Even Black Cats Cannot Stay Hidden in the Dark:
Full-band De-anonymization of Bluetooth Classic Devices

Marco Cominelli, Francesco Gringoli
CNIT/University of Brescia, Italy

Paul Patras
The University of Edinburgh, Scotland

Margus Lind
Context Information Security, Scotland

Guevara Noubir
Northeastern University, Boston, USA

***Abstract*—Bluetooth Classic (BT) remains the de facto connectivity technology in car stereo systems, wireless headsets, laptops, and a plethora of wearables, especially for applications that require high data rates, such as audio streaming, voice calling, tethering, etc. Unlike in Bluetooth Low Energy (BLE), where address randomization is a feature available to manufactures, BT addresses are not randomized because they are largely believed to be immune to tracking attacks. We analyze the design of BT and devise a robust de-anonymization technique that hinges on the apparently benign information leaking from frame encoding, to infer a piconet's clock, hopping sequence, and ultimately the Upper Address Part (UAP) of the master device's physical address, which are never exchanged in clear. Used together with the Lower Address Part (LAP), which is present in all frames transmitted, this enables tracking of the piconet master, thereby debunking the privacy guarantees of BT. We validate this attack by developing the first Software-defined Radio (SDR) based sniffer that allows full BT spectrum analysis (79 MHz) and implements the proposed de-anonymization technique. We study the feasibility of privacy attacks with multiple testbeds, considering different numbers of devices, traffic regimes, and communication ranges. We demonstrate that it is possible to track BT devices up to 85 meters from the sniffer, and achieve more than 80% device identification accuracy within less than 1 second of sniffing and 100% detection within less than 4 seconds. Lastly, we study the identified privacy attack in the wild, capturing BT traffic at a road junction over 5 days, demonstrating that our system can re-identify hundreds of users and infer their commuting patterns.**

I. Introduction

Wireless communications have profoundly changed how people share information and access services. Unfortunately, due to the intrinsic broadcast nature of wireless channels, the immense benefits unlocked often come at the cost of exposing users to a variety of privacy-invasion attacks. Location information leakage is a particular concern, as this underpins more sophisticated threats, such as user tracking, identity discovery, and pinpointing of home/work premises. Furthermore, discovery of behaviors, preferences, and individuals' social networks are at risk, which can potentially lead to effective social engineering.

Location privacy has been investigated extensively since the early days of cellular communication systems, when Temporary Mobile Subscriber Identity (TMSI) was introduced with GSM to increase the difficulty of user tracking. This later evolved into a completely anonymized 5G registration procedure, whereby the Subscription Permanent Identifier (SUPI) is never sent in the clear, but instead is encrypted using an asymmetric Elliptic Curve Integrated Encryption Scheme (ECIES) to generate Subscription Concealed Identifiers (SUCI) [1]. In recent years, however, the privacy attack surface has expanded significantly with the pervasiveness of mobile and sensing devices, open mobile platforms (running untrusted code), diverse wireless connectivity options, and the availability of SDR platforms. For instance, faulty implementations of paging messages in LTE networks allow attackers to collect IMSIs through passive sniffing [2]. This questions the effectiveness of SUCI, given that it is possible to downgrade a terminal's connectivity from 5G to 3G via jamming, and subsequently use one of the many SDR-based IMSI catching tools [3] to reveal a target's identity. Tracking threats were also identified in Wi-Fi, where the unique Medium Access Control (MAC) address of devices, which is present in periodic probe packets, has been exploited by marketing and location analytics companies [4], or to covertly identify individuals' routes in cities [5]. Such privacy threats led to MAC address randomization features released with popular mobile operating systems [6], making tracking

harder and receiving praise from privacy advocates. Naturally, consumers are increasingly concerned about the implications of location information disclosure, as confirmed by user surveys [7] and location privacy protection legislation [8], [9].

Four billion Bluetooth-powered devices are projected to be shipped by the end of 2019, making this technology embedded in virtually every phone, car, laptop, mouse, keyboard, game console, and wearable device [10]. Bluetooth comprises two main specifications [11]: Bluetooth Classic (BT) and Bluetooth Low Energy (BLE). BT remains the dominant standard, as it is the only one supporting the Advanced Audio Distribution Profile (A2DP) required for audio streaming applications, e.g., in cars (Apple CarPlay, Android Auto, etc.) and headsets. Despite its weak cryptographic foundations for protecting the devices address, there is a common belief that BT is immune to tracking attacks demonstrated against BLE [12]. This is in part due to the perceived difficulty of capturing and analyzing 79 channels over 79 MHz of spectrum, the fact that communicating devices hop at a rate of 1,600 hops/second (transmitting on each channel for less than a millisecond), and that a BT Device Address (BDADDR) is not sent in the clear but obfuscated through whitening mechanisms that depend on the clock of the master. Address randomization was incorporated in BLE, likely due to its simpler communications design and hence increased susceptibility to tracking. Instead, BT obfuscation measures were still believed to be secure against tracking, therefore addresses continue to be fixed, as per the initial design.

In this work, we demonstrate through a combination of signal processing and iterative inference that it is possible to overcome BT obfuscation and uncover the entire, meaningful part of a device's address, thereby enabling reliable user tracking. We show that implementing our approach on inexpensive hardware is practical and we can achieve high de-anonymization accuracy in *real-time*, even at a distance from targets. While this opens new avenues for constructive use, such as profiling vehicular traffic for planning purposes, or studying Bluetooth's co-existence with other wireless technologies, the privacy implications are significant. Importantly, while countermeasures such as address randomizations in BLE and Wi-Fi, correct usage of LTE paging messages, and replacement of IMSI with SUCI, may hinder the tracking of users connected to such networks, billions of BT-powered devices are already deployed, which may be impossible to patch with an evolved privacy-preserving BT version. In fact, no plans to address the privacy problems of BT are on the horizon and it is even unclear whether such evolution would be technically feasible. In addition, while IMSI-catching attacks are *active* and can be easily detected [13], [14], BT de-anonymization is purely *passive*, which renders the identification of attackers impossible. Fortunately, affordable and fully functional technical solutions that could break BT privacy within short observation windows have yet to be developed. Our work changes that.

Existing solutions: Due to the ever-growing popularity of BT technology, a number of solutions have been developed for analysis and debugging purposes. Professional high-end products enable full-band BT traffic analysis [15], [16], yet are very expensive and built on proprietary software that cannot be modified by users, and their tracking ability is unspecified.

Few open-source alternatives, such as Ubertooth One [17], exist. Albeit cheap, these are limited to capturing traffic on a single channel at a time, therefore cannot follow multiple connections concurrently. Still, as Ubertooth is advertised as capable of infringing BT privacy, we perform a thorough performance comparison between this solution and our approach in Sec. VIII. SDR solutions that employ inexpensive radio front-ends, such as HackRF or LimeSDR, have wider bandwidth and are more flexible, yet are either limited to capturing simple BLE control traffic [18] or not fully functional [19]. At best, all these can extract the LAP of a connection's master, which is insufficient for mounting privacy breaching attacks. We review relevant research that makes use of such platforms in Sec. X.

Challenges: Multiple BT sessions happen on different channels at the same time, following hopping sequences that are unknown to an adversary and derived from the unknown master clock. Even if sniffing with multiple receivers tuned simultaneously on all the 79 channels used by BT, synchronizing the traces must be precise, otherwise ambiguities arise in the explanation of a sequence of packets exchanged. Further, figuring out a connection's hopping sequence is hardly enough for an attacker to guess the master's clock and UAP, from which this sequence and data scrambling (whitening) are derived, which offer in some sense a level of confidentiality.

Contributions: To the best of our knowledge, we present the first full-band BT sniffer in which all the relevant computations related to synchronization, demodulation, dewhitening, and decoding are combined with a multi-frame iterative inference algorithm that we propose to overcome BDADDR obfuscation and de-anonymize all the meaningful parts of addresses. Our techniques

and system are flexible, as they can be instantiated with a range of SDR platforms with different bandwidths, can intercept traffic on all 79 BT channels simultaneously, while several of the system's components are amenable to parallelization on general-purpose workstations. With these, *(i)* we demonstrate that, contrary to widespread belief, layer 2 communication in BT is completely exposed to re-identification and tracking, as device addresses can be inferred and the obfuscation can be circumvented by attackers in real-time; *(ii)* we extensively evaluate the potential of privacy attacks enabled by our system with various distances, device densities, and traffic regimes, using controlled testbeds with 26 embedded devices, a connected car and a wireless headset. We show that it is feasible to track BT devices within a 85 m range, achieving 80% identification within less than 1 s and 100% within less than 4 s; and *(iii)* we study in the wild the effectiveness of the privacy attack uncovered, targeting moving vehicles, without storing sensitive information, and showing that an adversary may be able to infer users' daily commuting patterns. Lastly, we discuss the large-scale surveillance risks the privacy-infringing attack identified enables, as well as how to mitigate the vulnerability revealed. We release open-source all the code we used in this paper on GitHub[1].

