GSMem: Data Exfiltration from Air-Gapped Computers over GSM Frequencies

Mordechai Guri, Assaf Kachlon, Ofer Hasson, Gabi Kedma, Yisroel Mirsky, and Yuval Elovici, *Ben-Gurion University of the Negev*

https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/guri

This paper is included in the Proceedings of the
24th USENIX Security Symposium

August 12–14, 2015 • Washington, D.C.

ISBN 978-1-939133-11-3

GSMem: Data Exfiltration from Air-Gapped Computers over GSM Frequencies

Mordechai Guri, Assaf Kachlon, Ofer Hasson, Gabi Kedma, Yisroel Mirsky[1], Yuval Elovici[1]
{gurim, assafka, hassonof, gabik, yisroel, elovici}@post.bgu.ac.il
Ben-Gurion University of the Negev, Beer-Sheva, Israel
[1] Telekom Innovation Laboratories at Ben-Gurion University, Beer-Sheva, Israel

Abstract

Air-gapped networks are isolated, separated both logically and physically from public networks. Although the feasibility of invading such systems has been demonstrated in recent years, exfiltration of data from air-gapped networks is still a challenging task. In this paper we present *GSMem*, a malware that can exfiltrate data through an air-gap over cellular frequencies. Rogue software on an infected target computer modulates and transmits electromagnetic signals at cellular frequencies by invoking specific memory-related instructions and utilizing the multi-channel memory architecture to amplify the transmission. Furthermore, we show that the transmitted signals can be received and demodulated by a rootkit placed in the baseband firmware of a nearby cellular phone. We present crucial design issues such as signal generation and reception, data modulation, and transmission detection. We implement a prototype of GSMem consisting of a transmitter and a receiver and evaluate its performance and limitations. Our current results demonstrate its efficacy and feasibility, achieving an effective transmission distance of 1 - 5.5 meters with a standard mobile phone. When using a dedicated, yet affordable hardware receiver, the effective distance reached over 30 meters.

1. Introduction

Security-aware organizations take various steps to prevent possible theft or leakage of sensitive information. The computers responsible for storing and processing sensitive information often operate on air-gapped networks. These networks are physically disconnected from non-essential networks, primarily those in the public domain. With the growing awareness of negligent or malicious insiders compromising air-gapped networks, as evidenced in several incidents [1] [2], some organizations have begun to restrict USB access, to prevent malware infection or data leakage via USB thumb-drives [3].

Acknowledging the security risks of mobile phones equipped with cameras, Wi-Fi, or Bluetooth, some organizations has restricted their use, forbidding them in classified areas. For instance, an Intel Corporation best-practices document [4] asserts: "*Currently, manufacturing employees can use only basic corporate-owned cell phones with voice and text messaging features. These phones have no camera, video, or Wi-Fi.*" In another case, visitors at one of Lockheed-Martin's facilities [5] are instructed as follows: "*Because ATL is a secure facility, the following items are not allowed to our floor of the building: cameras (film, video, digital), imaging equipment, tape recorders, sound recording devices. Cell phones are allowed, but camera/recording features may not be used.*" Similar regulations are likely to be found in many other security-aware organizations. Clearly, the issue of information leakage associated with basic cellular phones or a phone without a camera, Wi-Fi and the like, has been overlooked in cases in which such phones are allowed in the vicinity of air-gapped computers. However, modern computers are electronic devices and are bound to emit some electromagnetic radiation (EMR) at various wavelengths and strengths. Furthermore, cellular phones are agile receivers of EMR signals. Combined, these two factors create an invitation for attackers seeking to exfiltrate data over a covert channel.

In this paper, we present an adversarial attack model in which any basic desktop computer can covertly transmit data to a nearby mobile phone. Transmission is accomplished by invoking specific memory-related CPU instructions that produce baseband compliant EMR at GSM, UMTS, and LTE frequencies. By using the functionality of multi-channel memory architecture, the signals are amplified and transmitted with increased power. These signals are received and decoded by a rootkit installed at the baseband of a standard mobile phone. To demonstrate the feasibility of the attack model, we developed *GSMem*, a bifurcated malware that consists of a transmitter that operates on a desktop computer and a receiver that runs on a GSM mobile phone. We implemented communication protocols for data modulation and channel reliability and provide extensive experimental results.

As will be explained later, the proposed method is applicable with GSM, UMTS, and LTE basebands. In this paper we focus on a prototype using a GSM mobile phone as receiver, hence the codename, *GSMem*.

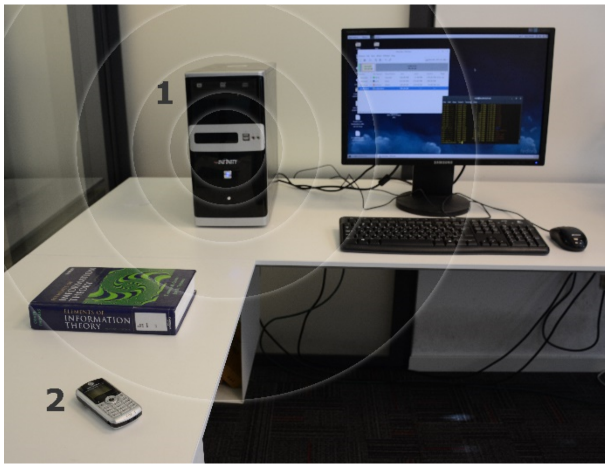

Figure 1: Demonstration of the covert channel in a working environment. Signals at GSM frequencies are emitted from the workstation and received by the nearby compromised mobile phone.

Figure 1 demonstrates the covert channel in a typical real-life scenario, in which rogue software on a computer (1) modulates sensitive information and transmits it over GSM cellular frequencies. The transmissions take place while the computer is at work, without affecting the user experience. A baseband level rootkit on the cellular phone (2) receives the signals and demodulates them, converting them into meaningful information. Note that the components exploited by the proposed model are present on virtually all computers and cellular devices, even on low-end cellular devices which are often allowed into classified environments.

1.1. The Closed Nature of the Baseband Industry

The baseband chip of a cellular device manages the low-level Radio Frequency (RF) connection with the cellular network, thereby making it an indispensable component. The baseband processor runs a real time operating system (RTOS), stored in its firmware. The code is closed to the public, and only the device manufacturer can access the baseband chip's functionality through a limited interface [6]. The RTOS source code, along with the protocol stack and other implementation details, are well-guarded trade secrets, kept off-limits by the protective baseband industry, which is led by a handful of high-ranking players that dominate the market [7]. Lacking access to this information, including documentation and implementation details, independent software vendors cannot intelligently develop new products and interfacing technologies for baseband chips.

It can be argued that the current state of affairs promotes "security through obscurity" by masking the internal workings of the baseband systems. However, this policy has only limited effectiveness. Skilled hackers working on behalf of advanced persistent attackers eventually manage to exploit baseband systems—obscure and isolated though they may be. Baseband exploitation and attacks are thoroughly discussed by Weinmann [8] [9] [10]. Welte and Markgraf [6] also point out several security problems associated with current commercial baseband technology and practices.

1.2. Paper Contributions

While emission security (EMSEC) in itself is not a new concept [11], this paper offers the following original contributions: (1) a novel method for transmitting signals at cellular frequency bands from an ordinary desktop computer, using multi-channel memory related CPU instructions without any special or additional hardware, and (2) a novel method for receiving and demodulating EMR signals using a rootkit in the baseband firmware of a mobile phone, thus turning virtually any mobile phone into an effective EMR eavesdropping device without the use of specialized equipment. We believe the proposed adversarial attack model constitutes a new security threat that security experts should be aware of.

While the bulk of this paper focuses on the mobile phone as a receiver, we also evaluate an alternative communication method in which the transmitter uses memory-related CPU instructions to emit EMR, and the receiver uses software defined radio (SDR) with dedicated, yet affordable hardware. This allows us to study the capabilities and boundaries of the transmission method on a wider scale.

The remainder of this paper is organized as follows: In Section 2 we present assorted related works, along with a concise review of our contributions. Next, in Section 3, we present the adversarial attack model. In Section 4 we present the essential technical background. Section 5 provides a detailed description of the transmitter, followed by Section 6 which describes the receiver. In Section 7 we evaluate GSMem and present the results. Next, in Section 8, we discuss possible defensive countermeasures. Finally, we conclude in Section 9.

2. Related Work

EMSEC, reviewed by Anderson [11], addresses attacks which use compromised emanations of either conducted or radiated electromagnetic signals. Concern about this issue dates back to World War I, but for decades it was relegated solely to governmental and military agencies [12]. However in 1985, van Eck [13] showed how the so-called TEMPEST exploits can be conducted using affordable equipment. He managed to reconstruct an image from electromagnetic signals produced by a video card at a considerable distance, using a modified TV set. Around 2000, Kuhn and Anderson released several publications related to TEMPEST [14] [15], demonstrating that EMR emissions originating from a desktop computer can be manipulated by appropriate software, in either a defensive or offensive manner. Public interest in EMSEC and TEMPEST was amplified by web publications, offering a glimpse into classified TEMPEST-related official standards [16], or providing 'do it yourself' tutorials related to TEMPEST exploits. Thiele [17] provides an open source program dubbed "TEMPEST for Eliza", utilizing the computer CRT monitor to modulate and transmit radio signals at AM frequencies.

