A Practical Bluetooth Traffic Sniffing System: Design, Implementation, and Countermeasure

Wahhab Albazrqaoe, *Student Member, IEEE*, Jun Huang, *Member, IEEE*, and Guoliang Xing, *Member, IEEE*

***Abstract*— With the prevalence of personal Bluetooth devices, potential breach of user privacy has been an increasing concern. To date, sniffing Bluetooth traffic has been widely considered an extremely intricate task due to Bluetooth's indiscoverable mode, vendor-dependent adaptive hopping behavior, and the interference in the open 2.4 GHz band. In this paper, we present *BlueEar*–a practical Bluetooth traffic sniffer. BlueEar features a novel *dual-radio architecture* where two Bluetooth-compliant radios coordinate with each other on learning the hopping sequence of indiscoverable Bluetooth networks, predicting adaptive hopping behavior, and mitigating the impacts of RF interference. We built a prototype of BlueEar to sniff on Bluetooth classic traffic. Experiment results show that BlueEar can maintain a packet capture rate higher than 90% consistently in real-world environments, where the target Bluetooth network exhibits diverse hopping behaviors in the presence of dynamic interference from coexisting 802.11 devices. In addition, we discuss the privacy implications of the BlueEar system, and present a practical countermeasure that effectively reduces the packet capture rate of the sniffer to 20%. The proposed countermeasure can be easily implemented on Bluetooth master devices while requiring no modification to slave devices such as keyboards and headsets.**

***Index Terms*— Bluetooth security, Bluetooth traffic sniffing, personal area wireless networks.**

I. Introduction

IN RECENT years, Bluetooth has enjoyed an unprecedented penetration rate in mobile devices. About 4 billion Bluetooth devices are expected to ship worldwide in 2018 [2]. In particular, Bluetooth has become the *de facto* connectivity interface for wireless accessories and smart devices including keyboard/mouse, headsets, wearables like fitness trackers and smart watches, smart appliances, and in-car telematic systems like Android Auto [3] and CarPlay [4]. Because the communication of these devices is privacy-sensitive in nature, Bluetooth employs a two-level stream cipher to encrypt packets at the link-layer [2]. Unfortunately, recent studies have revealed many critical flaws of this encryption scheme [5]–[11]. In particular, it is shown that Bluetooth is subject to several practical attacks that can circumvent, compromise, or even crack the link-layer encryption, leading to potential user privacy breach [7].

Despite the well-documented flaws of Bluetooth encryption, to date, sniffing Bluetooth traffic has been considered an extremely intricate task due to the following reasons. *(i)* Bluetooth employs frequency hopping spread spectrum, where carrier frequency is rapidly switched following a pseudo-random hopping sequence. The hopping sequence is hidden when Bluetooth is in indiscoverable mode, making it difficult for a sniffer to hop following the target; *(ii)* when coexisting with other wireless devices on overlapping frequencies, Bluetooth performs adaptive hopping, where the hopping sequence is frequently modified to adapt spectrum use. Such an adaptive hopping behavior is vendor-dependent, and may differ significantly across different devices; and *(iii)* in the crowded 2.4 GHz band, the sniffer may experience intensive interference from coexisting wireless devices, especially the prevalent 802.11 based WLANs, resulting in poor sniffing performance.

There exist a few proprietary off-the-shelf Bluetooth sniffers [12], which are primarily designed for protocol diagnosis in controlled wireless settings. In addition, to sniff the traffic of frequency hopping Bluetooth, they rely on specialized radios to monitor all Bluetooth subchannels in parallel, which makes them costly. As an example, off-the-shelf sniffers manufactured by Frontline Test Equipment [12]–the leading provider of Bluetooth protocol analyzer–cost $10K to $25K per unit. Recently, a few low-cost Bluetooth packet sniffers have been developed based on open-source wireless platforms [13]–[20]. These systems are exclusively designed for sniffing Bluetooth traffic in basic hopping mode. In practice, they suffer prohibitively poor sniffing performance due to the misprediction of adaptive hopping behavior, as well as excessive packet corruptions caused by the interference in the open 2.4 GHz band.