II. Bluetooth Classic Overview

BT is a wireless technology operating in the 2.4 GHz Industrial, Scientific, and Medical (ISM) band, whose specification version 5.2 has been recently released [11]. The standard amounts to over 3,000 pages and navigating this is rather involved. Thus, we begin by describing the BT frame format and the procedure adopted by devices to whiten (and hence obfuscate) frames and identity prior to transmission. For completeness, we include an overview of the BT protocol operation in Appendix A.

Frame Format and Identity Obfuscation

The structure of a BT frame is shown in Fig. 1. Similar to other wireless technologies, BT frames are preceded by a Preamble (4 bits). This is followed by a Sync Word (64 bits) and an 18-bit header. Payloads are optional in BT frames, as some of these are used for discovery/control functions. The Sync Word is obtained from the 24-bit LAP to which 6 bits of a Barker sequence [20] are appended to improve auto-correlation properties. Based on this, an expurgated (64,30) block code with bit-wise XOR of a 64-bit pseudo-random

[1]Interested readers can download and test the code from https://github/bsnet/btsniffer

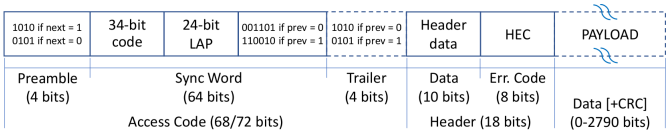


Fig. 1: BT frame format. Only the optional payload field can be eventually encrypted.

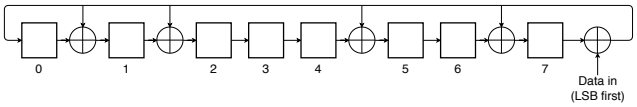


Fig. 2: LFSR used for computing the HEC based on header data; initialized with UAP and LSB set to 0; final content is the HEC, transmitted from bit 7 to bit 0.

noise (PN) sequence is derived [21]. This preserves the LAP while preceding it with 34 coded bits that guarantee a large Hamming distance between sync words of different addresses. In some cases, a fixed 4-bit Trailer encompassing a zero-one pattern follows, to be used for extended DC compensation. The preamble together with the Sync Word (and Trailer) form the Access Code. The Access Code is not subject to any further encoding and as such *the LAP will appear in clear*.

The frame header consists of two parts: the header data (10 bits) that encompasses a 3-bit Active Member Address (AMA) of a slave, a 4-bit type field, and three 1-bit flags; and the Header Error Check (HEC) (8 bits). The HEC is generated using the Linear Feedback Shift Register (LFSR) shown in Fig. 2, whose internal 8-bit state is initialized with the master's UAP. The whole header is then whitened using another LFSR (shown in Fig. 3) whose 7-bit state is initialized with bits $c_6, \ldots, c_1$ of the master's clock (clk) and by setting the bit in position 6 to 1. We summarize this procedure in Fig. 4. The whitened header is then passed through a 1/3 Forward Error Correction (FEC) block.

Note that different UAPs generate different HEC values, while different master clock values produce different whitened sequences. Reversing the UAP and clk for every frame is arguably computationally expensive, since

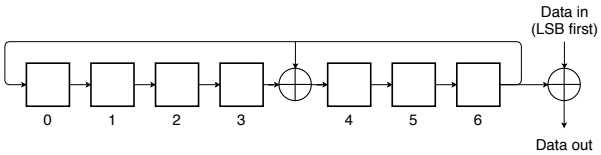


Fig. 3: LFSR used for frame (de-)whitening. Bits 0 to 5 are initialized with bits clk_{1-6} of the master's clock; bit 6 is always initialized with 1.

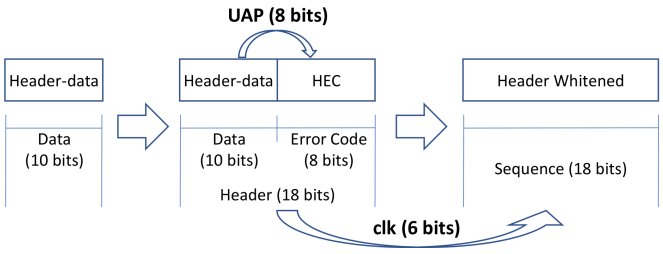


Fig. 4: BT HEC generation and header whitening.

brute-forcing these from intercepted frames would require 2^{14} iterations per frame. Thus the HEC and header whitening procedure is expected to ensure enough identity privacy. In what follows, we debunk this assumption.

III. Threat Model

Next we discuss the attacker capabilities envisioned and overview a set of plausible adversarial scenarios that would be enabled by de-anonymizing BT devices.

A. Attacker Capabilities

We assume attackers control portable computers, to which SDR front-ends that can be tuned on the 2.4 GHz band are attached (e.g. HackRF, LimeSDR, USRP, etc.). These could be battery powered or attached to fixed or mobile power supplies in covert locations (rooftops, balconies, tunnels, power buses, or cars). The attackers should be within wireless reception range of the victim devices, while this range could be potentially extended by employing directional antennas. We expect attackers to have some knowledge of signal processing, familiarity with the BT wireless communications standard, and reasonable command of computer programming.

B. Adversarial Scenarios

We distinguish three main types of attacks that are enabled through exploitation of BT device re-identification: (1) user tracking and surveillance; (2) stalking and espionage; and (3) compromising physical assets.

1) User tracking and surveillance: It is conceivable that policing agencies and state-sponsored entities would deploy BT sniffing and de-anonymization tools on public transport and in key transport hubs (airports, train stations, bridge crossings, tunnels, etc.) to *(i)* gauge footfall or traffic flow; *(ii)* identify movement patterns of groups of individuals; or *(iii)* track the precise whereabouts of a sensitive asset. BT device identity could be linked to individuals via sales databases, car plate recognition software, or CCTV and face recognition algorithms. Likewise, commercial actors would use similar infrastructure in theaters, cafés, shopping malls, etc. to monitor customers and orchestrate targeted marketing campaigns. On the other hand, city councils could hinge on knowledge of citizen flows to improve the provisioning of public services (including waste management, transportation, lighting), admittedly at a privacy cost.

2) Stalking and espionage: As SDR hardware is increasingly more affordable and open-source tools abound, a crowd-sourced stalking systems would be straightforward to design if the identity of a BT device could be reversed from overheard frames. For instance, a malicious user would post the BT identifier of an ex-partner or celebrity to a community controlled sniffing network, in order to know their whereabouts and cause emotional distress. Similarly, competing businesses or rival states could send victims allegedly free-replacements of BT-powered gadgets (earpods, smartwatches, etc.) with known identifiers, which would be subsequently tracked via crowdsourced location-sensing, to infer undisclosed locations of the victims.

3) Compromising physical assets: De-anonymization of BT devices can also underpin Man-in-the-Middle (MITM) attacks that can have severe consequences on victims, without necessarily being immediately obvious. For example, a team of attackers could coordinate to fake the presence of a victim near personal assets located remotely. This would enable unlocking smart-locks, vehicles, or access to computing infrastructure.

IV. Full-band Bluetooth Sniffing

We develop an SDR-based sniffing system that enables fast interception of BT traffic, in view of breaking the communication secrecy and re-identifying devices. We describe the system architecture, data acquisition process, and data processing pipeline implemented.

A. System Architecture

Fig. 5 gives an overview of the full-band BT sniffing system developed. This relies on an SDR front-end for raw wireless signal acquisition in the 2.4 GHz band. The front-end is connected via USB 3.0 to and driven by signal processing software running on the host computer. Our design is sufficiently flexible to allow for SDR platforms operating with different spectral widths, e.g., a single board such as the Ettus N310 capable of capturing the entire 79 MHz bandwidth used by BT, two Ettus B210 boards, each covering 40 MHz of spectrum and their output being synchronized, or any of these tuned to capture an arbitrary spectral width.