Note that side-channels have a variety of possible uses, beyond intentional exfiltration of information as described in this paper. Side-channels may be used for eavesdropping, attacking sophisticated encryption methods, defensive detection of hidden malicious activities, and other uses. Furthermore, side-channels are not limited to electromagnetic radiation (EMR). Clark, Ransford et al [18] refer to power consumption as a side-channel that can reveal hidden information or activities. They present 'WattsUpDoc', a system that detects the presence of malware on medical embedded devices by measuring their power consumption. Rührmair et al [19] discuss the use of power and timing side-channels to attack physical unclonable functions (PUFs). Other researchers investigating side-channels go beyond EMR emanations. Halevy and Saxena [20], explore acoustical eavesdropping, focusing on keyboard acoustical emanations. Hanspach and Goetz [21] present so-called "covert acoustical networks". Their method is based on near-ultrasonic waves, transmitted by the speaker of one laptop computer and received by the microphone of a nearby laptop computer. Callan et al [22] provide a method for measuring the so-called "signal available to the attacker" (SAVAT), with a side-channel based on instruction-level events. Their method is based on the EMR emitted by rather generic CPU/memory operations. The receiver, however, comprises expensive dedicated equipment, and the range of explored distances is quite limited. Guri et al [23] present AirHopper, a bifurcated malware in which the transmitter exploits the EMR emanated by the VGA cable. The receiver is an FM-enabled standard cellular phone.

2.1 Comparison of Relevant Covert Channels

Current state-of-the-art covert channels methods that could be used to exfiltrate data from air-gapped networks involve various physical effects, such as FM transmissions from a display cable [23], ultrasonic acoustic emissions from a speaker [21] [24], EMR emitted by generic CPU operations [22], and thermal emission [25]. Our method, GSMem, uses emissions produced by multi-channel memory data bus. Table 1 provides a brief comparison between GSMem and other current models.

Method	Transmitter	Receiver	Distance (m)	Rate (bit/s)
AirHopper **[23]** (78MHz -108MHz)	Display cable	Cellular FM receiver	7	104-480
Ultrasonic **[21] [24]**	Speaker	Microphone	19.7	20
SAVAT **[22]** (~80KHz)	CPU/memory (laptops)	Dedicated equipment	1.0	N/A
BitWhisper **[25]**	Computer CPU/GPU	Computer Heat Sensors	0.4	8 bit/hour
GSMem (cellular frequencies)	RAM bus (multi-channel)	Baseband	5.5	1-2
		Dedicated equipment	30+	100-1000

Table 1: Comparison of current covert channels for air-gapped networks

As can be seen, all five methods utilize basic computer equipment as the transmitter. However, whereas a display cable or a speaker may not be present on every conceivable computer configuration [26], the CPU and memory, utilized by GSMem and SAVAT, are always present. On the receiver's end, a microphone may not be present on every computer, particularly within a classified zone [26]. A cellular FM receiver (as used by AirHopper) may not be present on every mobile phone, while the baseband processor (used by GSMem) is an integral part of any mobile phone.

In terms of bandwidth, with the dedicated hardware receiver we achieved bit rates of 100 to 1000 bit/s. However, when using a mobile phone as the receiver, the bit rate was much slower (2 bit/s) – making this equipment suitable for leaking small amount of data. It is important to note that our concept was demonstrated on a nine year old low-end phone, the only available alternative with open source firmware, given the protective nature of the baseband industry. Demonstrating the same concept on newer basebands will likely yield better results, and is left as a future research direction.

3. The Adversarial Attack Model

GSMem, viewed as a concept, contributes to the general domain of covert channels. However, we describe a particular attack model which might utilize this covert channel for the purpose of data exfiltration. The adversarial attack model is bifurcated since it requires both a contaminated computer to serve as a *transmitter* and a contaminated mobile phone to serve as a *receiver*. Infecting a computer within an air-gapped network can be accomplished, as demonstrated by the attacks involving Stuxnet [27] [28], Agent.Btz [2] and others [1] [29] [30] [31]. Compromising a mobile phone can occur via social engineering, malicious apps, USB interface, or physical access [32] [33] [34]. Once a compromised mobile phone is in the vicinity of an infected computer, it can detect, receive and decode any transmitted signals and store the relevant acquired information. Later, the phone can transmit the data to the attacker via mobile-data, SMS, or Wi-Fi (in the case of smartphones). Although this attack model is somewhat complicated, attackers have grown more sophisticated, and complex attack patterns have increasingly been proven feasible during the last few years [35] [36] [37] [38].

4. Technical Background

The exfiltration channel is based on the emission of electromagnetic signals, in the frequencies allocated to cellular bands. These signals can be picked up by a malicious component located at the baseband level of a nearby mobile phone. In this section, we provide an overview and some helpful technical background information about cellular networks and frequency bands, along with the basics of baseband components in mobile phones.

4.1. Cellular Networks

2G, and the newer 3G and 4G networks are three 'generations' of mobile networks. Each generation has its own set of standards, network architecture, infrastructure, and protocol. 2G, 3G, and 4G networks are commonly referred to as GSM, UMTS, and LTE respectively, generally reflecting, the implementation of these standards. In this paper, we use the terms GSM, UMTS, and LTE to denote the three generations.

4.1.1. Cellular Network Bands

Wireless communication between mobile-handsets (i.e., mobile phones) and the cellular network takes place through a base transceiver station (BTS), which handles the radio link protocols with the handsets. Communication with the BTS takes place over 'frequency bands' allocated for the cellular network. Various standards define the radio frequencies allocated to each band. In practice, the standard in use depends on the country, region, and support of the cellular provider. Modern mobile phones support all common frequency bands for GSM, UMTS, and LTE, although some phones are region specific. Table 2 shows the main frequency bands supported by modern mobile phones. Each band encompasses frequencies within a range surrounding (above and below) the main frequency. For example, GSM-850 has a frequency range between 824.2MHz and 894.2MHz. Lists of bands and their allocated frequencies are specified by the standards [39].

Standard	Frequency band (MHz)
GSM	850 / 900 / 1800 / 1900
UMTS	850 / 900 / 1900 / 2100
LTE	800 / 850 / 900 / 1800 / 1900 / 2100 / 2600

Table 2: The main frequency bands for GSM, UMTS and LTE cellular networks.

4.1.2. ARFCN

The communication (transmission and reception) between the mobile phone and the BTS occurs over a subset of frequencies within the entire frequency band. The absolute radio-frequency channel number (ARFCN) specifies a pair of radio carriers used for transmission (uplink) and reception (downlink) in GSM networks. For example, the GSM-850 band consists of 123 ARFCN codes (ARFCN 128 to ARFCN 251), in which the ARFCN 128 code represents the uplink frequency of 824.2MHz and the downlink frequency of 869.2MHz. In UMTS and LTE, the ARFCN are replaced with UARFCN and EARFCN respectively. The mapping of each ARFCN on the corresponding carrier frequency is given in [40].

4.2. Baseband in Mobile Phones

Modern mobile phones consist of at least two separate processors [9] [41]. The application processor runs the main operating system (e.g., Android or iOS) and is responsible for handling the graphical user interface, memory management and process scheduling. The baseband processor runs a dedicated RTOS which manages the radio communication and maintains the protocol stack. The application processor and the baseband processor work independently from one another and have separate memory space. However, it is necessary to exchange data between the two processors on a routine basis, for example, when the dialer application initiates a call (application processor to baseband processor) or when an SMS notification is received (baseband processor to application processor). Communication between the processors is commonly handled through a shared-memory segment or a dedicated serial interface [9] [41]. Unlike modern smartphones, low-end mobile phones, also referred to as feature-phones, employ a single processor to manage both user-interface and cellular communication. On feature-phones, this single processor is also referred to as a baseband processor.

4.2.1. Baseband Chip Architecture

The baseband processor is an integral part of the baseband chip. The chip consists of: (1) the RF frontend, (2) the analog baseband, (3) the digital baseband, and (4) the baseband processor [6] [41].

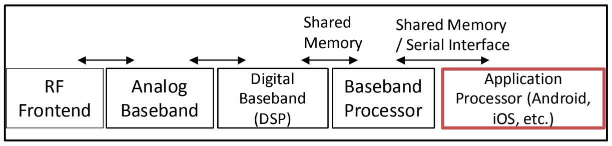

Figure 2: The baseband components and application processor in modern mobile phones. In low-end phones, an application processor doesn't exist.