As Bluetooth becomes increasingly popular worldwide, an in-depth study on its resistance to traffic sniffing becomes imperative. In this paper, we explore the feasibility and privacy implications of sniffing Bluetooth traffic in practical environments using general, inexpensive wireless platforms. Our major contribution is two-fold.

The BlueEar System: We present the design, implementation, and evaluation of *BlueEar*–the first practical Bluetooth traffic sniffer that consists only of inexpensive, Bluetooth-compliant radios. BlueEar features a novel *dual-radio architecture*, where two radios–named as *scout* and *snooper*–coordinate with each other on learning the hopping sequence of indiscoverable Bluetooth, predicting adaptive hopping behavior, and handling complex interference conditions.

Manuscript received September 7, 2017; revised April 1, 2018; accepted October 31, 2018; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor G. Zussman. Date of publication December 3, 2018; date of current version February 14, 2019. This work was supported in part by the U.S. National Science Foundation under Grant CNS1423221. This work was presented in part at the 14th ACM International Conference on Mobile Systems, Applications, and Services (MobiSys 2016), Singapore, Jun. 26-30, 2016. *(Corresponding author: Jun Huang.)*

W. Albazrqaoe is with the College of Engineering, University of Kufa, Najaf 54001, Iraq (e-mail: albazrqa@gmail.com).

J. Huang is with the School of EECS, Peking University, Beijing 100871, China (e-mail: jun.huang@pku.edu.cn).

G. Xing is with the Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: glxing@cuhk.edu.hk).

Digital Object Identifier 10.1109/TNET.2018.2880970

Specifically, the scout hops over all Bluetooth subchannels to survey interference conditions and monitors the target's packet transmissions. By fusing these measurements, BlueEar uses a probabilistic clock matching algorithm to determine the basic hopping sequence, and then integrates statistical models and a lightweight machine learning algorithm to predict the adaptive hopping behavior, which allows the snooper to hop following the target. To maintain reliable sniffing performance in complex interference conditions, the scout runs a selective jamming algorithm, which manipulates the hopping of the target to mitigate the impacts of interference. We have implemented a prototype of BlueEar for sniffing the traffic of Bluetooth Classic, which offers enhanced data rates and a more complex hopping protocol than Bluetooth Low Energy (BLE). Our prototype can be expended to sniff on BLE traffic. The prototype employs two Ubertooths [13] to implement the scout and the snooper, and interfaces them to a controller running on a Linux laptop. We identify critical issues in Ubertooth firmware that severely degrades its performance during frequency hopping, and present novel solutions to address these flaws. Extensive experiments results show that BlueEar can maintain a packet capture rate higher than 90% consistently in practical environments, where the target Bluetooth network exhibits diverse hopping behaviors in the presence of interference from coexisting 802.11 based WLANs.

Privacy Implications: We discuss the implication of the BlueEar system on BLE privacy. Moreover, we evaluate the performance of BlueEar when eavesdropping on audio traffic, which is known to be extremely sensitive to packet loss. We show that the packet capture rate achieved by BlueEar translates to a high audio quality with a mean opinion score (MOS) of 3.5, which is translated into `Fair' and `Good' from the listener's perspective. Furthermore, we present a practical countermeasure, that can be easily implemented in the user-space driver of the Bluetooth master device, while requiring no modification to existing slave devices like keyboards and headsets. The countermeasure effectively reduces the packet capture rate of the sniffer to 20%, and degrades the MOS of eavesdropped audio to 1.5, which is between 'Bad' and 'Poor'.

The rest of the paper is organized as follows. Section II reviews related work. In section III, we introduce the background on Bluetooth system in general. We present system overview and architecture in section IV. In section V, we discuss BlueEar system design in detail. Evaluation of BlueEar system performance is presented in section VII. We discuss the impacts of BlueEar on privacy breach for Bluetooth system, including BLE, in section VIII. Section IX concludes the paper.

II. Related Work

Bluetooth Classic employs E_0–a two-level stream cipher based on 128-bit link key–to encrypt packet payloads. The link key is established through *pairing*, where Bluetooth devices authenticate each other using a secret PIN. In the literature, studies have revealed many critical flaws of this encryption scheme [5]–[11].