The acquired signal samples are then processed. First, they are passed to a channelizer, which can be configured

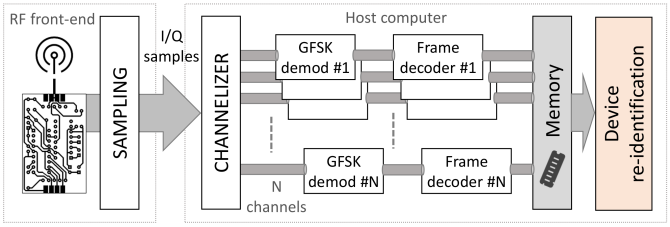


Fig. 5: Proposed full-band BT sniffing system. Raw signal is captured using SDR front-end while channelization, demodulation and frame decoding are performed on host computer. BT master re-identification is achieved by reversing HEC computation and whitening.

to output the Radio Frequency (RF) signals observable on a precise set of individual BT channels, depending on the computational capabilities of the host and whether these allow for real-time or off-line sample processing and analysis. The signals on each acquired channel are transferred to Gaussian Frequency-shift Keying (GFSK) demodulation blocks, which output the corresponding bit sequences. A separate module identifies for each channel the beginning of BT frames, based on the bit streams resulting from demodulation. The structure of the system makes it easy to implement in parallel the demodulation and frame decoding pipeline.

Depending on the number of channels selected for capture and the computing power, the BT frames acquired are stored in RAM or on PCI Express Solid State Drive (SSD), together with a timestamp, to aid frame sequence reconstruction. Data acquisition and processing can either work sequentially ("off-line" functionality), or concurrently, if the time required to process fixed-length traces is less than the time needed to acquire the same trace ("on-line" operation). The latter is dependent on platform computational power and achievable, e.g., with a double-buffer approach (where a buffer is filled with new data while data in the other one is processed).

A separate module that we describe in Sec. V fetches from memory the sniffed frames and exploits weaknesses in the HEC computation and header whitening to re-identify the BT master devices of target connections.

B. Data Acquisition

The first task performed by our system is sampling from the full 79 MHz band used by BT. Different SDR platforms are suitable for this operation. For our experiments we adopt two Ettus B210 boards. Each of these supports full-duplex operation with up to 56 MHz of real-time bandwidth, which is not sufficiently wide to capture all BT channels. Hence, the need for deploying two such boards. We tune the central frequency of the two boards on 2,421.5 MHz and 2,460.5 MHz respectively, and configure them with 44 MHz of receiving bandwidth each. We allow a small overlap (5 MHz) between the bandwidths of the two receivers, to facilitate output trace synchronization without the need for an expensive external clock.

Synchronization across channels is of paramount importance to our system, as de-anonymizing BT addresses requires to analyze frames captured by different SDRs with a common time reference. This is further needed for debugging purposes, to be able to explain a sequence of messages transmitted over different channels at different time instants. Hence, we devise a μs granularity synchronization method that doesn't require coherent capture. In essence, we use an external BT dongle to transmit periodically (every 1 s), on a channel that is captured by both interfaces, a reference frame with a known address and whitening parameters, in which we embed a Sequence Number. Should such a dongle not be available, we allow for transmitting the sync frame with one of the SDRs used for capture, given their common full-duplex capabilities. The sync frame shall be received by both SDR boards and therefore be present in both traces. Reception should be simultaneous, hence the timing offset between captures is easily compensated.

SDRs usually employ a complex sampling technique; this means that with a 44 MHz sampling frequency we can effectively acquire 44 MHz of spectrum. Two values are recorded with each sampling interval, each corresponding to a component of a complex sample and generally referred to as in-phase (I) and quadrature (Q) components. The complex samples are also called I/Q samples. We quantize with 1 byte the I/Q sample components, which results in an 88 Msamples/s sampling rate. In turn, this translates into a 176 MB/s data rate that the host system must handle. This can be managed if writing to `ramdisk` or to an m.2 SSD (which supports at least twice the required rate).

C. Data Processing

Channelizer: To be able to separate the spectral components of the wideband signal acquired previously and distinguish the frames transmitted on the different 79 BT channels, the first component of the data processing chain we implement is a channelizer. This comprises a digital down-converter scheme, by which the complex input signal is first shifted to baseband, then passed through a Finite Impulse Response (FIR) filter [22] with 1 MHz bandwidth. Since high-frequency components are

removed by the filter, the output signal is decimated to reduce the data rate. By tuning the local oscillator onto the central frequency of each target BT channel, we extract the corresponding narrowband I/Q symbols. Channelizer speed can be greatly increased by using polyphase filters to directly separate all the narrowband BT channels from the wideband signal [23].

All channels are subsequently processed by 79 demodulation and frame decoding blocks, fed with the corresponding baseband signals.

Demodulation: The I/Q samples for a single channel are fed into a GFSK demodulation block that outputs the corresponding binary data. GFSK is a digital frequency modulation technique whereby symbols are first filtered by a Gaussian filter and then used to modulate the carrier signal. BT employs a binary modulation scheme with bandwidth-bit period product of 0.5.

A well-known technique for demodulating FSK signals is based on measuring the phase difference $\Delta\phi$ between two consecutive samples of the corresponding baseband signal. Assuming that the high frequency components are filtered out by the channelizer, $\Delta\phi$ will have the same sign as the frequency deviation of the signal from the carrier and will satisfy the relation $-\pi < \Delta\phi < \pi$. Given two successive I/Q samples (I_1, Q_1) and (I_2, Q_2), the phase difference between them can be measured using some simple trigonometric computation. However, since what is relevant to our task is only the sign of $\Delta\phi$, we can avoid trigonometry by verifying that for $-\pi < \Delta\phi < \pi$ the following holds:

$$\text{sign}\left(\Delta\phi\right) = \text{sign}\left[\sin(\Delta\phi)\right] = \text{sign}\left(I_1 Q_2 - I_2 Q_1\right).$$

Frame decoding: Once the demodulation step is completed, we can detect and decode BT frames on each channel. Recall that every bit stream output by demodulators contains two samples per bit period; this oversampling proves necessary to counteract phase noise effects at the receiver. Instead of recovering one bit value from the samples within the same bit period, the decoder will treat each binary sample as a bit. All the processing performed here is intended to be repeated for each sample. The following and preceding bits are evaluated, advancing in a two by two fashion in the stream of binary samples.

By examining possible preambles with candidate sync words that follow, we can detect with high confidence the boundaries of BT frames, which we subsequently examine for re-identifying masters of target connections.

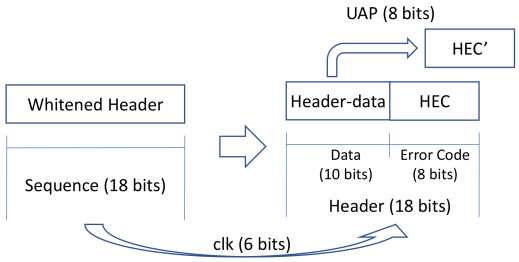


Fig. 6: Processing logic for inferring the UAP of a BT master from the whitened header.

V. Re-identifying Bluetooth Devices

BT has been long considered to provide good user privacy, because (i) devices stop responding to Inquiry frames after establishing a connection, (ii) they change channels every 625 μs following a "secret" pattern that is only known to communicating peers (hopping), and (iii) their identity remains ambiguous, as the frames exchanged only contain half of the BDADDR (the LAP) of the master and the contents are obfuscated using a per-frame "pseudo-key" that depends on the master's UAP and part of its clock (whitening). Our full-band sniffing system presented in the previous section breaks the first two identity protection features, as it enables adversaries, which supposedly neither know the channel nor the pseudo-key, to capture frames in a target session. In what follows, we show that it is possible to re-identify devices by exploiting weaknesses in the design of the header error check and header whitening mechanisms. These enable us to derive the master's UAP.[2]

To find the UAP of a device, we need to (i) first identify which 6-bits of the master clock were used to whiten a frame header and de-whiten it, and (ii) infer what UAP value produces a HEC value that matches the HEC in the de-whitened header. We illustrate this logic in Fig. 6. Note that the HEC is produced using a polynomial that is initialized with the UAP. For any given 6 bits of the master clock (clk), only one UAP will recover a valid HEC. With this in mind, we first employ Algorithm 1 to identify the (UAP, clk) pairs that could be valid. In general, for each sniffed WhitenedHeader wh, we look for (u_i, clk_i) pairs and the corresponding DeWhitened HeaderData hd_i, with $0 \leq i < 64$ such that the following holds:

$$wh = [hd_i \mid \text{HEC}(hd_i, u_i)] \oplus \text{w}(clk_i),$$

[2]Recall that the 2-byte Non-significant Address Part (NAP) is never used, but merely present for compliance with EUI-48 standards. Knowing the LAP and UAP, the NAP can be inferred using L2CAP echo requests.