The **RF frontend** handles received and transmitted signals on the physical level. This component consists of items such as: an antenna, a low-noise amplifier (LNA), and a mixer. The **analog baseband** contains, among other components, an analog to digital converter (ADC) and a digital to analog converter (DAC) to mediate between the digital baseband and the RF frontend. The **digital baseband** includes the digital signal processor (DSP) which is responsible for the lowest parts of the protocol stack (i.e., modulation/demodulation and error-correction). The **baseband processor** is responsible for handling the higher and more complex layers of the protocol stack. Communication between the DSP and the baseband processors takes place through a shared-memory interface (Figure 2).

5. The Transmitter

The physical effect underlying our transmission method is electromagnetic radiation (EMR), a form of energy emitted by certain electromagnetic processes. The emitted waves propagate through space in a radiant manner. Electromagnetic waves have two defining properties: the frequency f measured in Hertz (Hz) and the amplitude (i.e., strength) measured in decibel-milliwatts (dBm). In many cases, electronics (such as wiring, computer monitors, video cards, and communication cables) emit EMR in the radio frequency spectrum. Their frequencies and amplitudes depend on their internal currents and voltage. An exploitation of intentional and unintentional emissions from computer components has been addressed in previous research [14] [23] [13] [42].

We propose that a computer's memory bus can be exploited to act as an antenna capable of transmitting information wirelessly to a remote location. When data is exchanged between the CPU and the RAM, radio waves are emitted from the bus's long parallel circuits. The emission frequency is loosely wraps around the frequency of the RAM's I/O bus clock with a marginal span of +/-200MHz. The casual use of a computer does not generate these radio waves at significant amplitude, since it requires a major buildup of voltage in the circuitry. Therefore, we have found that by generating a continuous stream of data over the multi-channel memory buses, it is possible to raise the amplitude of the emitted radio waves. Using this observation, we are able to modulate binary data over these carrier waves by deterministically starting and stopping multi-channel transfers using special CPU instructions.

In the remainder of this section, we describe the design and implementation of the transmitter from the bottom up. First, we discuss the carrier wave (channel frequency) of the emitted radio waves. Next, we discuss a method for modulating binary data over a bus. Last, we propose a simple bit framing protocol to help the receiver demodulate the received signal. It is important to note that since the focus of this paper is the feasibility of the proposed covert channel, we do not exhaustively explore all possible signal modulations or bit framing protocols. Improvements to the communication protocol are a subject of future research.

5.1. EMR Emissions

Multi-channel memory architecture is a technology that increases the data transfer rate between the memory modules and the memory controller by adding additional buses in between them. The address space in multi-channel memory is spread across the physical memory banks, consequentially enabling data to be simultaneously transferred via multiple (two, three, or four) data buses. In this way, more data can be transferred in each read/write operation. For example, motherboards with dual-channel support have 2x64 bit data channels. Some computers support triple-channel memory and modern systems even have quadruple-channel support. Multi-channel architecture is implemented in all modern Intel and AMD motherboards.

In Figure 3, the radio emissions from an ordinary desktop workstation with dual channel memory are plotted on the frequency plane, comparing emissions from casual activity to those associated with intentional actions. When all channels are used, the radio emissions from the buses increase (red) in comparison to the emissions from casual activity (blue). We observed an increase of at least 0.1 - 0.15 dB across the frequency band 750-1000MHz, where some specific sub-bands showed an increase of about 1 - 2.1dB. A full summary of the radio emissions of different motherboards and memory technologies can be found in Table 3.

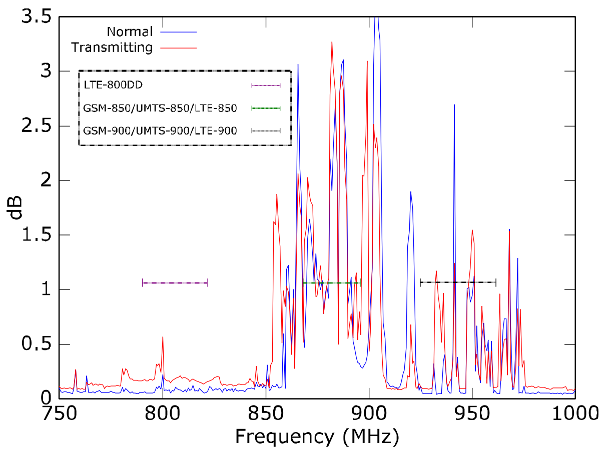

Figure 3: A plot of the amplitude of the radio waves emitted from a motherboard with an 800MHz I/O bus using DDR3-1600 RAM. Blue: casual use of the computer. Red: our transmission algorithm while using the dual channel data paths.

Based on our experiments, we have found that the use of three or four channels increases amplitude emissions across nearly the entire band depicted in Figure 3. This means that as the memory architectures mature, the quality of the proposed covert channel will increase. Note that these radio emissions fall within the frequency bands of GSM, UMTS and LTE, making them detectable by all modern basebands.

Standard Name	I/O bus clock (f_c)	EMR Range
DDR3-1600	800MHz	600MHz-1100MHz
DDR3-1866	933MHz	750MHz-1150MHz
DDR4-2133	1066MHz	750MHz-943MHz (fragmented) 1.04GHz-1.066GHz

Table 3: Summary of radio emissions from different memory buses.

5.2. Signal Modulation

In communications, modulation is the process where analog waveforms are varied to carry information over some medium. Typically, a carrier wave (for wireless a radio wave at the frequency f_c) is selected as the channel frequency, where most of the energy from the modulation can be found in the band around f_c.

There are many techniques for modulating a carrier wave to carry binary data. For simplicity and as a show of feasibility, we use a variant of the two level amplitude shift keying (B-ASK) modulation; to send a ‘1’ or ‘0’ the transmitter raises or lowers the amplitude of f_c accordingly over set time intervals T (in seconds) [43]. In other words, the time domain is partitioned into intervals of length T, and the symbol (i.e., signal amplitude) that corresponds to the current bit is transmitted over that entire interval. Our variation of B-ASK is that ‘0’ is not represented by a near zero amplitude, but rather by the average level of the casual emissions. It is assumed that the receiver can differentiate between average and high emission levels (described in detail later in Section 6). The motherboard bus’s radio emissions can be modulated to carry a B-ASK signal in the following way: to transmit a ‘1’ all memory channels are utilized for T seconds, and to transmit a ‘0’ nothing special is done (casual emissions are emitted). In this case, f_c is the motherboard’s memory clock.

5.3. Modulation Algorithm

In order to transmit a ‘1’, it is necessary to consistently utilize multiple memory channels for T seconds. To do this we generate a long random data transfer from the CPU to the main memory using the single instruction multiple data (SIMD) instruction set. SIMD utilizes special CPU registers of 64-bits and 128-bits in order to process wider chunks of data in a single instruction. SIMD instructions are usually used for vectorized calculations such as 2D/3D graphics processing, and includes instructions to load/store data between the main memory and special registers.

Algorithm 1 transmit32 (data)

```
1: $buffer \leftarrow$ ALIGNED_ALLOCATE(16,4096)
2: $tx\_time \leftarrow 500000$
3: for $bit\_index \leftarrow 0$ to 32 do
4:     if (data[$bit\_index$] = 1) then
5:         $start\_time \leftarrow$ CURRENT_TIME()
6:         while ($tx\_time >$ CURRENT_TIME() - $start\_time$) do
7:             $buffer\_ptr \leftarrow buffer$
8:             for $i \leftarrow 0$ to $buffer\_size$ do
9:                 SIMDNTMOV($buffer\_ptr$, 128$bit\_register$)
10:                $buffer\_ptr \leftarrow buffer\_ptr + 16$
11:            end for
12:        end while
13:    else
14:        SLEEP($tx\_time$)
15:    end if
16: end for
```

We implemented the B-ASK modulation algorithm using the Streaming SIMD Extension (SSE) instruction set found in Intel and AMD CPUs. The SSE specifies a set of 128-bit (quadword) registers numbered xmm0-xmm16, and includes a group of instructions for moving data between these xmm registers and the main memory [44] [45]. Using these instructions it is possible to instruct the CPU to utilize the multi-channel data paths, thereby amplifying the radio emissions.

One of the challenges we had to overcome resulted from the use of the CPU caching mechanisms. When the processor employs a cache hierarchy, transferring data between xmm registers and the main memory does not guarantee any immediate activity over the bus. This inconsistency presents an issue regarding the use of the proposed B-ASK modulation, since the symbols must start and stop precisely within the symbol interval (T).

Beginning with SSE version 2, there is a set of instructions that enable read/write operations directly to/from the main memory, while bypassing all cache levels (non-temporal). Specifically, we use the Move Double Quadword Non-Temporal instruction, MOVNTDQ *m128, xmm*. The intent of this instruction is for copying double quadwords from the xmm register to the 128-bit memory address, while minimizing pollution in the cache hierarchy.