First, recent studies on E_0 cryptanalysis have shown that the 128-bit link key of E_0 is considerably weaker than what is originally intended [5], [6], [9]. The encryption key can be cracked with 2^{27} online operations and $2^{21.1}$ offline operations instead of 2^{128}. Specifically, given the headers of $2^{22.7}$ Bluetooth packets, the attack takes a few seconds to restore the target encryption key. Such attack is implemented on a single core PC and requires 4 MB RAM of memory. Furthermore, the effective security of the link key will further degrade when a packet sniffer is employed by the attacker.

Recent studies have demonstrated the feasibility of circumventing Bluetooth encryption using expensive, proprietary packet sniffers. For instance, the traffic pattern of popular fitness trackers is found to be strongly correlated with the user's activity, making it possible to track user gait and identity. As a result, a passive traffic sniffer can uncover important private information about the user, even without decrypting packet payloads [7]. Moreover, due to computation and power constraints, many Bluetooth peripherals–including most Bluetooth mice manufactured by major vendors like Logitech [21]–do not support encryption, which may result in user privacy breach.

Despite the well-documented flaws of Bluetooth encryption, sniffing Bluetooth traffic has been widely considered an extremely intricate task. There exist several proprietary and open source systems for sniffing Bluetooth traffic. For example, the firmware of a few Bluetooth chipsets allow the radio to report packet-level diagnosis by working in sniffing mode [18]–[20]. However, the sniffing device must pair with the target, which makes it incapable of passive sniffing. The GNURadio/USRP [14], [15] platform can be programmed to decode Bluetooth packets [16]. Due to large signal processing overhead and frequency switching delay, they are limited to sniffing one subchannel at a time. There exist several proprietary Bluetooth packet sniffers designed for protocol diagnosis in controlled wireless settings [12]. They rely on specialized radio to monitor all subchannels in parallel, which makes them extremely costly. For instance, off-the-shelf sniffers manufactured by Frontline Test Equipment [12] – the leading provider of Bluetooth protocol analyzer–cost $10K to $25K per unit.

Recently, several low-cost Bluetooth packet sniffers [13] [17] have been developed based on Ubertooth [13]–an open-source Bluetooth development platform. However, existing Ubertooth-based sniffers are exclusively designed for sniffing Bluetooth traffic in basic hopping mode. In the crowded 2.4 GHz band, Bluetooth rarely hops in basic hopping mode because of the interference from coexisting wireless devices, especially the prevalent 802.11 based WLANs [22] [23] [24] [25]. As a result, existing low-cost sniffers suffer prohibitively poor sniffing performance in practice due to the misprediction of adaptive hopping behavior, as well as excessive packet corruptions caused by interference.

Compared with existing Ubertooth-based systems, BlueEar is designed for sniffing Bluetooth traffic in practical environments where both the sniffer and the target may suffer from intensive interference from coexisting wireless devices.

To achieve this goal, we address the key challenges posed by the indiscoverable mode of Bluetooth, the vendor-dependent adaptive hopping behavior, and the difficulties in maintaining consistent sniffing performance in the crowded 2.4 GHz band. In addition, we identify various critical issues in Ubertooth firmware that significantly degrade its performance during frequency hopping, and present solutions to address these flaws. We note that although our prototype is developed based on Ubertooth, the design of BlueEar is platform-independent and can be easily ported to other systems.

III. Bluetooth Background

Piconet: Bluetooth networks employ a master-slave structure called a *piconet*. The device that has the least computation and power constraints is usually selected as the master to manage communication. Other devices are slaves. Bluetooth piconets use the MAC address of the master device as the *piconet address*. All devices from the same piconet are synchronized to the *piconet clock*–a clock signal generated by the master.