where | is the concatenation operator, $\text{HEC}(hd_i, u_i)$ is the bilinear map that generates the 8-bit HEC sequence and $\text{w}(clk_i)$ is the map that generates the whitening sequence. In the following we will express the latter as $\text{w}(clk_i) = \text{w'}(clk_i \oplus 2^6)$, where w' is the linear map implemented by a LFSR that is identical to the one that generates the whitening sequence but without the static initialization of the internal state's Most Significant Bit (MSB), and 2^6 makes such initialization explicit. We also introduce notation for this mapping's upper and lower parts, i.e., $\text{w}' = \text{w}'_{hd}|\text{w}'_{hec}$, which whiten respectively the HeaderData and the HEC that are concatenated in the above equation. Let (u, clk) be the actual UAP and clock value, and hd the actual HeaderData. We can use the following equation to compute the other valid (albeit incorrect) UAP values from the (also incorrect) corresponding clocks and the associated (incorrect) HeaderData values:

$$\begin{aligned} \text{HEC}(hd_i, u_i) &= \text{HEC}(hd, u) \oplus \text{w}'_{hec}(clk_i \oplus clk) \\ hd_i &= hd \oplus \text{w}'_{hd}(clk_i \oplus clk). \end{aligned}$$

After introducing $\overline{clk}_i = clk_i \oplus clk$ and reworking the equations, we obtain

$$\text{HEC}(0, u_i) = \text{HEC}(\text{w}'_{hd}(\overline{clk}_i), u) \oplus \text{w}'_{hec}(\overline{clk}_i).$$

The above shows that values u_i of the UAP that our search algorithm computes can be obtained by taking all possible values of the clock $0 \leq \overline{clk}_i < 64$, they depend only on the correct value of the UAP, and they do not change over consecutive (and likely different) transmitted HeaderData. We also note that the UAP values for clocks that are each the 1's complement of the other are the same. After observing that the 1's complement of clock $\overline{clk}_i$ can be written as $\overline{clk}_i \oplus (2^6 - 1)$ (remember that these are 6-bit values), the above follows from the following equality:

$$\text{HEC}(\text{w}'_{hd}(2^6 - 1), 0) = \text{w}'_{hec}(2^6 - 1).$$

Finally, let u be the correct UAP and $clk(n)$ the sequence of actual clock values. The search algorithm cannot distinguish them from incorrect candidates u' and $clk'(n)$ that verify the following equalities:

$$\begin{aligned} clk'(n) &= clk(n) + 32, \\ \text{HEC}(0, u') &= \text{HEC}(\text{w}'_{hd}(32), u) \oplus \text{w}'_{hec}(32). \end{aligned}$$

The dewhitened HeaderData corresponding to the candidate can be easily determined from the "correct" one, as $hd' = hd \oplus \text{w}'_{hd}(32) = hd \oplus \texttt{0x}\texttt{C0}$. This means that after dewhitening, the incorrect candidate would have a different MSB in the packet type and different flow control bit. This could be later used for discriminating such incorrect candidate from the real UAP.

Algorithm 1 Identifying plausible (UAP,clk) pairs.

```
set good_list = []
for clk = 0:63 do
    Header = DeWhiten (WhitenedHeader, clk)
    for UAP = 0:255 do
        HEC' = ComputeHEC(HeaderData, UAP)
        if HEC' == HEC then
            push (UAP, clk) into good_list
        end if
    end for
end for
```

One key observation is then that in this list of 64 pairs there are only 32 different UAPs. From this list, we can remove wrong candidates by executing Algorithm 2 on successive frames with the same LAP, until the list is reduced to u, u'. This algorithm verifies the consistency between the timestamps of the frames and the clock values that are associated to a candidate UAP. It is also worth noting that the execution of Algorithm 2 on the first two frames received allows us to discard at least the 1's complement of the clock for any candidate UAP, unless the time difference between the received packets is exactly $64 \cdot 625$ μs. Thus, with only two frames we can effectively reduce our search space from 256 to 32 unique pairs (UAP,clk) or less. When only two possible clock values (with the 32 tick delay as demonstrated above) remain (and hence two possible UAPs), further ambiguity can be resolved only by examining the Header data de-whitened with the two possible clocks and keeping the UAP for which the inferred Header makes sense.

VI. Testbeds

We implement the designed full-band BT sniffing system using two Ettus USRP B210 SDR boards, connected to the same antenna using a splitter and to the host via separate USB 3.0 controllers. The host runs a GNU/Linux operating system and is equipped with a quad-core Intel Xeon W-2123 CPU, 32 GB of memory, and a Samsung NVMe SSD with 480 GB of storage.

To evaluate the potential of the devised system to intercept BT traffic, its ability to compromise user privacy through re-identification and sustain tracking, we employ the following three set-ups: a controlled indoor multi-device testbed, a controlled single-connection set-

Algorithm 2 Removing implausible UAPs from candidate list.

set t_1, t_2 the times in μs when consecutive frames with same LAP were received
ΔT = round($(t_2 - t_1)/625$)
for (UAP, clk) $\in$ good_list **do**
    clk' = (clk + ΔT) $\mod 64$
    Header = DeWhiten(WhitenedHeader, clk')
    HEC' = ComputeHEC(HeaderData, UAP)
    if HEC' == HeaderHEC **then**
        update (UAP, clk) = (UAP, clk')
    else
        remove (UAP, clk) from good_list
    end if
end for

up, and an "in the wild" environment. We detail the particularities of each of these testbeds next.

A. Controlled Multi-device Testbed

We conduct the first set of experiments using 26 Raspberry Pi 3 (RP) embedded boards, which have an integrated Broadcom Bluetooth chipset. We further attach a YBLNTEK Bluetooth USB dongle with CSR chipset to 24 of them. We establish two BT sessions between each pair of devices with two BT interfaces (internal plus dongle), and a single session between the remaining two RPs. This allows us to establish 25 simultaneous BT connections, which we seek to monitor. We develop scripts to enable dynamic control of the connections and traffic exchanged between peers. With this testbed, we are able to evaluate the performance of our system against known ground truth, thereby establishing a system performance baseline.

B. Controlled Single-connection Testbed

The second testbed serves to investigate the success of sniffing a single connection while varying the distance between the 'attacker' and 'target'. For this, we consider two representative use cases, namely (1) a Ford S-Max car (equipped with CarPlay streaming functionality) that connects to a mobile phone (Apple iPhone SE) and (2) a phone (Huawei P20) streaming to a headset (Sony WH1000-XM3). Also in these scenarios, a full-band trace is first recorded using two B210 SDRs and processed on the same workstation as before.

C. In-the-wild Environment

Lastly, we measure the performance of our system in the wild, using two distinct set-ups: (1) a smaller version of our system with a single SDR and capable of processing up to 16 channels in *real-time*, which is deployed on the 4$^{\text{th}}$ floor of a building (14.5 m elevation) and connected to a Yagi-Uda antenna with 13 dB gain, pointed at a traffic junction for a total of 5 days, and (2) a vehicular testbed, whereby the sniffing system is deployed within a car that travels on the highway for $\sim$2.5 hours and processes 4 channels over 8 MHz of spectrum. With both set-ups we aim to estimate how many distinct cars equipped with BT can be observed and infer user commuting patterns.

Privacy preservation in data collection: In our in-the-wild experiments, we do not persistently store any information that could identify individual users (e.g., the actual discovered BT address). Instead, we compute on the fly a hash of this information, which is kept in RAM for as little as necessary and only to generate the statistics presented here. We also dispose of RF recordings to prevent future privacy breaches that could arise, e.g., via physical interface fingerprinting.

VII. Evaluation

In this section we provide a comprehensive evaluation of the designed BT sniffing and re-identification system. We begin by assessing in our controlled environment the time required to re-identify devices, in the presence/absence of traffic and with varying number of devices. We compare the performance of our system in terms of detection time and accuracy against that of Ubertooth [17], which is the only existing open-source commodity platform for BT sniffing. We then study the impact of the distance to target on device detection time, via experiments with CarPlay and wireless headset systems. We use this set-up to also investigate how choosing different (sub)sets of channels for sniffing affects the re-identification performance. Finally, we investigate the potential of attacks on user privacy in the wild, assessing how many connected cars we can identify and inferring user commuting patterns. The following section gives a detailed comparison with Ubertooth, the only open platform with functionality similar to that of our system, while in Appendix B we take the sniffer on the highway and offer further perspectives on the seriousness of the privacy threats our platform enables. Our systematic study will demonstrate that concerns for BT user privacy are well justified.

A. Re-identification Time

In the controlled multi-device environment where we establish 25 BT sessions (see Sec. VI-A), each session

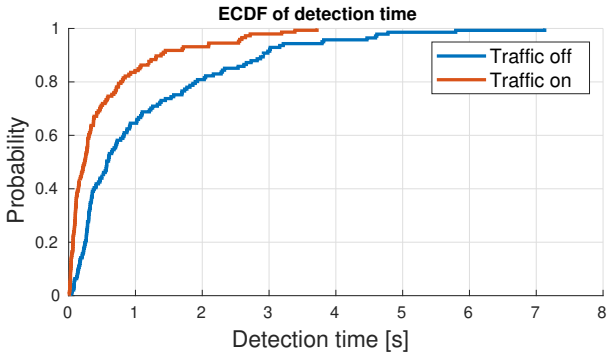


Fig. 7: ECDF of the detection time (i.e., active sniffing until UAP detection), with and without traffic. Experiments with the multi-device setup (25 connections).