Our implementation of the B-ASK modulation (Algorithm 1) works in the following way. The *transmit32()* method receives the outbound binary as an array of 32 bits. A temporary buffer of 4096 bytes (32x128) is allocated on the heap (lines 1-2) as a destination for the MOVNTDQ memory operations. Note that the allocated memory has to be 16-bytes aligned, as required for SSE memory operands. Next, on line 2, we set T to 500ms. Although a shorter T would provide a faster bit transmission rate, doing so directly increases the error rate. For the tested Motorola C123 phone with the Calypso baseband, a value of 500ms appears to provide satisfying results. Basebands of modern smartphones are probably capable of higher sampling quality, and therefore might require a shorter T. With specialized receiver hardware, setting T to 1-10ms provided good reception quality (Section 6).

The outer loop (line 3) iterates over the 32-bit array and performs the memory operations to generate the radio emissions. When the current bit is a '1' a loop repeatedly uses the MOVNTDQ instruction to copy data from xmm registers to the heap, until T seconds have elapsed. Conversely, when the current bit is a '0' the algorithm sleeps for T seconds.

5.4. Bit Framing

As mentioned earlier, when our variant of B-ASK modulates a '0' the amplitude of the transmitted signal is that of the bus's average casual emissions, and anything significantly higher than that (by some threshold) is considered a '1'. This incurs two issues: (1) the receiver has no prior information as to what the optimum threshold should be making it difficult for the receiver to detect activity in its area, and (2) the strength of amplitudes surrounding f_c is dependent on the distance between the transmitting desktop and the receiver; this means that if the mobile phone is moving during a transmission or other interference exists, a '1' and '0' can be decoded incorrectly.

Therefore, in order to assist the receiver in dynamically synchronizing with the transmitter, we place the data into frames. The binary stream is partitioned into sequential payloads of 12 bits, and the payloads are transmitted with a header consisting of the preamble sequence '1010' (Table 4). The preamble is used by the receiver to determine when a frame is being transmitted and to determine the amplitude levels of a '1' and a '0'. This process is discussed in depth in Section 6. The framing process takes place before data transmission. Once the frame has been built, it is passed to Algorithm 1 as the outbound data.

Preamble	*Payload*	*Preamble*	*Payload*
1010	12 bits	1010	12 bits

Table 4: The basic frame format used to send segments of a bit stream, using the *transmit32()* function.

5.5. Transmitter Stealth and Compatibility

The transmitting program has a small memory and CPU footprint, making the activities of the transmitter easier to hide. In terms of memory consumption, the program consumes merely 4K of memory allocated on the heap. In terms of CPU intake, the transmitter runs on a single, independent thread. At the OS level, the transmitting process can be executed with no elevated privileges (e.g., root or admin). Finally, the code consists of bare CPU instructions, avoiding API calls to escape certain malware scanners. In short, the transmission code evades common security mechanisms such as API monitoring and resource tracing, making it hard to detect.

As for compatibility, since 2004 SIMD instructions have been available for x86-64 Intel and AMD processors [46] [47], making the transmission method is applicable to most modern workstations and servers. Similar instructions on IBM's Power architecture have been in place since Power ISA v.2.03 was initiated [48]. The proposed transmitter has been implemented and successfully tested on several operating systems, including Microsoft Windows platform (Windows 7, 64bit), Linux Fedora 20 and 21 (64bit), and Ubuntu 12.1 (64bit).

6. The Receiver

In this section we describe how a mobile phone in close proximity to a transmitting computer can successfully receive and decode emitted signals. We implement the GSMem receiver component by modifying the firmware of a mobile phone's baseband. We present the receiver architecture and implementation, along with the modulation and decoding mechanisms. Interestingly, we found that under certain circumstances, the GSMem signals can be indirectly received by an application running on a modern Android smartphone with a non-modified baseband. This optional implementation yields rather limited effective distance of 10cm, and provides a conceptual rather than an immediate practical contribution. Therefore, to stay in line with the core of this paper, the description of this implementation is deferred to Appendix A.

6.1. Receiver Implementation

Reception of the transmitted data is accomplished in the following manner: (1) sample the amplitude of the carrier wave, (2) performs noise mitigation, (3) search for bit frame header (preamble detection), and (4) demodulate the frame's payload. We will describe each of these steps in this order after discussing the implementation framework.

6.1.1. Baseband Firmware

As discussed in Section 1, the baseband industry is highly protective, keeping information about baseband architecture, the RTOS, and the protocol stack, guarded from the public [9] [10] [49]. The secrecy and complexity of the baseband technology makes it extremely difficult to make modifications at the binary level, particularly without the availability of information such as source code [10] [49]. However, there have clearly been cases where attackers have used explicit access to the device firmware in order to perform malicious activities [29] [31] [33] [50]. Our implementation of the GSMem receiver is based on '*OsmocomBB,'* an open source GSM baseband software implementation [51].

The open source project, launched in 2010, is the only way to freely examine the implementation of a mobile's GSM baseband software. OsmocomBB provides source code for the GSM protocol stack, along with device drivers for digital and analog basebands chips. The project currently supports about 13 models of mobile phones. Most of the supported phones are OEM by Motorola and works with Calypso baseband chipsets made by Texas Instruments. For our experiments, we selected the Motorola C123 model [52] that supports 2G bands but has no GPRS, Wi-Fi, or mobile data traffic capabilities. The Motorola C123 is a limited mobile phone, supporting our attack scenario described in Section 3. It is worthwhile to note that the baseband components of modern smartphones are more advanced in terms of RF reception, sampling rate and processing power due to the improved hardware and the support in new technologies such as the LTE [6] [53]. That means that implementation of the GSMem receiver on modern device may yield better results in terms of reception quality and transfer-rates.

The GSM protocol stack at the baseband consists of three main layers [49]. Layer 1 is the most relevant layer in term of GSMem implementation. It handles the RF interface which modulates the data over the air. In OsmocomBB, the lower part of the layer 1 is handled by the DSP, while the baseband processor handles the upper layers. Layer 1 includes, among other functionalities, the power management, which is responsible for acquiring the raw signal power measurements (in dBm) of specific frequencies (ARFCNs). Note that measuring RF power levels is a basic functionality of any baseband chip [39]. The interaction between the baseband processor and the DSP is depicted in Figure 4.

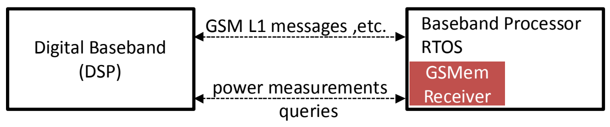

Figure 4: Interaction between the baseband processor and the DSP.

6.1.2. Firmware Modification

The receiver is implemented by patching the main event handler in the baseband RTOS. Figure 5 shows the outline of the OsmocomBB initialization and main loop. After initialization (lines 1-2), the baseband processor enters the event loop (line 3). The event loop continuously processes a sequence of event handlers, including the keypad handler, timer updates, and layers 2 and 3 handlers, interrupts from the DSP, power measurements, etc.

```
1: // baseband initialization
2: // specific receiver initialization
3: while true do
4:     // keypad handlers
5:     // layer 2/3 handlers
6:     // power management handler
7:     // ...
8:     RECEIVERHANDLER()
9: end while
```

Figure 5: Calypso RTOS code outline

In order to implement the functionalities of the receiver, we added a routine of our own called *ReceiverHandler()* (line 8). Since it is placed in the main loop, the routine is run continuously at every iteration.

The *ReceiverHandler()* has three possible states: (1) scan for best frequency (2) search for bit frame header (preamble), and (3) B-ASK signal demodulation. Scan state is the initial state of the routine. The pseudo code for *ReceiverHandler()* is presented in Algorithm 2.

Algorithm 2 ReceiverHandler

```
1: $dBm \leftarrow$ MEASURE($f_c$)
2: $filtered\_signal \leftarrow$ UPDATEMOVINGAVERAGE($dBm$)
3: if ($state = SCAN$) then
4:     $f_c \leftarrow$ SCANFREQ()
5:     SETSTATE($PREAMBLE$)
6: end if
7: if ($state = PREAMBLE$) then
8:     if (IDENTIFYPREAMBLE($filtered\_signal$) $= true$) then
9:         SETSTATE($RECEIVE$)
10:    end if
11: end if
12: if ($state = RECEIVE$) then
13:     $b \leftarrow$ DEMODULATEBIT($filtered\_signal$)
14:     $bitSequence$.add($b$)
15:     if ($bitSequence$.size%16 $= 0$ or SIGNALLOST($filtered\_signal$)) then
16:         SETSTATE($PREAMBLE$)
17:     end if
18: end if
```

6.1.3. Signal Sampling

The first step in detecting a GSMem transmission is to sample the amplitude of the carrier wave f_c. Note that this step takes place only after f_c has been determined in a initial scanning phase. Each time the main loop runs *ReceiverHandler(),* Algorithm 2 causes the DSP module to sample the power level (amplitude) of f_c (line 1) and stores it in a buffer (line 2). This data is used later in the demodulation routines. The function *Measure()* invokes an amplitude measurement request on the DSP using a function called *l1a_l23_rx()*. The DSP measurements are performed in bands of 0.2MHz. Our tests show that the tested Calypso baseband was able to sample power measurements at a rate of 1.8kHz, hence 1.8kbps is the fastest bit rate that this device can demodulate at. This is a much faster bit rate than we achieved due to the limited processing capabilities of the device. However, the power measurements rate is an important consideration to take into account when implementing an improved GSMem receiver on a more advanced device in the future.

