catcher. *Chair for Communication Security, Ruhr-Universität Bochum* (2007).
[62] Nils Ole Tippenhauer, C. Pöpper, Kasper Bonne Rasmussen, and S. Capkun. 2011. On the Requirements for Successful GPS Spoofing Attacks. In *Proceedings of the ACM Conference on Computer and Communications Security*.
[63] Yu-Ju Tu and Selwyn Piramuthu. 2020. On Addressing RFID/NFC-Based Relay Attacks: An Overview. *Decision Support Systems* 129 (2020).
[64] Zack Whittaker. 2017. Security Flaw Shows 3G, 4G LTE Networks are Just as Prone to Stingray Phone Tracking. (2017). https://www.zdnet.com/article/stingray-security-flaw-cell-networks-phone-tracking-surveillance/
[65] Zack Whittaker. 2019. New 5G Flaws Can Track Phone Locations and Spoof Emergency Alerts. *Tech Crunch* (2019). https://techcrunch.com/2019/11/12/5g-flaws-locations-spoof-alerts/
[66] Kim Zetter. 2020. How Cops Can Secretly Track Your Phone. *The Intercept* (2020). https://theintercept.com/2020/07/31/protests-surveillance-stingrays-dirtboxes-phone-tracking/
[67] Zhou Zhuang, Xiaoyu Ji, Taimin Zhang, Juchuan Zhang, Wenyuan Xu, Zhenhua Li, and Yunhao Liu. 2018. FBSleuth: Fake Base Station Forensics via Radio Frequency Fingerprinting. In *Proceedings of the Asia Conference on Computer and Communications Security (ASIACCS)*.