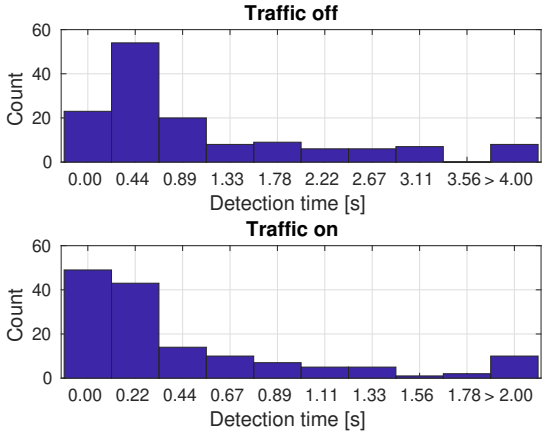


Fig. 8: Histogram of the time required to determine the master's UAP in a BT session, with and without traffic.

can be configured to generate IP packets using `iperf`. We consider two scenarios: (1) with no IP traffic, so that only BT keep-alive frames are transmitted, and (2) with IP traffic between peers (similar to streaming applications). We position the sniffer's antenna at approximately 2 meters from the targets (all RP devices are placed next to each other on a 1 meter wide board). We perform full-band sniffing for 30 seconds and process each captured trace starting from three different points in time, separated by 2 seconds. We then measure the time required to detect all sessions. The obtained results are shown in Fig. 7, where we report the Empirical Cumulative Distribution Function (ECDF) of the time required to retrieve LAPs and identifying the UAPs for each connection. Once found, we discard a UAP and restart re-identification from scratch. In this way, we can obtain statistical significance of detection times by running our technique on a single trace.

Observe that when IP traffic is present, our system can detect and de-anonymize 80% of the BT sessions within less than 1 s of sniffing. In the absence of data traffic, we still require only 2 s of traffic to detect 80% of sessions. All connections are identified in just over 3 seconds when connections exchange traffic and in less than 7 seconds if only keep-alive frames are present.

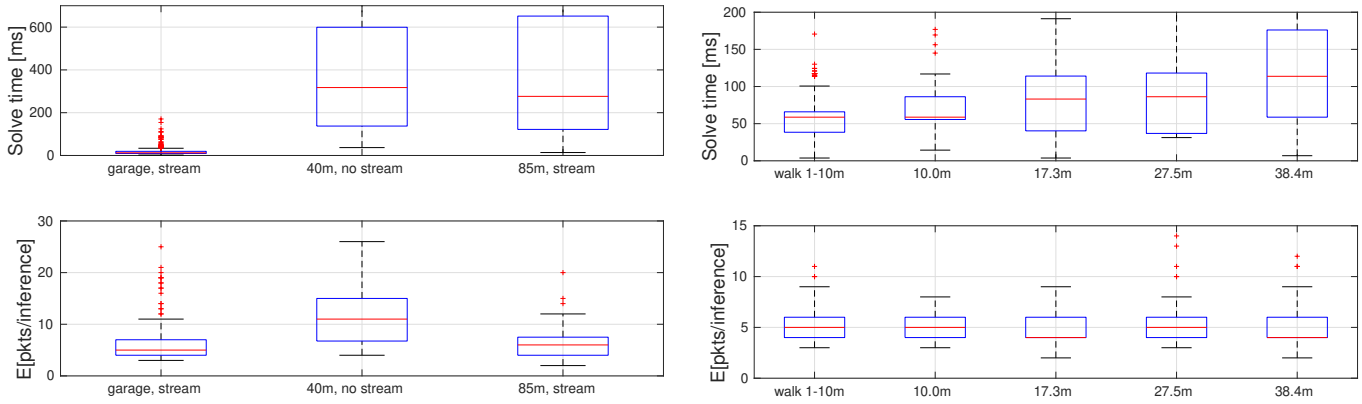
Turning attention to the time required to compute the UAP of the master for each connection, in Fig. 8 we plot the histograms of the trace time required when IP traffic is absent/present. We remark that the average time between the first observation of a BT session and the successful resolution of the associated UAP is only 516 ms when IP traffic exists on the connection, while 1.119 s are required when the connection is up but not actively used to transmit data.

B. *Impact of Distance to Target*

Next, we focus on assessing the impact of the distance to a target connection on re-identification performance. For this purpose, we work with the controlled single device testbed (see Sec. VI-B), considering both outdoor (CarPlay) and indoor (headset) scenarios.

We report in Fig. 9a the performance of our system when we seek to intercept and de-anonymize a connection between the vehicle and the phone, as we increase the distance from the sniffer, while measuring the average number of UAPs solved per second (top sub-plot) and the time require to compute the target UAP (bottom sub-plot). Frames are assumed to be correctly received if the Access Code is valid and the 1/3 FEC decoding of the Header does not need error correction (i.e., all bits within 3-bit groups have the same value). We compute statistics when the target connection is 40 m away from the sniffer and does not perform any audio streaming and respectively when at a 85 m distance, with streaming on. To put things into perspective, we also consider a "garage" scenario where the car is very close to the sniffer and the phone streams music to it.

The first thing to observe is the notable difference between the "garage" scenario and the "outdoor" ones in terms of time required to successfully solve a target UAP (bottom). This can be attributed to the fact that RF signals attenuate with the distance and the medium becomes more prone to noise and external interference. The median time to solve UAPs outdoors is in the 300 ms range. We also note that the number of frames processed before a successful UAP resolution, when the BT connection is idle and respectively used to stream music, is comparable. On average, fewer than 7 frames need to be sniffed when the CarPlay connection is



(a) CarPlay experiment (b) Headset experiment

Fig. 9: Success of privacy attacks as a distance to target increases in the two single-connection setups. Boxplots of the time required to solve the UAPs (top) and the number of packets needed (bottom).

actively used, while approximately 11 frames are needed if the connection is up but not used to stream music.

In Fig. 9b, we show the results of similar experiments conducted in the headset scenario, i.e., where a connection between a mobile phone that streams music to a wireless headset is targeted at different distances. In the "walk" experiments, the target is moving within a 10 m range from the sniffer, while the rest of the measurements correspond to cases where the user is in a fixed location at the indicated distances. It is clear that the performance of our system depends on the distance to the target connection, i.e., we can solve the target UAP much faster if the BT devices are closer. However, in all cases less than 6 frames are needed to re-identify the master UAP. This is consistent with the CarPlay experiments in which the phone was streaming music.

C. Impact of Number of Channels Sniffed

Undoubtedly, the number of channels employed for sniffing impacts on the accuracy of the sniffing and re-identification system, but also on applicability. Sniffing fewer channels at a time would make real-time surveillance possible, but can also miss some potential targets. To understand how monitoring different parts of the spectrum used by BT affects the success of attacks, we conduct new experiments in the CarPlay scenario where the phone streams music, whilst we sniff frames on all channels, half of them, then 20, 10, 2, and respectively 1 channel(s). Results are summarized in Fig. 10.

It comes at no surprise that we are able to infer the UAP of the target connection within milliseconds, if all channels are observed. The performance degrades only marginally if we listen on either the lower or the upper part of the spectrum. The target UAP can still

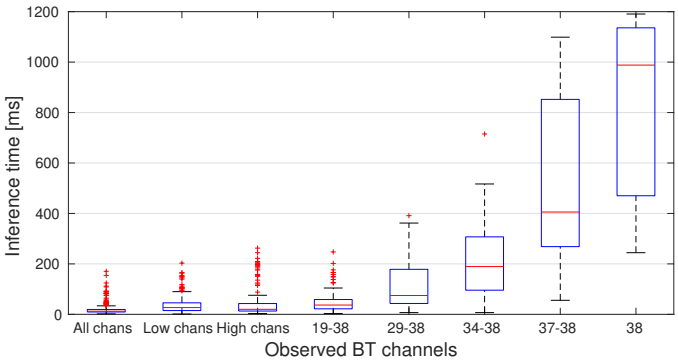


Fig. 10: Time required to solve the target UAP in the CarPlay scenario, as number of channels sniffed varies.