6.1.4. Noise Mitigation

After the power measurement, a noise mitigation function is applied to the current sample by averaging it with the last W original samples. This operation is essentially a moving average filter, an effective technique for mitigating high frequency noise. In our experiments with the Motorola C123, we tried a W of 50-750 samples and found that the size of W directly affects the bit rate. A larger W provided better noise mitigation, while a smaller one produced a faster bit rate.

6.1.5. Detecting the Best Carrier Wave

In the *SCAN* state, the receiver searches for the best f_c to use for demodulating GSMem transmissions. Note that since the radio emissions of the transmitter fallout across the GSM-850/GSM-900 bands (Figure 3, Section 5.1), the f_c can be set in advance to any frequency in those bands. However, we observed that some frequencies have more interference than others (e.g., the channels actively used by nearby cellular base stations). Therefore, during the scanning state, the better f_c is determined as the frequency that provides the best carrier to interference ratio (CIR). This frequency is found by scanning the range of the entire GSM-850 range and selecting the frequency with the minimum average amplitude (in dBm). The assumption is that the minimum average amplitude indicates a low level of interferences, making it easier to detect a ‘1’ using our variant of B-ASK. In our implementation, the scanning takes place after the device boots, and after every 30 seconds of noisy or lost signals. After the f_c value is set, the algorithm moves to the *PREAMBLE* state.

6.1.6. Preamble Detection and Demodulation

If state is set to *PREAMBLE*, the receiver searches for a preamble sequence (lines 7-11 of Algorithm 2). If the sequence ‘1010’ is detected, then it is assumed to be the start of a frame, and state is changed to *RECEIVE* to complete the B-ASK demodulation process (lines 12-18). The preamble sequence allows the GSMem receiver to: (1) synchronize with the GSMem transmitter (2) identify ‘1’ and ‘0’ amplitude levels δ and (3) determine the signals’ duration t, if unknown. Dynamically setting δ for every frame is necessary for demodulating signals while the mobile is moving. For example, a frame may be received at close proximity to the transmitter where f_c is much stronger thereby setting amplitude levels to be high. The subsequent frame may be sent while the mobile phone is farther away – where smaller amplitude would be more appropriate. Once a preamble has been detected, the payload is demodulated in a similar manner using the updated parameters.

6.1.7. Signal Loss

On line 15 in Algorithm 2, the state of the receiver returns to *PREAMBLE* if the whole payload has received, or if the signal has been lost. The function *SignalLost()* returns true if during the data reception, the measured signal power is weaker than the amplitude of the '0's from the preamble for three seconds straight. In this case, any partially received data discarded or marked appropriately.

7. Evaluation

In this section we evaluate GSMem's performance as a communication channel. We present in detail the evaluation using a tampered cellular baseband receiver. We also examine the signal reception using a dedicated hardware receiver programmed via software defined radio (SDR).

7.1. Experiment Setup

We used the Motorola C123 with the modified firmware as the receiver for all experiments in this section. As for the transmitters, we used three different models of desktop workstations (WS), each with a different configuration and different case. The details of these computers and their tested settings can be found in Table 5. Note that WS3 is a much stronger transmitter than the others since its RAM has a quad channel operation mode, which employs wider data paths. In all the experiments, the transmitter used the 4kb allocation method described in Section 5, with a T of 1.8 seconds. The receiver listened to the carrier frequency (f_c) ARFCN 25 downlink (940MHz), unless otherwise mentioned.

	WS1	WS2	WS3
OS	Linux Fedora 20		
Chassis (metal)	infinity chassis	GIGABYTE Setto 1020 GZ-AX2CBS	Silverstone RL04B
CPU	Intel i7-4790	Intel i7-3770	Intel i7-5820K
Motherboard	GIGABYTE GA-h87M-D3H	GIGABYTE H77-D3H	GIGABYTE GA-X99-UD4
RAM Type	2 x 4GB 1600MHz		4 x 4GB 2133MHz
RAM Frequencies Tested	1333/1600 MHz		1833/2133 MHz
RAM Operation Modes Tested	Single / Dual		Dual / Quad

Table 5: Configuration of the transmitting workstations.

There are several major factors that affect the quality of a wireless communication channel. Typically, the quality of a channel is measured by taking the signal to noise ratio (SNR), where $SNR \equiv 10\log(P_{signal}/P_{noise}) = P_{signal}dB - P_{noise}dB$ and P is the power level (a larger SNR is better than a smaller one). The noise power P_{noise} can originate from naturally occurring noise and from other interferences such as the emissions from nearby computers in the same office space. Therefore, in order to match our attack scenario from Section 3, the experiments in this section all take place in a regular work space with several active desktop workstations within a 10m radius.

There are many factors which can decrease the SNR of a wireless channel when the location of the receiver is changed. Because we are dealing with a low power transmission, we do not consider properties such as multipath propagation (fading). Instead, we focus on how different receiver distances and positions affect the channel's SNR.

7.2. Channel Signal to Noise Ratio (SNR)

The first set of experiments tests the SNR of the WSs from different distances. Figure 6, Figure 7 and Figure 8 show the receiver's maximum measured amplitudes at different distances from WSs 1, 2, and 3 respectively. Here, WSs 1 and 2 have their RAM set to dual mode at 1600MHz, and WS3 has its RAM set dual / quad mode at 1833 / 2133MHz. As illustrated by Figure 9, the SNR remains positive (more signal power than noise) even up to a distance of 160cm. This gives a good indication of the proposed covert channel's effective distance. Given these observations, we assume that a distance of 160cm from a workstation is within the normal range where a mobile device is expected to be held while working on the workstation.

Note that WS3 in dual mode has a significant advantage in range over WSs 2 and 3. This is due to the fact that WS3 uses a higher RAM frequency than all other WSs in the workplace scenario. This means that it is subject to less interference, thereby improving its SNR. When quad channel mode is used, the range increases further, demonstrating that a higher number of active memory channels increases the signal's amplitude.

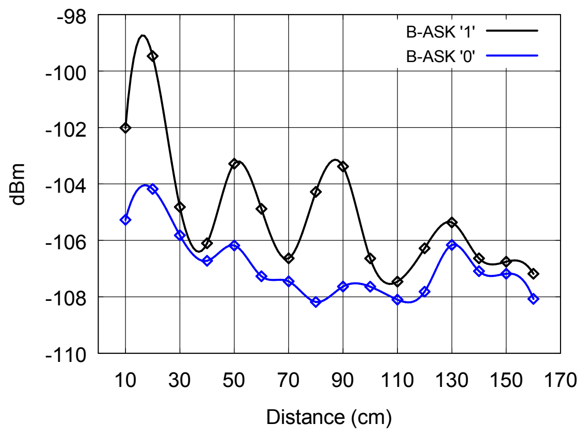

Figure 6: Signal strength received from WS2 (1600MHz, Dual) at various distances from the backside of the chassis. The blue line can also be viewed as the casual emissions (noise).

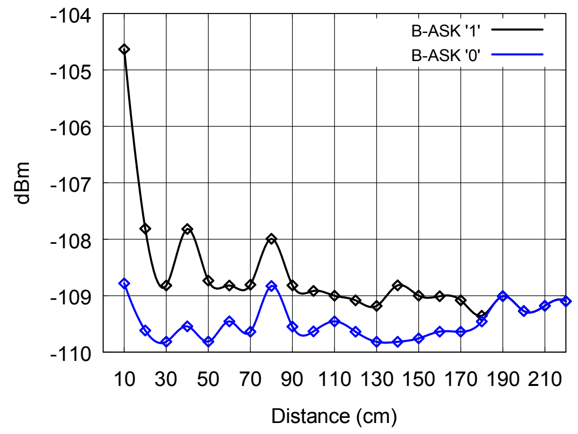

Figure 7: Signal strength received from WS1 (1600MHz, Dual) at various distances from the backside of the chassis.

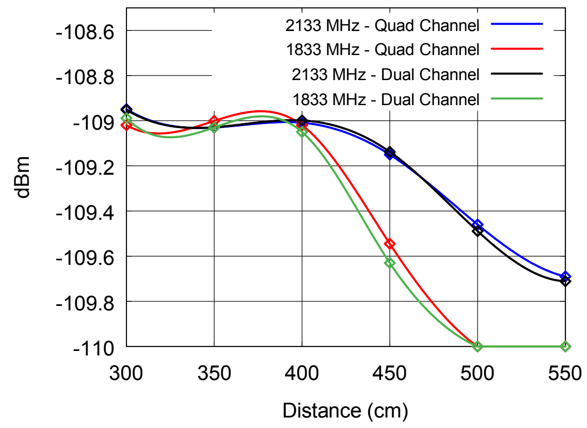

Figure 8: Signal strength received from WS3 (1833/2133MHz, dual/quad channels) at various distances from the front side of the chassis.