The Hopping Protocol: We now introduce the hopping protocol of Bluetooth Classic, which is more complex than that of BLE. The hopping protocol of BLE is discussed in Section VIII. In Bluetooth Classic, the hopping protocol is defined by a *physical channel*, which is characterized by pseudo-random hopping over 79 subchannels from 2.4 to 2.48 GHz. The carrier frequency is switched every 625 μs, resulting in a maximum hopping rate of 1600 hops/s. Specifically, there are two types of physical channel for data communication, including,

(i) Basic channel. In each hop of the basic channel, the subchannel index is calculated by $\mathcal{H}(A, c)$, where $\mathcal{H}(\cdot)$ denotes the basic hop selection kernel specified by the Bluetooth standard [2], A is the piconet address, and c is the piconet clock. In Bluetooth Classic, the piconet clock is a 27-bit logic counter that ticks every hop. Because piconet clock wraps around every 2^{27} ticks, the basic channel repeats itself every 134,217,728 hops, which take approximately 24 hours. In other words, the basic channel can be characterized by a *basic hopping sequence* and a *hopping phase*. The basic hopping sequence, which is determined by the piconet address, is composed of 2^{27} subchannel indices $\{i_0, ..., i_{2^{27}-1}\}$, where $i_c = \mathcal{H}(A, c)$. The hopping phase, which is determined by the piconet clock, is the index of the current hop.

(ii) Adapted channel. When coexisting with other wireless devices on overlapping frequencies, Bluetooth performs adaptive hopping where the basic channel is frequently modified to adapt spectrum use. The adaptation is guided by a *subchannel map*, which classifies subchannels as good and bad based on interference conditions. When the subchannel selected in a hop is bad, a *remap* function is called to compute a pseudo-random index i based on piconet address and clock. The bad subchannel is then replaced by the i-th good subchannel. Although adaptive hopping is the *de facto* scheme used by off-the-shelf Bluetooth devices, the Bluetooth standard does not specify the implementation of subchannel classification, resulting in different adaptive hopping behaviors across devices manufactured by different vendors.

Indiscoverable Mode: When establishing the piconet, Bluetooth devices authenticate each other through a *pairing* process. To enhance privacy, Bluetooth piconets can be put in indiscoverable mode to hide key technical parameters from unpaired devices. These parameters–including piconet address, piconet clock, and subchannel map–determine hopping behavior. Although recent study has demonstrated the possibility of extracting piconet address from packet preambles [16], a Bluetooth packet sniffer cannot hop following the target unless it acquires all hidden parameters.

IV. BlueEar Overview

A. Objectives and Challenges

We study Bluetooth privacy by exploring the feasibility of sniffing Bluetooth traffic in practical environments. To this end, BlueEar is designed as a *passive* traffic sniffer that leverages general, inexpensive wireless platform to capture Bluetooth packets without pairing with the target piconet. To achieve this goal, we tackle the following challenges.

(i) Secret hopping phase. In indiscoverable mode, the piconet clock that indicates the hopping phase is hidden from BlueEar. In adaptive hopping mode, determining the hopping phase is particularly challenging because the basic hopping sequence of the target is subject to frequent modifications.

(ii) Vendor-dependent adaptive hopping. The adaptive hopping of Bluetooth is guided by a subchannel map, which classifies subchannels as good and bad based on dynamic interference conditions. However, the Bluetooth standard does not specify the implementation of subchannel classification, resulting in vendor-dependent implementation, where Bluetooth chipsets manufactured by different vendors may hop over different subchannels even in the same spectrum context.

(iii) Interference in the crowded 2.4 GHz band. When coexisting with other wireless devices, BlueEar may experience hidden and exposed interference. When an RF signal that interferes with the target is too weak to be measured at BlueEar, the spectrum contexts observed by BlueEar and the target may differ, making it difficult to predict adaptive hopping behavior. When an RF source interferes with the target but BlueEar, significant degradation of sniffing performance may occur. While authorized devices can maintain packet reception performance by coordinating their hopping, designed as a passive sniffer, BlueEar cannot coordinate with the target, which may lead to substantial packet corruptions.

B. System Architecture

Instead of using specialized radio to monitor all subchannels in parallel, BlueEar tackles the above challenges based on a simple *dual-radio architecture* that consists of two inexpensive, Bluetooth-compliant radios. The two radios–named as *scout* and *snooper*–are interfaced to a controller, which employs a suite of novel algorithms to coordinate the two radios, gluing them as a powerful passive traffic sniffer. Fig. 1 illustrates the architecture of BlueEar. Specifically, the working flow of BlueEar can be divided into the following steps.