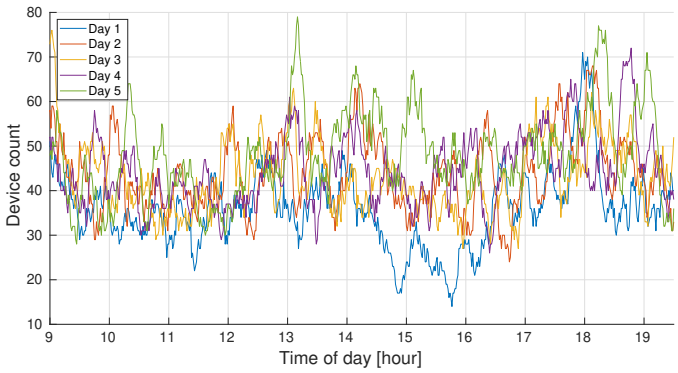


Fig. 11: Number of unique car UAPs solved by our system, averaged every 15 mins, over 5 days of activity.

be determined within tens of milliseconds if 20 or 10 channels are observed. Performance degrades rapidly though, as with 2 channels the median solving time is approximately 400 ms, while a single channel yields 1 s median solving times. We further investigate the impact of sniffed spectral width in the wild, in Sec. VII-D.

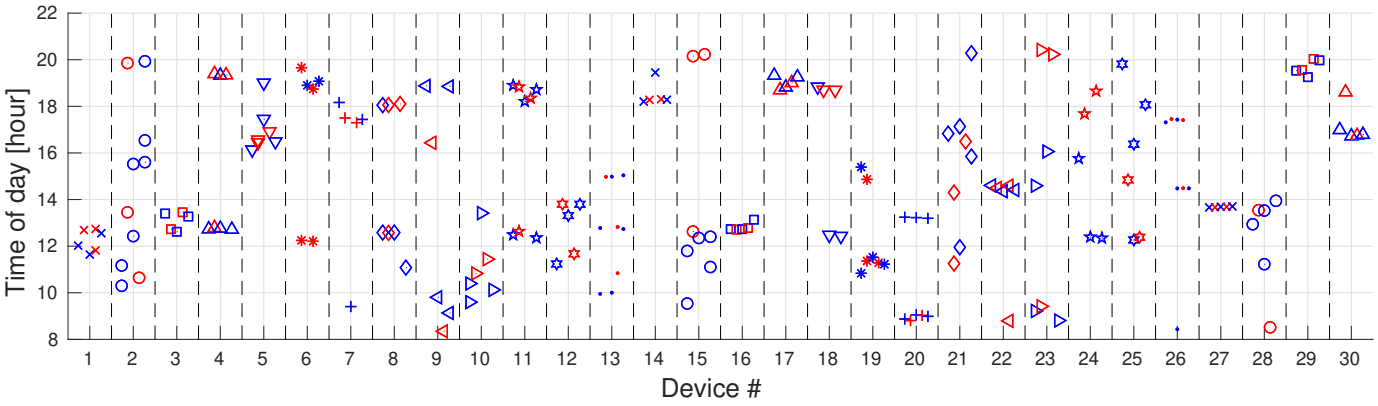


Fig. 12: Commuting patterns for 30 of the recurring users detected over 5 days of capture. Symbols are shown in alternate blue/red color to better discriminate different days of capture.

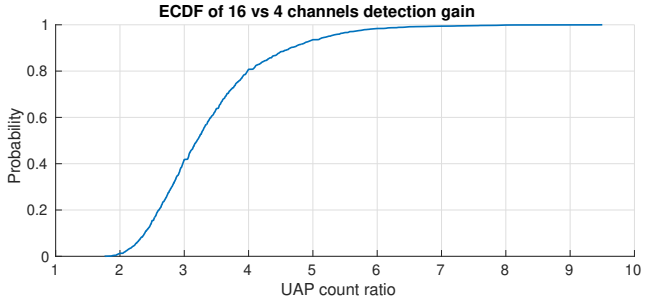


Fig. 13: Performance comparison between the 16-channel and 4-channel sniffer in terms of number of unique UAPs resolved in the wild. 60% of the time, at least 3× more UAPs solved with 16-channel sniffer.

D. Surveillance Attacks

Up to this point, all experiments have been conducted in controlled environments. In what follows we present results in the wild (see Sec. VI-C), whereby we use our system to demonstrate its surveillance capabilities.

With our sniffing and re-identification system pointing at a one-way road segment ahead of a traffic junction, we first count the number of vehicles that have BT technology on board and which we can be de-anonymized by an attacker during typical working hours (9 AM to 7:30 PM). We are able to detect cars up to a distance of 114.38 m, as confirmed by measurements with a car we control. We illustrate the statistics gathered in Fig.11, where we plot the number average number of cars observed every 15 mins. On average we detect ∼200 devices every hour. As expected, we note more intense traffic around 9 AM, 12:30 PM and 6 PM – typical start/end of work shift and lunch times.

We further examine the commuting patterns of the BT-powered cars discovered. In particular, we record when a de-anonymized device has been seen by our sniffing system during each of the days when we collected measurements. We report these results in Fig. 12, revealing the serious privacy issues to which BT users are exposed. Arguably, one may infer information about a user's personality, routine, and behavior from the observed commuting patterns. For instance, we note the precise commuting times of cars 1, 3, 16, 20, 26, and 27.

We further examine the implications of the number of channels observed in the wild on the number of connections that can be detected. Specifically, we investigate how many more BT connections could be detected and de-anonymized when sniffing on 16 channels versus 4. The obtained results are shown in Fig. 13. Observe that on average we are able to detect and re-identify 3.31 times more connections when using 16 channels during the same amount of time, which is largely consistent with the controlled experiments reported in Fig. 10.

We also experiment with our system when this is placed inside a car and assess its sniffing performance while driving on the highway. We include the obtained results in Appendix B

VIII. Comparison Against Existing Solutions

Ubertooth One is an "open source wireless development platform suitable for Bluetooth experimentation". It connects to hosts via USB and handles the MAC and PHY layers through custom firmware that controls a CC2400 transceiver. Its main advantage is the low price tag, which comes with the drawback of only being able to capture a single channel at a time. To discover ongoing sessions, the platform either stay on a set channel (which can be useful when trying to detect multiple active sessions) or hops "randomly" (to increase the chances of meeting a session). Being hardware based, the platform cannot be updated and the CC2400 radio

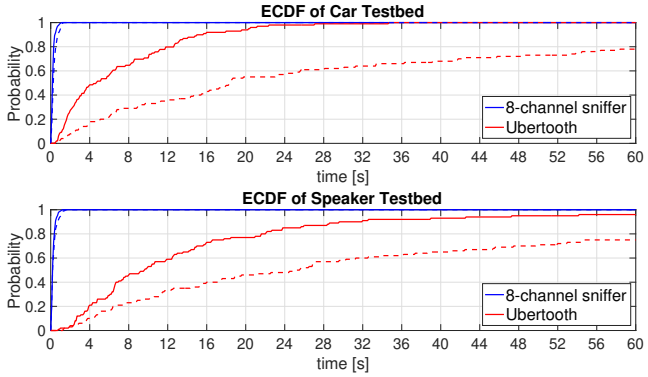


Fig. 14: ECDF of the time needed to resolve UAPs in two different testbeds. Continuous lines: data traffic on. Dashed lines: no traffic.

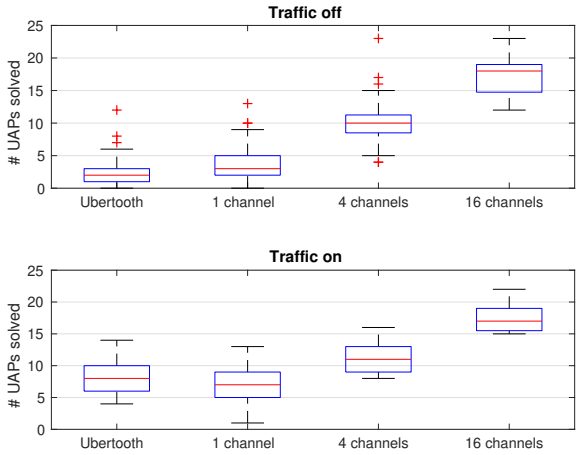


Fig. 15: Comparison in terms of UAPs resolved in 10 s of sniffing between Ubertooth and our system limited to different numbers of channels.

can only work with BT frames encoded at the Basic Rate (i.e., 1 Mb/s).

Next, we give a detailed performance comparison between Ubertooth and our system, considering different scenarios; namely, we examine the time required by each platform to discover the UAP of an Android Auto session and of a connection between a BT loudspeaker and a smartphone, with and without active traffic on the BT link. Fig. 14 shows the ECDF of these measurements. In both cases, while our approach takes less than a second to discover the UAP in over 95% of the experiments, irrespective of whether traffic is present or not, Ubertooth struggles to solve the UAPs. When no data is exchanged, it requires more than a minute in 20% of the experiments conducted. This makes it incompatible with high mobility scenarios where observation times can be very short.