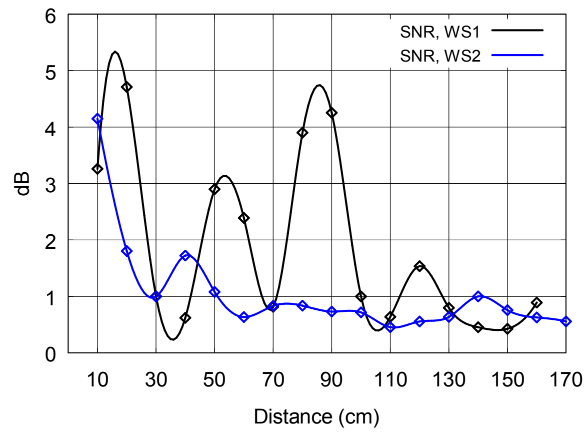

Figure 9: Receiver SNR from WS1 and WS2 (1600MHz, Dual) at various distances from the backside of the chassis.

During the experiments, we observed that the position of the receiver with respect to the transmitter has a significant impact on the SNR. For instance, using WS2, an SNR of 0.5 is achieved at a farther distance from the front of the chassis as opposed to the back. Furthermore, the best position for WS1 (using 1600MHz) is from the front, while the best position for WS2 is from the back. These differences make sense considering that each case has variations in shape and metal content. In all cases, we observed that the optimum position for the receiver to be is in front of the chassis. This may have to do with the fact that the front of an ATX case is mainly made of plastic (blocking less of the signal).

Figure 10 and Figure 11 depict the distance at which an SNR of 0.5dB can be achieved at different positions around the WSs.

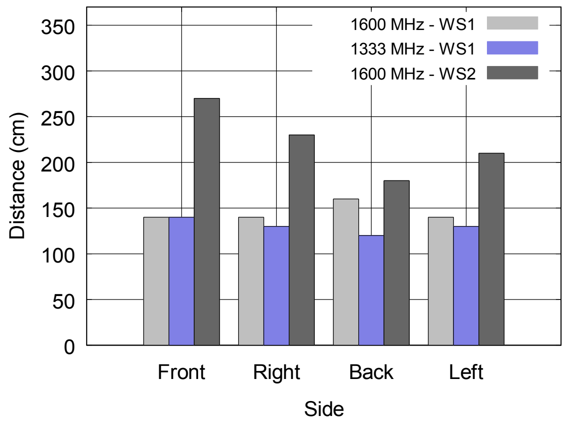

Figure 10: The distance at which an SNR of 0.5dB is achieved at various positions around the transmitters WS1 and WS2 using dual mode and different clock speeds

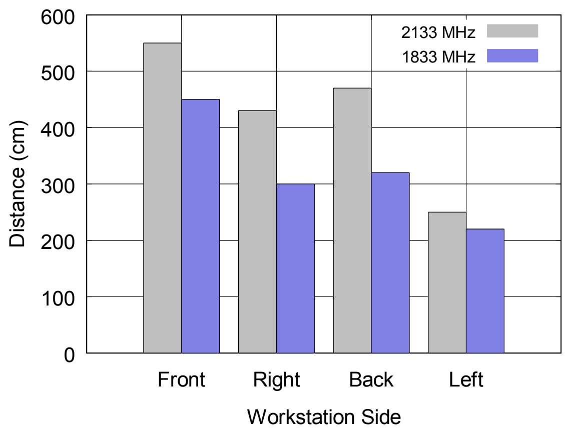

Figure 11: The distance at which at least 0.5dB of SNR is achieved at various positions around the transmitter WS3 using quad mode and different clock speeds.

7.3. Bit Rates

The GSMem receiver implemented using OsmocomBB on the nine year old mobile phone significantly limit the channel's quality. Although this device provides the advantage of GSM baseband programmability, it has limited real-time processing power and inadequate access to the DSP's full capabilities. Due to these limitations, we preferred using simple ASK type modulations over other more sophisticated options. Using the proposed B-ASK modulation with this device, we were able to receive binary data from the GSMem transmitter at a bit rate of 1 to 2 bit/s. This allows exfiltration of small amounts of information such as identifiers, passwords, and encryption keys, within several minutes. We examined the bit error rate (BER) by transmitting a set of 256-bit encryption keys from a workstation. Figure 12 depicts the BER over varying distances between the transmitting workstation and a nearby mobile phone.

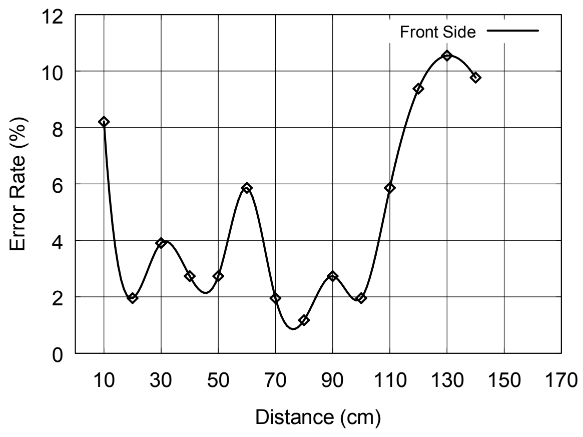

Figure 12: The Motorola C123's BER plot from a B-ASK transmission using WS1 as the transmitter.

7.4. Software Defined Radio (SDR)

Much higher bit rates - at even further distances - are achievable when more modern equipment is used and the full capabilities of the baseband component are accessible. To demonstrate this fact, we implemented a GSMem receiver using GNU-Radio software on an affordable SDR kit; the Ettus Research Universal Software Radio Peripheral (USRP) B210 [54], which is capable of capturing data at velocities up to 32 million samples per second. The USRP was connected to Lenovo ThinkPad T530 (through the USB 3.0 interface), with dedicated software suitable for capturing signals from the USRP, i.e. GNU-Radio v3.7.5.1. The OS is Linux Ubuntu 14.10 (64 bit).

Since we had full access to the DSP's capabilities, we implemented the receiver using the frequency shift keying modulation scheme (FSK) where a '1' and '0' were modulated by using two distinct frequencies. Creating two carrier waves was accomplished by adding a slight delay inside the memory transfer operation loop. Since this version of the GSMem transmitter was not implemented on a cellular device, we omit the rest of its details from the body of this paper. Using this hardware, we were able to improve the signal quality and the reception distance significantly. At a distance of 2.6m and where $T = 0.001$, we achieved a bit rate of 1000 bit/s, with a BER of approximately 0.087%. Table 6 summarizes the time needed to transfer certain pieces of sensitive information at the rates of T=0.5 (using Motorola C123) and T=0.001 (using USRP).

Data	Length (bit)	Rx Time Motorola C123	Rx Time USRP
MAC Address	48	30 sec	48 ms
Plain Password	64	40 sec	64 ms
MD5	128	1.3 sec	128 ms
GPS Coordinate	128	1.3 sec	128 ms
SHA1 Hash	160	1.6 min	160 ms
Disk Encryption Key	256	2.6 min	256 ms
RSA Private Key	2048	21.3 min	2.04 sec
Fingerprint Template	2800	29.1 min	2.8 sec

Table 6: Transmission times

In order to increase the effective distance, we used a directed printed circuit board (PCB) log periodic antenna [55], optimized for capturing signals at the range of 400 MHz – 1000 MHz. The antenna connected to the USRP via its standard connectors.

We measured the signal levels of '1' and '0' emitted from a transmitting WS3 over varying distances. The transmitter resides in a regular work space with several active desktop workstations situated within a 10m radius. As can be seen in Figure 13, the signals were received in 30 meters and beyond. This is a significant improvement when compared to the mobile phone receiver. Furthermore, these results were obtained with a rather affordable hardware receiver, using commonly available components.

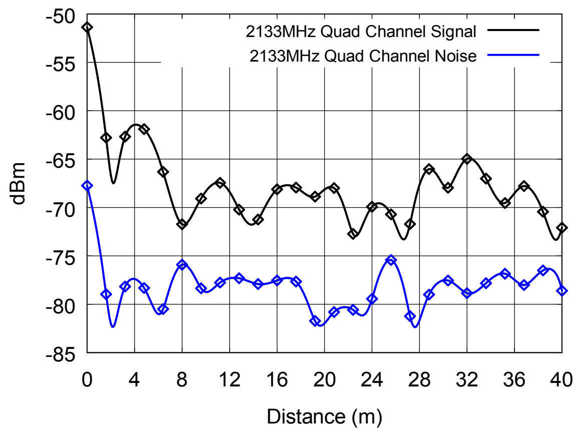

Figure 13: Signal strength received on f_{c1} as transmitted from WS3 at distances of 0-40 meters from the front side of the chassis.