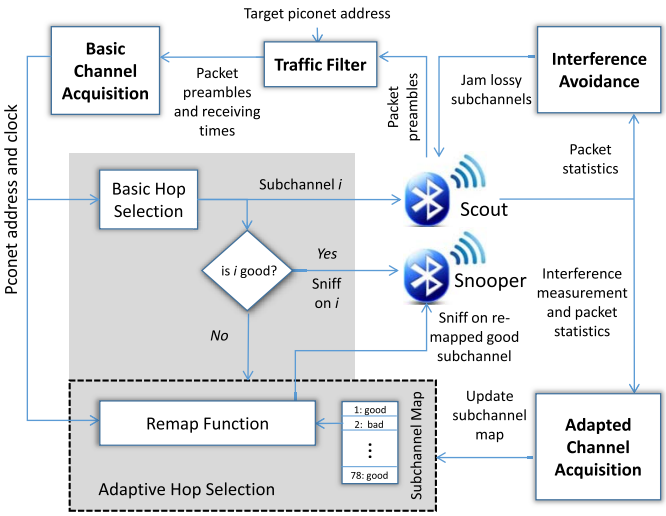


Fig. 1. The dual-radio architecture of BlueEar. Components in the gray area comprise a standard-compliant hop selection kernel.

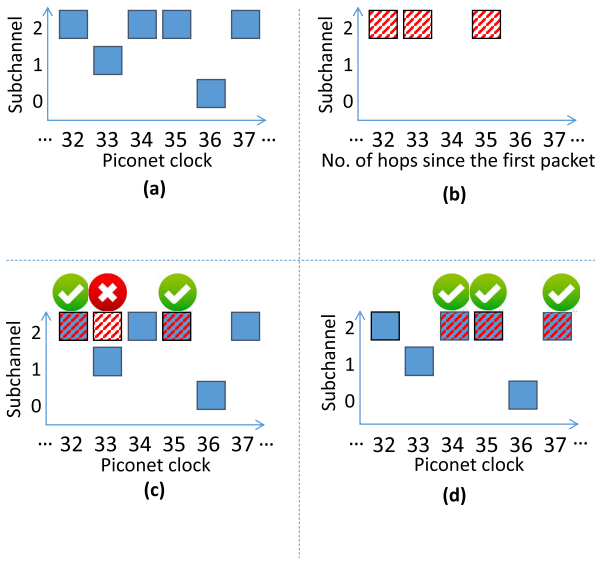


Fig. 2. An illustrative example of brute force search for clock acquisition.

(i) Traffic filtering. When multiple piconets coexist in the same environment, BlueEar separates their traffic based on the piconet address of captured packets. Specifically, the preamble of each Bluetooth packet carries a synchronization word, which is derived from the piconet address using an encoding function specified by the standard [2]. BlueEar employs the brute force algorithm proposed in [16] to extract piconet address from the synchronization word, and then filters out the packets whose piconet address mismatches the target piconet address specified by the BlueEar user.

(ii) Basic channel acquisition. To acquire the basic channel of the target piconet, the scout listens on an arbitrary subchannel to monitor the target's packet transmissions. After extracting piconet address from packet preamble, BlueEar derives the entire basic hopping sequence, and then reverses the piconet clock using an interference resilient, probabilistic clock matching algorithm. Specifically, the receiving times of captured packets are compared with the basic hopping sequence at different hopping phases, until a correct phase is found. The acquired piconet address and clock are then fed to a standard-compliant basic hop selection kernel to compute the basic channel. *(iii) Adapted channel acquisition.* To acquire the adapted channel, BlueEar needs to predict how the target reacts in dynamic spectrum context. To this end, the scout hops over all subchannels on the acquired basic channel to survey interference conditions, and monitors packet transmissions to infer the target's subchannel utilization. By fusing these measurements, BlueEar trains a subchannel classifier at run-time, which accurately derives the target's subchannel map despite vendor-dependent adaptive hopping behavior and the possible disparity between the spectrum contexts of the scout and the target. The subchannel map is then fed to a standard-compliant adaptive hop selection kernel for computing the adapted channel.