We also assess how many devices could be discovered within 10 s in the multi-device testbed described in Sec. VI-A, and report the results obtained in Fig. 15. Note that the performance of our system in single-channel mode of operation is comparable to that of Ubertooth (fixed on the same channel); while Ubertooth detects on average one more UAP than our system when traffic is exchanged, our system again performs better in the absence of traffic. To understand the reason behind Ubertooth's behaviour discrepancy, we examine its sniffing code. We find that, differently to what is reported in the documentation, Ubertooth does not use timestamps of consecutively captured frames in recovering target UAPs, as we do in our system. Instead, it removes implausible UAPs iteratively, based on some sanity checks on the frame's payload. While this is fairly effective when payloads are present in the frames, the approach does not work when only NULL or POLL frames are exchanged for keeping sessions alive. In these situations, Ubertooth has to wait a considerable time before collecting a useful data frame. For instance, in the previous Car Testbed experiment, such frames are those carrying instructions for updating the car's display. To make things worse, only a fraction of these frames can be captured, since the hopping sequences followed by Ubertooth is different than that followed by the devices. In addition, it is worth noting that while our solution only identifies valid LAPs, Ubertooth might exhibit false positives in these scenarios.

Finally, we note that, even though we relied on B210 SDRs by Ettus for implementation and testing, it would be straightforward to port our system to other platforms, as long as they support IQ sampling. For instance, it would be interesting to evaluate our system using HackRF One, an SDR platform that provides up to 20 MS/s, and synchronising multiple devices through clock daisy-chaining. With a single HackRF One, and limiting the capture to just 8 or 16 channels, we anticipate it is possible to achieve very good performance at a price even lower than that of Ubertooth. We leave such experimentation for future work. We remark that the open-source nature of our system and the flexibility it offers (as compared to proprietary 'black-box' commercial platforms) lowers the entry barrier for attackers and future research into BT security alike.

IX. Discussion

The BT vulnerability we uncover can have serious privacy implications as users rely increasingly more on connected devices. Arguably, the most serious risk is that of protracted surveillance, whereby a user's locations can be tracked with concealed wireless equipment, once their identity is linked to that of a BT device they own. For

instance, road video footage used together with vehicle license plate recognition software could be employed to identify a smart car owner and the car's BT address. It would then be possible to track the person without expensive video infrastructure, by only using BT sniffers, as the system we presented. This is feasible with our approach if deploying a camera next to the developed sniffer, pointing this towards incoming traffic. One can acquire a video frame every 100 ms, process it, and discard the data. Character segmentation and optical recognition [24] can be used to detect plate numbers.

As wearables adoption grows [25], both the risk of surveillance and potential benefits of passenger flow monitoring could be exploited with our BT sniffing (and de-anonymization) system in urban settings. This would require deployment of sniffing infrastructure only at strategically selected hubs/checkpoints, such as airport terminals, train stations, road tunnels, or bridge crossings. For instance, there are only 21 main bridges and 16 tunnels that connect Manhattan to other boroughs of New York City and New Jersey. Deploying our system at such locations could offer insights into commuter flows.

Mitigation: Preserving BT device anonymity would require a new revision of the Bluetooth standard. While it is unreasonable to expect the whitening and HEC generation procedures to be modified, given the number of BT devices already on the market, full address or UAP randmomization should be feasible. Cryptographically generated addresses similar to those used in BLE or proposed for IPv6 [26] could be used, by which the 64-bit device identifier would be created with a cryptographic hash of information exchanged by peers during pairing.

X. Related Work

Connection oriented Bluetooth tracking was proposed in [27] for room-level indoor localization of users, with the goal of colleague searching and optimization of building heating/cooling. However, the system only works as long as target devices deliberately perform a one-off registration. Early room-level tracking without explicit user consent exploits the Bluetooth inquiry process (see, e.g., [28], [29]). Bluetooth devices commonly become "undiscoverable" after pairing, which questions the practicality of early tracking approaches. Spill and Bittau investigate the feasibility of eavesdropping on undiscoverable devices and develop the first open-source BT sniffer [30]. This solution is limited to a single channel, relies on cyclic redundancy checks that are not present in many BT profiles (or are encrypted), which yields high false positive and miss detection rates, questioning it practicality, as we have shown. Recent work uses passive sniffing based on Ubertooth together with active inquiry scanning to empirically verify the feasibility of this approach for forensics and surveillance purposes [31]. The value of the findings is limited, given the shortcomings of single-channel sniffing and imprecise detection of this platform. A dual-radio Ubertooth setup is used in [32] to jam and predict adaptive hoping sequences, in view of BT/BLE sniffing. De-anonymization is however not pursued, unlike in our approach which doesn't require active jamming, since we are able to eavesdrop on all channels simultaneously.

Previous work also scrutinizes BLE privacy. Beacons were used to establish a user's indoor location [33], which together with physical fingerprinting [34] can underpin user tracking. M. Ryan highlights the simplicity of snooping on BLE by exploiting the advertisement messages sent periodically on dedicated channels [35]. Although the standard introduces address randomization, Fawaz et al. show that more than 200 BLE devices studied reveal their presence to adversaries and propose an external management solution to mitigate this problem [12]. An SDR tool for BLE/Wi-Fi debugging is proposed in [36], where multi-channel capture is considered. BT is however substantially different than BLE and with the advent of SDR platforms and the growing popularity of connected car/wireless entertainment based on BT, user privacy is at risk, as we revealed.

XI. Conclusions

We practically demonstrated that BT is inadequate for ensuring user identity and location privacy. We proved that apart from a 1/2 uncertainty about a master's UAP, which can be resolved through header data inspection, the meaningful part of the master BT address is recoverable with a limited number of packets and without requiring to examine the frames' payloads, which can be encrypted. We empirically proved the benefits of capturing the entire BT spectrum with a full-band SDR system that we developed. With the decreasing costs of SDR platforms, BT sniffing will no longer be confined to single-channel sniffing. As such, user privacy is at risk and calls for revising the Bluetooth specification.

Acknowledgements

This material is based upon work partially supported by Arm Ltd, the National Science Foundation under Grant NSF/DGE-1661532, and the European Commission (EC) in the framework of the H2020-ICT-2016-2017 project ORCA (Grant agreement no. 732174).

REFERENCES

[1] E. C. Jimenez, P. K. Nakarmi, M. Naslund, and K. Norrman. Subscription identifier privacy in 5g systems. In *2017 International Conference on Selected Topics in Mobile and Wireless Networking, MoWNeT 2017*, 2017.

[2] C. Sørseth, S. X. Zhou, S. F. Mjølsnes, and R. F. Olimid. Experimental analysis of subscribers' privacy exposure by lte paging. *Wireless Personal Communications*, 109(1):675–693, 2019.

[3] S. F. Mjølsnes and R. F. Olimid. Experimental assessment of private information disclosure in lte mobile networks. In *ICETE 2017 - Proceedings of the 14th International Joint Conference on e-Business and Telecommunications*, volume 4, pages 507–512, 2017.

[4] Bloomberg. Euclid Analytics Inc - Company profile and news, Accessed: Sept. 2019.

[5] Wired. Tracking devices hidden in London's recycling bins are stalking your smartphone, August 2013.

[6] Arstechnica. iOS 8 to stymie trackers and marketers with MAC address randomization, June 2014.

[7] Kassem Fawaz and Kang G Shin. Location privacy protection for smartphone users. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 239–250. ACM, 2014.

[8] United States Senate Judiciary Committee. Location Privacy Protection Act of 2014, June 2014.

[9] Official Journal of the European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), May 2016.

[10] Bluetooth SIG, Inc. Bluetooth Market update, 2019.

[11] Bluetooth SIG, Inc. Bluetooth Core Specification v5.1, Jan. 2019.

[12] Kassem Fawaz, Kyu-Han Kim, and Kang G. Shin. Protecting privacy of BLE device users. In *USENIX Security*, August 2016.

[13] A. Dabrowski, N. Pianta, T. Klepp, M. Mulazzani, and E. Weippl. Imsi-catch me if you can: Imsi-catcher-catchers. In *ACM International Conference Proceeding Series*, volume 2014-December, pages 246–255, 2014.

[14] S. Park, A. Shaik, R. Borgaonkar, and J. . Seifert. Anatomy of commercial imsi catchers and detectors. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 74–86, 2019.

[15] Ellisys. Bluetooth analyzers comparison chart. https://www.ellisys.com/products/btcompare.php, Accessed: Sept. 2019.

[16] Frontline. Sodera Wide Band Bluetooth Protocol Analyzer. http://www.fte.com/products/sodera.aspx, Accessed: Sept. 2019.