8. Countermeasures

Official governmental and military standards concerning EMSEC countermeasures are mainly classified, despite some occasional leaks [16], [56]. With the exfiltration method described in this paper, the "zones" approach may be used as a countermeasure, defining spatial regions where mobile phones, including simple devices, are prohibited. As discussed earlier, however, the signal reception distance may grow when dedicated hardware receiver is being used. In this context, the insulation of partition walls may help. Structural building elements, such as reinforced concrete floors, seem to provide insulation by acting as a Faraday cage. However, enclosing each computer within a Faraday cage seems impractical. Shielding the transmitting component within the computer, i.e., the multi-channel memory bus is a challenging task, particularly when compared to shielding other emanation sources, such as monitor cables. Another defensive strategy may involve behavioral (dynamic) analysis and anomaly detection, trying to detect GSMem activities at runtime on the process level [9] [57]. However, when the baseband firmware is utilized as the GSMem receiver, it is particularly hard to detect because of the separation of the baseband component from the main operating system [49]. In this case, a meticulous forensic analysis of the device may be required.

9. Conclusion

In this paper we present GSMem, a method for exfiltrating data from air-gapped networks. Our major contributions include a unique covert channel, consisting of a feasible transmitting method, and a ubiquitous receiver that doesn't arouse suspicion. The covert channel is based on electromagnetic waves emitted at frequency bands of GSM, UMTS and LTE cellular networks. The transmitting software exploits specific memory-related CPU instructions, utilizing the multi-channel memory bus to amplify the transmission power. Subsequently, the transmitted signals are received and demodulated by a rootkit residing at the baseband level of a cellular phone. Note that, unlike some other recent work in this field, GSMem exploits components that are virtually guaranteed to be present on any desktop/server computer and cellular phone. Furthermore, elementary cellular phones, those without Wi-Fi, camera, or other nonessential instrumentation, are often allowed into classified facilities, even in security-aware organizations. We provide essential technical background information about cellular networks and an overview of baseband components in mobile phones. Next, we discuss the design considerations of the transmitter and the receiver, regarding signal generation, data modulation, transmission detection, noise mitigation, and handling a moving receiver. Our GSMem transmission software - implemented on Windows and Linux - has a small computational footprint, which makes it hard to detect. The GSMem receiver is implemented on a mobile phone, by modifying the baseband firmware of a low-end device. We present its architecture and discuss its capabilities and limitations. We go on to evaluate the method's using extensive configurations, settings, and various parameters. Our current results demonstrate the overall feasibility of the method, at a distance of 1-5.5 meters when using a standard cellular baseband receiver. We also evaluated the wider boundaries of GSMem using a dedicated yet affordable hardware receiver. The associated experiments yielded an effective distance of 30 meters and beyond. We believe that exposing this new covert channel will serve to raise professional awareness and academic interest.

References

[1] GReAT team, "A Fanny Equation: "I am your father, Stuxnet"," Kaspersky Labs' Global Research & Analysis Team, 17 2 2015. [Online]. Available: https://securelist.com/blog/research/68787/a-fanny-equation-i-am-your-father-stuxnet/.

[2] A. Gostev, "Agent.btz: a Source of Inspiration?," SecureList, 12 3 2014. [Online]. Available: http://securelist.com/blog/virus-watch/58551/agent-btz-a-source-of-inspiration/.

[3] N. Shachtman, "Under Worm Assault, Military Bans Disks, USB Drives," Wired, 19 11 2008. [Online]. Available: http://www.wired.com/2008/11/army-bans-usb-d/.

[4] IT@Intel White Paper, IT Best Practices, *Enabling Smart Phones in Intel's Factory,* Intel IT, 2011.

[5] L. Martin, "Important Information," Lockheed Martin, [Online]. Available: http://www.lockheedmartin.com/us/atl/maps/cherryhill/information.html. [Accessed 17 2 2015].

[6] H. Welte, "Anatomy of contemporary GSM cellphone hardware," Unpublished paper, c, 2010.

[7] M. Degrasse, "Broadcom looks to sell baseband unit," RCRWirelessNews, 2 6 2014. [Online]. Available: http://www.rcrwireless.com/20140602/wireless/broadcom-exploring-sale-baseband-unit.

[8] R. P. Weinmann, "All your baseband are belong to us," hack. lu., 2010.

[9] R. P. Weinmann, "Baseband Attacks: Remote Exploitation of Memory Corruptions in Cellular Protocol Stacks," in *WOOT*, 2012.

[10] R. P. Weinmann, "Baseband exploitation in 2013: Hexagon challenges," in *Pacsec 2013*, Tokyo, Japan, 2013.

[11] R. J. Anderson, "Emission security," in *Security Engineering, 2nd Edition*, Wiley Publishing, Inc., 2008, pp. 523-546.

[12] R. J. Aldrich, "Shootdowns, Cyphers and Spending," in *GCHQ – The Uncensored Story of Britains Most Secret Intelligence Agency*, Harper Press, 2010, pp. 201-226.

[13] W. van Eck, "Electromagnetic Radiation from Video Display Units: An Eavesdropping Risk?," *Computers and Security 4,* pp. 269-286, 1985.

[14] M. G. Kuhn and R. J. Anderson, "Soft tempest: Hidden data transmission using electromagnetic emanations," in *Information Hiding*, 1998, pp. 124--142.

[15] M. G. Kuhn, "Compromising emanations: Eavesdropping risks of computer displays," University of Cambridge, Computer Laboratory, 2003.

[16] J. McNamara, "The Complete, Unofficial TEMPEST Information Page," 1999. [Online]. Available: http://www.jammed.com/~jwa/tempest.html. [Accessed 4 10 2013].

[17] E. Thiele, "Tempest for Eliza," 2001. [Online]. Available: http://www.erikyyy.de/tempest/. [Accessed 4 10 2013].

[18] S. S. Clark, B. Ransford, A. Rahmati, S. Guineau, J. Sorber, K. Fu and W. Xu, "WattsUpDoc: Power side channels to nonintrusively discover untargeted malware on embedded medical devices," in *USENIX Workshop on Health Information Technologies (Vol. 2013)*, 2013.

[19] U. Rührmair, X. Xu, J. Sölter, A. Mahmoud, M. Majzoobi, F. Koushanfar and W. Burleson, Efficient power and timing side channels for physical unclonable functions, Springer Berlin Heidelberg, 2014.

[20] T. Halevi and N. Saxena, "A closer look at keyboard acoustic emanations: random passwords, typing styles and decoding techniques," in *ACM Symposium on Information, Computer and Communications Security*, 2012.

[21] M. Hanspach and M. Goetz, "On Covert Acoustical Mesh Networks in Air.," *Journal of Communications,* vol. 8, 2013.

[22] R. Callan, A. Zajic and M. Prvulovic, "A Practical Methodology for Measuring the Side-Channel Signal Available to the Attacker for Instruction-Level Events," in *Microarchitecture (MICRO), 2014 47th Annual IEEE/ACM International Symposium on*, IEEE, 2014, pp. 242-254.

[23] G. Mordechai, G. Kedma, A. Kachlon and Y. Elovici, "AirHopper: Bridging the air-gap between isolated networks and mobile phones using radio frequencies," in *Malicious and Unwanted Software: The Americas (MALWARE), 2014 9th International Conference on*, IEEE, 2014, pp. 58-67.

[24] M. Hanspach and M. Goetz, "Recent Developments in Covert Acoustical Communications.," in *Sicherheit*, 2014, pp. 243-254.

[25] M. Guri, M. Monitz, Y. Mirski and Y. Elovici, "BitWhisper: Covert Signaling Channel between Air-Gapped Computers using Thermal Manipulations," in *arXiv preprint arXiv:1503.07919*, 2015.

[26] P. John-Paul, "Mind the gap: Are air-gapped systems safe from breaches?," Symantec, 5 May 2014. [Online]. Available: http://www.symantec.com/connect/blogs/mind-gap-are-air-gapped-systems-safe-breaches.

[27] C. A., Q. Zhu, P. R. and B. T., "An impact-aware defense against Stuxnet," in *American Control*, 2013.

[28] J. Larimer, "An inside look at Stuxnet," IBM X-Force, 2010.

[29] D. Goodin, "Meet "badBIOS," the mysterious Mac and PC malware that jumps airgaps," ars technica, 31 10 2013. [Online]. Available: http://arstechnica.com/security/2013/10/meet-badbios-the-mysterious-mac-and-pc-malware-that-jumps-airgaps/.

[30] J. Zaddach, A. Kurmus, D. Balzarotti, E.-O. Blass, A. Francillon, T. Goodspeed, M. Gupta and I. Koltsidas, "Implementation and implications of a stealth hard-drive backdoor," *Proceedings of the 29th Annual Computer Security Applications Conference,* pp. 279-

288, 2013.