(iv) Interference avoidance. The snooper hops following the target on the adapted channel to sniff traffic. BlueEar handles complex interference in the crowded 2.4 GHz band using a selective jamming algorithm. Specifically, a loss detector is employed to monitor the sniffing performance of all subchannels. When substantial packet corruptions are detected on a subchannel i, the scout deliberately generates interference on i while the target visits i during hopping. Because of adaptive hopping, the target will be driven away from lossy subchannels where BlueEar observes poor sniffing performance. The objective is to manipulate the target's hopping to enforce implicit coordination.

V. Design of BlueEar

In this section, we present the design of key BlueEar components in detail.

A. Clock Acquisition

In the following, we define some terminology and present key concepts of clock acquisition.

Definition 1 Basic hopping sequence $\beta = (i_0, .., i_c, .., i_N)$ is an n-tuple of integers $i \in \{0, .., 78\}$ that represents subchannel index, c refers to the piconet clock, i.e. hopping phase, and $N = 2^{27} - 1$.

Definition 2 Observed hopping pattern $P = \{p_0, p_1, .., p_n\}$ is a set of captured packets.

Next, we first introduce our basic idea for reversing the piconet clock when the target hops in the basic mode. Second, we present a probabilistic matching approach to determine the piconet clock in the presence of interference. Finally, to accelerate clock acquisition, we introduce a maximum-likelihood (ML) based algorithm, which significantly reduces clock acquisition delay.

1) Brute Force Clock Acquisition: Because the entire hopping sequence β is known after the piconet address is extracted from packet preamble [16], it is possible to reverse the piconet clock c through a simple brute force search, which compares an observed hopping pattern with the entire hopping sequence at all phases to search for a match. Fig. 2 shows an illustrative example. At the beginning, the scout listens on a single subchannel to monitor the target's packet transmissions. As shown in Fig. 2, the scout listens on subchannel 2 where three packets are captured, which defines P. The receiving times of these

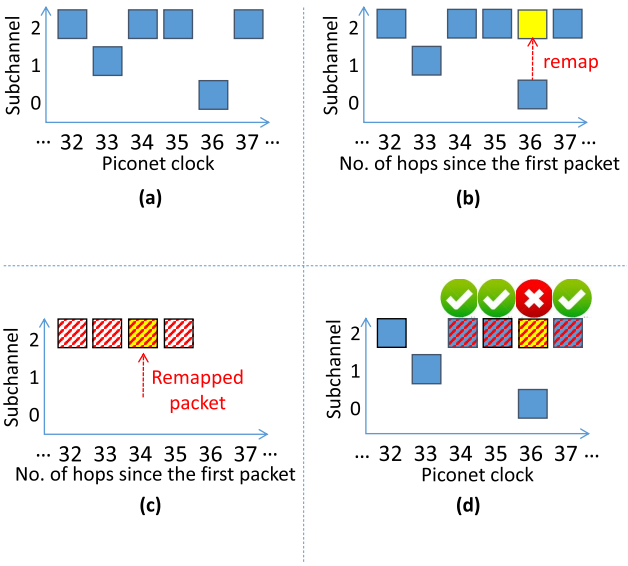


Fig. 3. The effect of subchannel remapping on brute force search based on the same example shown in Fig. 2.

packets compose a hopping pattern that describes how the target visits the monitored subchannel. As shown in Fig. 2(c) and (d), BlueEar then compares the observed hopping pattern with the entire basic hopping sequence at all phases. A match is found at clock 34.

Before capturing a sufficient number of packets, the observed hopping pattern may happen to match the basic hopping sequence at multiple clock values, resulting in clock ambiguity. Because the basic hopping sequence is pseudo-random, probability that an observed hopping pattern that comprises n packets matches the basic hopping sequence at an arbitrary clock can be computed as $\frac{1}{79^n}$. Therefore, clock ambiguity decreases as the number of captured packets increases.