[17] Project ubertooth. http://ubertooth.sourceforge.net/, Accessed: June 2019.

[18] Photosware. The Photos Project. https://github.com/pothosware/PothosCore, Accessed: June 2019.

[19] gr-bluetooth. Bluetooth for gnu radio. http://gr-bluetooth.sourceforge.net/, Accessed: June 2019.

[20] R. H. Barker. Group synchronizing of binary digital sequences. *Communication Theory*, pages 273–287, 1953.

[21] L.H. Charles Lee. *Error-Control Block Codes for Communications Engineers*. Artech House, 2000.

[22] Alan V. Oppenheim, Alan S. Willsky, and Ian T. Young. *Signals and Systems (2nd Edition)*. Pearson, 1996.

[23] F. J. Harris, C. Dick, and M. Rice. Digital receivers and transmitters using polyphase filter banks for wireless communications. *IEEE Transactions on Microwave Theory and Techniques*, 51(4):1395–1412, April 2003.

[24] Tesseract Open Source OCR Engine. https://github.com/tesseract-ocr/tesseract, Accessed: Oct. 2019.

[25] Mordor Intelligence. *Smart Wearable Market - Growth, Trends, and Forecast (2019 - 2024)*. 2019.

[26] Tuomas Aura. Cryptographically generated addresses (cga). In *International Conference on Information Security*, pages 29–43. Springer, 2003.

[27] Simon Hay and Robert Harle. Bluetooth tracking without discoverability. In *Proc. International Symposium on Location and Context Awareness (LoCA)*, Tokyo, Japan, May 2009.

[28] Mortaza S. Bargh and Robert de Groote. Indoor localization based on response rate of bluetooth inquiries. In *Proc. ACM International Workshop on Mobile Entity Localization and Tracking in GPS-less Environments*, MELT '08, pages 49–54, San Francisco, California, USA, 2008.

[29] V. Kostakos. Using bluetooth to capture passenger trips on public transport buses. *Personal and Ubiquitous Computing*, pages 1–13, 2008.

[30] Dominic Spill and Andrea Bittau. Bluesniff: Eve meets alice and bluetooth. In *USENIX WOOT*, 2007.

[31] M. Chernyshev, C. Valli, and M. Johnstone. Revisiting Urban War Nibbling: Mobile Passive Discovery of Classic Bluetooth Devices Using Ubertooth One. *IEEE Transactions on Information Forensics and Security*, 12(7):1625–1636, July 2017.

[32] Wahhab Albazrqaoe, Jun Huang, and Guoliang Xing. Practical bluetooth traffic sniffing: Systems and privacy implications. In *Proc. ACM MobiSys*, 2016.

[33] S. Kajioka, T. Mori, T. Uchiya, I. Takumi, and H. Matsuo. Experiment of indoor position presumption based on rssi of bluetooth le beacon. In *Proc. IEEE Global Conference on Consumer Electronics (GCCE)*, Oct 2014.

[34] Tien Dang Vo-Huu, Triet Dang Vo-Huu, and Guevara Noubir. Fingerprinting Wi-Fi Devices Using Software Defined Radios. In *Proc. ACM WiSec*, 2016.

[35] Mike Ryan. Bluetooth: With low energy comes low security. In *USENIX Workshop on Offensive Technologies*, Washington, D.C., 2013.

[36] Francesco Gringoli, Nahla Ali, Fabrizio Guerrini, and Paul Patras. A flexible framework for debugging iot wireless applications. In *IEEE Workshop on Metrology for Industry 4.0 and IoT*, 2018.

[37] B. S. Peterson, R. O. Baldwin, and J. P. Kharoufeh. Bluetooth inquiry time characterization and selection. *IEEE Transactions on Mobile Computing*, 5(9):1173–1187, Sept 2006.

APPENDIX

A. Bluetooth Protocol Operation

We briefly describe basic concepts that we use to explain our device address de-anonymization attack.

a) Physical Layer: BT adopts a Frequency-hopping Spread Spectrum (FHSS) channel access scheme. The 2.4 GHz band is divided into 79 contiguous channels, each of 1-MHz. Data frames are modulated using binary GFSK after bits are obfuscated through a

whitening procedure. By this, data is passed through a LFSR initialized with part of the internal clock.

All BT frames start with an Access Code followed by a header. Different types of frames are defined for different services (e.g., keep-alive, audio streaming, etc.), each of which has specific FEC following the header that is protected with Hamming codes. As we show, *we exploit the header protection mechanism to reverse the obfuscation applied over the entire frame.*

b) MAC Layer: Throughout the different phases a BT device undergoes when communicating, from advertising (i.e. when a device is "discoverable" or "scannable"), to scanning, and data exchange, it hops across different channels 1,600 times per second. The exact hopping sequence is negotiated and shared by actively communicating devices. Each piece of BT equipment is identified by a 6-byte MAC address that does not change over time. This address is logically divided into three parts: a 2-byte NAP, a 1-byte UAP, and a 3-byte LAP. Devices form "piconets", where a master periodically polls slaves and all the frames exchanged contain the LAP of the master.

c) Network Formation: A master establishing a piconet initiates a discovery procedure to identify and connect to other devices within range. For this, the master broadcasts inquiries over 32 wake-up carriers, which are equally spaced in the 79 MHz range, hopping following a pseudo-random sequence that is derived from its MAC address. Such packets are identified by an Inquiry Access Code (IAC) that is known to all devices.

All devices listen periodically (every 1.28 s) for inquiries on a *single frequency* chosen from the set of 32 wake-up carriers, for a total duration of 11.25 ms. This "inquiry scanning" frequency also changes, according to the device's own hopping sequence. When receiving an inquiry frame, a device enters a back-off procedure and remains on the channel where the inquiry was received, for a random number of time slots uniformly distributed in the $[0, 1024)$ range (in order to reduce the probability of collision with others that have received the same inquiry). After back-off, the device returns to inquiry scan mode and, upon receiving a second inquiry, it replies in the next slot with a Frequency Hopping Synchronization (FHS) message, which contains the address of the device and its clock offset [37]. Note that *non-discoverable devices may no longer respond to inquiries after establishing communication with a master.*

If an inquiry is successful, the master enters a paging mode and hops on a sequence derived from the slave's address, sending a page message to the device it wants to connect to. This contains a so-called Device Access Code (DAC) derived from the lower 24 bits of the paged device's address. An acknowledgement is sent back, which contains the slave ID. The master then sends a FHS frame, which the slave will use to subsequently follow the master's hopping sequence; this is *computed based on the master's UAP and part of its clock*. This sequence is confirmed with another page response. The master then assigns a 3-bit AMA to the slave and the connection is established.

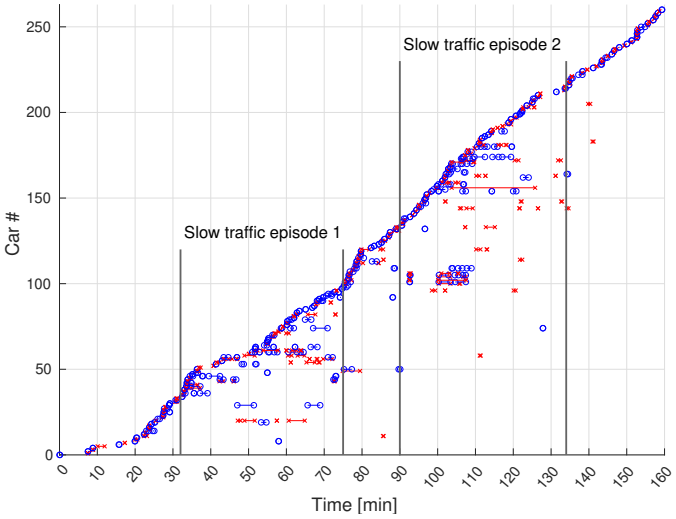


Fig. 16: Unique UAPs detected at different times during an approximately 2.5-hour drive on the highway.

B. Privacy Infringements on the Move

Another experiment we conduct is with our sniffing system deployed inside a car, with the antenna pointing opposite to the direction of movement, while we drive the car on the highway. Our aim is to assess how many unique UAPs an attacker could infer under different traffic conditions, as the sniffing system moves with speeds ranging between 5 and 120 km/h depending on road congestion levels, and verify how often we identify the same target, thereby offering perspectives on the surveillance risks to which connected cars are exposed through the de-anonymization attack we uncover.

Fig. 16 illustrates our findings. Over the entire travel duration, our system is able to detect over 250 distinct BT-powered cars. More importantly, during periods when traffic is slow, we can re-identify several targets multiple times. This highlights the potential of employing the privacy attack we uncover to follow targets while remaining visually undetectable, since many of these vehicle are hundreds of meters away from the sniffer, mostly not within line of sight.