[31] D. Goodin and K. E. Group, "How “omnipotent” hackers tied to NSA hid for 14 years—and were found at last," ars technica, 2015.

[32] J. Linden, "DeathRing: Pre-loaded malware hits smartphones for the second time in 2014," Lookout, 4 December 2014. [Online]. Available: https://blog.lookout.com/blog/2014/12/04/deathring/.

[33] M. Kelly, "MouaBad: When your phone comes pre-loaded with malware," Lookout, 11 April 2014. [Online]. Available: https://blog.lookout.com/blog/2014/04/11/mouabad/.

[34] M. Guri, G. Kedma, A. Kachlon and Y. Elovici, "AirHopper: Bridging the air-gap between isolated networks and mobile phones using radio frequencies," in *Malicious and Unwanted Software: The Americas (MALWARE), 2014 9th International Conference on*, IEEE, 2014, pp. 58-67.

[35] RSA Research Labs, "Anatomy of an Attack," 1 4 2011. [Online]. Available: https://blogs.rsa.com/anatomy-of-an-attack/.

[36] J. Scahill and J. Begley, "THE GREAT SIM HEIST: HOW SPIES STOLE THE KEYS TO THE ENCRYPTION CASTLE," TheIntercept, 19 February 2015. [Online]. Available: https://firstlook.org/theintercept/2015/02/19/great-sim-heist/.

[37] F. Obermaier, H. Moltke, L. Poitras and J. Strozyk, "Snowden-Leaks: How Vodafone-Subsidiary Cable & Wireless Aided GCHQ’s Spying Efforts," sueddeutsche, 25 November 2014. [Online]. Available: http://international.sueddeutsche.de/post/103543418200/snowden-leaks-how-vodafone-subsidiary-cable.

[38] D. Sanger and N. Perlroth, "Bank Hackers Steal Millions via Malware," NY times, 14 February 2015. [Online]. Available: http://www.nytimes.com/2015/02/15/world/bank-hackers-steal-millions-via-malware.html?_r=0.

[39] "Digital cellular telecommunications system (Phase 2+), Radio transmission and reception 12.4.0 12," ETSI 3GPP, January 2015. [Online]. Available: http://www.etsi.org/deliver/etsi_ts/145000_145099/145005/12.04.00_60/ts_145005v120400p.pdf.

[40] Cellmapper, "Frequency Calculator," Cellmapper, [Online]. Available: https://www.cellmapper.net/arfcn.

[41] M. Shiraz, M. Whaiduzzaman and A. Gani, "A study on anatomy of smartphone," *Computer Communication \& Collaboration,* vol. 1, pp. 24-31, 2013.

[42] S. S. a. M. H. a. R. B. a. S. J. a. F. K. a. X. W. Clark, "Current events: Identifying webpages by tapping the electrical outlet," in *Computer Security--ESORICS 2013*, Springer, 2013, pp. 700-717.

[43] G. Patents, "Frequency shift keying". Patent US Patent 2,461,456, 8 February 1949.

[44] Intel, "Intel® Streaming SIMD Extensions 2 Store Intrinsics," Intel, [Online]. Available: https://software.intel.com/sites/products/documentation/doclib/iss/2013/compiler/cpp-lin/GUID-19F086CA-B0AE-4FC0-B5B5-A99AD5D62CFE.htm.

[45] AMD, "AMD64 Architecture Programmer’s Manual 128-Bit and 256-Bit XOP and FMA4 Instructions," 11 2009. [Online]. Available: http://support.amd.com/TechDocs/43479.pdf.

[46] "Intel Instruction Set Architecture Extensions - Advanced Vector Extensions," Intel, [Online]. Available: https://software.intel.com/en-us/intel-isa-extensions#pid-16007-1495.

[47] AMD, "All SIMD All the Time," AMD, September 2007. [Online]. Available: http://developer.amd.com/community/blog/2007/09/10/all-simd-all-the-time/.

[48] IBM, "Power ISA™," 3 May 2013. [Online]. Available: https://www.power.org/wp-content/uploads/2013/05/PowerISA_V2.07_PUBLIC.pdf.

[49] H. Welte, "Anatomy of contemporary GSM cellphone hardware," unpublished paper, c, 2010.

[50] SRlabs , "Turning USB peripherals into BadUSB," [Online]. Available: https://srlabs.de/badusb/.

[51] OsmocomBB, "OsmocomBB," [Online]. Available: http://bb.osmocom.org/trac/. [Accessed 13 1 2015].

[52] "Motorola C123," GSM arena, [Online]. Available: http://www.gsmarena.com/motorola_c123-2101.php.

[53] C. Turner, "New CortexTM - R Processors for LTE and 4G Mobile Baseband," ARM, 22 February 2011. [Online]. Available: http://arm.com/files/pdf/new_cortex-r_processors_for_lte_and_4g_mobile_baseband.pdf.

[54] "Ettus Research," [Online]. Available: http://www.ettus.com/.

[55] "LP0410 Antenna," Ettus Research, [Online]. Available: http://www.ettus.com/product/details/LP0410.

[56] USAF, "AFSSI 7700: Communications and information emission security," Secretary of the Air Force, 2007.

[57] M. Guri, G. Kedma, B. Zadov and Y. Elovici, "Trusted Detection of Sensitive Activities on Mobile Phones

Using Power Consumption Measurements," *Intelligence and Security Informatics Conference (JISIC), 2014 IEEE Joint,* pp. 145-151, 2014.

[58] ETSI, "Digital cellular telecommunications system: Radio subsystem link control," ETSI, July 1996. [Online]. Available: http://www.etsi.org/deliver/etsi_gts/05/0508/05.01.00_60/gsmts_0508v050100p.pdf.

Appendix A

Receiver Implementation (*Android Application Level*)

In this appendix we show how, under certain circumstances, the GSMem signals can be received by an application run on a modern Android smartphone with an untampered baseband. This technique is limited to close proximity to the transmitter (10cm).

Android Radio Interface Layer (RIL)

Android, as part of its open source framework, defines the upper software layers with respect to its hardware peripherals. In Android, the Radio Interface Layer (RIL) component interfaces between high level telephony services (*android.telephony*) and the baseband hardware. Each vendor supplies its own implementation for the RIL interface. The vendor RIL is closed source and shipped with the stock Android firmware as a shared object (.so) binary file.

Signal Demodulation

We developed a reception method which we refer as *'neighbor cell jamming'*. According to the GSM standard, mobile equipment must periodically listen to the broadcasted pilot channels of neighboring cells in order to provide service reliability [58]. Generally, the mobile must always be registered to a cell preferably the one with the best reception. These broadcasts are sent over logical channels called broadcast control channels (BCCH), which carry information such as that cell's ID and configuration. The GSM baseband component maintains a list of best neighboring cells along with their received power level (in dBm or equivalent units) and other information. Since GSMem operates at the same frequency as the neighboring BTSs, it is possible for a GSMem transmitter to affect a drop in the reception of a station that is rather far away. This jamming effect can be used as a side channel to detect the B-ASK modulation such a sudden drop in reception quality represent a '1' and otherwise a '0'.

Implementation

Android allow obtaining the neighboring cells' information from the baseband. E.g., by invoking the method *telephonyManager.getNeighboringCellInfo()*.It includes the received signal strength indication (RSSI) of each neighboring cell. Our Android application repeated an algorithm similar to Algorithm 2 (Section 6) with a few modifications. It continuously sampled and stored the signal strength of the weakest cell out of the neighboring cells (the cell which our transmission will likely override). The modulation is inversed: low RSSI represents '1' (transmission occurred) and high RSSI represents '0' (no transmission). Figure 14 shows the reception of a single bit, as received by our application on the Samsung Galaxy S5 smartphone. The phone was located 10cm away from a transmitting workstation. The 'jammed' cell had signal strength of 23asu (equal to -67dBm) before it was jammed. At second 6, the GSMem at the workstation transmit '1', causing a drop in the RSSI measurement for that cell. The transmission stops at second 8.

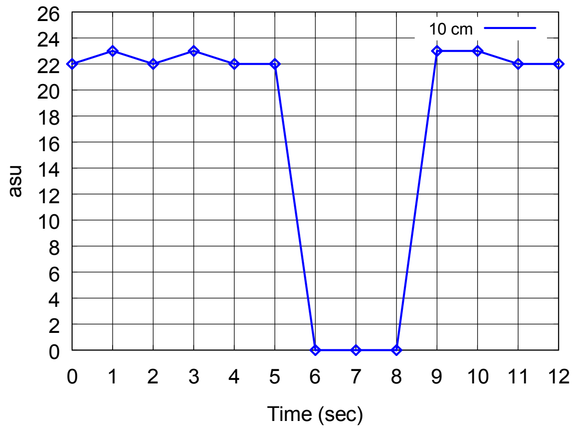

Figure 14: Neighbor cell reception level during transmission of a single bit.