2) Probabilistic Clock Matching: In the crowded 2.4 GHz band, Bluetooth devices rarely hop in the basic mode because of the impacts of interference from coexisting wireless devices [22]–[25]. To adapt spectrum use, Bluetooth modifies the basic channel by remapping bad subchannels subjected to interference with pseudo-randomly selected good subchannels. Since the adapted channel may differ from the basic channel in various phases, the hopping pattern observed by the scout may mismatch the basic hopping sequence even at the correct clock. Fig. 3 illustrates the impact of subchannel remapping using the same example shown in Fig. 2. As illustrated in the figure, brute force search may fail to find a perfect match due to the packet transmitted on the remapped subchannel.

To acquire the piconet clock c in the presence of interference, BlueEar leverages the following key observation. When comparing an observed hopping pattern P with the basic hopping sequence β at the correct clock, the ratio of mismatches should equal the ratio of remapped subchannels. As required by FCC, Bluetooth Classic must use at least 20 subchannels for frequency hopping [2]. Therefore, the ratio of remapped subchannels is at most $\frac{59}{79}$. Hence, if the ratio of mismatches at a clock c is significantly larger than $\frac{59}{79}$, then c is likely an incorrect clock.

Driven by the above observation, BlueEar employs a probabilistic clock matching (PCM) algorithm for clock acquisition. Instead of seeking a perfectly matched clock, BlueEar determines the correct clock by eliminating incorrect clocks based on the number of mismatches. A clock is identified as incorrect if the 95% confidence interval of its mismatch ratio exceeds $\frac{59}{79}$. Specifically, let β be the basic hopping sequence, $P = \{p_0, .., p_n\}$ be a set of observed packets, and $C = \{c_0, c_1, ..c_j\}$ and $D = \{d_0, d_1, ..d_j\}$ be the sets of clock candidates and the corresponding mismatches numbers, respectively. When comparing P with β, the PCM algorithm returns C and D, where d_i is the number of mismatches at clock candidate c_i. If c_i is the correct clock, the ratio of mismatches $\mu = \frac{d_i}{n}$, $n = |P|$, should be close to the ratio of unused subchannels. Based on the central limit theory, the distribution of $\sqrt{n}(\frac{d_i}{n} - \mu)$ should approach normality when n is reasonably large. The 95% confidence interval of μ can be estimated as $\frac{d_i}{n} \pm 2\frac{\sigma}{\sqrt{n}}$, where σ^2 is the variance. Therefore, clock c_i is determined as incorrect if $\frac{d_i}{n} - 2\frac{\sigma}{\sqrt{n}} \geq \frac{59}{79}$. When a new packet is captured, the algorithm updates the mismatch ratios for remaining clock candidates.

3) Accelerating Clock Acquisition: The PCM algorithm acquires the correct clock by gradually eliminating incorrect candidates; this may incur some time delay while waiting for new packets to be captured. To reduce clock acquisition delay, BlueEar resorts to estimate the piconet clock. In particular, given a set of observed packets P, the ML algorithm derives several subsets $P_1, .., P_z \subset P$, and invokes the PCM algorithm to obtain corresponding sets of clock candidates $C_1, .., C_z$. From the later sets, the algorithm finds probability distribution of candidates and identifies a candidate that maximizes the distribution function. The algorithm proceeds with the following steps.

Input: $P = \{p_0, .., p_n\}$ and $C = \{c_0, .., c_j\}$, and $b < n$
Step 1: Defines a counter $a = 1$, and
Step 2: Finds the subset $P_a \subset P$, $P_a = \{p_a, .., p_b\}$
Step 3: Invokes the PCM algorithm with P_a and obtains C_a
Step 4: Subtracts q from all elements in C_a. Intuitively, if c was the piconet clock at the time of transmitting p_0, then $c+q$ must be the piconet clock at the time of transmitting p_a, where q is the number of clock ticks between p_0 and p_a.
Step 5: Finds $C \leftarrow C \cup C_a$
Step 6: Increments a and b, while $b < n$, goto *Step 2*
Step 7: Finds probability distribution of candidates in C, and identifies c_{winner} that maximizes the distribution function.

BlueEar evaluates optimality of the ML estimator based on some *loss function L*, which we define as follows: $L(c_i) = d_i$, where c_i is an approximation for the piconet clock c, and d_i is the number of mismatches at clock c_i as defined earlier. According to the PCM algorithm, a clock candidate c_i, that is aligned with high mismatches d_i, is most likely to be eliminated from C as sufficient number of packets are overheard. Based on this observation, the ML estimator c_{winner} is considered if it is aligned with the minimum mismatches number d_m; otherwise, the ML algorithm waits for new packets to be captured.

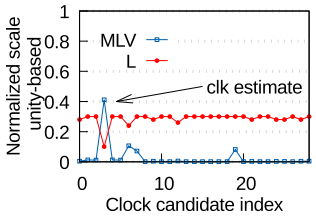


Fig. 4. Estimation of a piconet clock based on the ML algorithm and loss function $L(c_i) = d_i$.

Fig. 4 shows an example, where the ML algorithm successfully identifies the piconet clock based on a set of 20 observed packets. In this figure, it is clear that the winner candidate aligns with the minimum value of the loss function L.

B. Subchannel Classification

The adaptive hopping of Bluetooth is guided by a subchannel map that classifies subchannels as good and bad based on dynamic interference conditions. To acquire the adapted channel, BlueEar employs a subchannel classifier, which infers how the target classifies subchannels. The subchannel classifier must meet the following requirements.

(i) Accuracy. When a subchannel is misclassified, the snooper may hop to a wrong subchannel different from the one used by the target. Poor subchannel classification accuracy may result in substantial packet misses.

(ii) Responsiveness. In dynamic spectrum contexts, the subchannel classifier must be responsive to the change of interference conditions.

In the following, we present two complementary subchannel classification algorithms, and discuss their advantages and limitations in achieving the above goals. We then discuss how BlueEar integrates the two algorithms for subchannel classification.

1) Packet-Based Classifier: Design: As Bluetooth only transmits on good subchannels, it is possible to infer the status of a subchannel based on its *packet rate*, which indicates how frequently the target transmits on a subchannel. To measure packet rates, the scout hops over all 79 subchannels on the acquired basic channel to monitor the target's packet transmissions. For each subchannel i, BlueEar computes its packet rate as $r_i = \frac{q_i}{v_i}$, where q_i is the number of packets received on i, and v_i is the number of times the scout visits i. When a subchannel is classified as bad by the target, the packet rate measured by the scout will be substantially lower than that of good subchannels.

A key challenge to achieve accurate packet-based classification is that packet rates differ significantly across different applications ($e.g.$, data transfer, audio, and keystroking, etc.), and may vary with time depending on the traffic dynamics in the target piconet. To address this challenge, we leverage the fact that, as required by FCC, Bluetooth Classic uses at least 20 subchannels for frequency hopping [2]. As a result, the 20 subchannels that have the highest packet rates can be used as a reference to estimate the current packet rate of the target piconet. Driven by this observation, the packet-based classifier identifies bad subchannels using the following algorithm.

- *Step 1*: Initially, the 20 subchannels that have the highest packet rates are classified as good. Let $\mathcal{R}_g = \{r_{i_1}, ..., r_{i_{20}}\}$ be the set of their packet rates.
- *Step 2*: In remaining unclassified subchannels, the classifier searches for the one with the highest packet rate. Denote this subchannel as i, and its packet rate as r_i.
- *Step 3*: The packet-based classifier determines whether subchannel i is bad by checking if r_i is an outlier of $\mathcal{R}_g$. If r_i is an outlier, i and all remaining subchannels of even lower packet rates are classified as bad. Otherwise, i is classified as good and its packet rate r_i is inserted to $\mathcal{R}_g$. The algorithm then goes back to step 2 until a bad subchannel is identified.

We determine if $r_i = \frac{q_i}{n_i}$ is an outlier of $\mathcal{R}_g$ as follows. Let r_g be the average packet rate of $\mathcal{R}_g$. Assuming subchannel i is good, probability that the target transmits less than q_i packets after v_i visits can be computed as,

$$\rho_i = \sum_{n=0}^{q_i} \binom{v_i}{n} (1 - r_g)^{v_i - n} r_g^n \tag{1}$$

We identify outliers under a given confidence level, denoted as θ. Subchannel i is classified as bad if $\rho_i \leq 1 - \theta$.

Fig. 5 gives two examples of packet-based subchannel classification for data and audio traffic in different spectrum contexts. The upper figure shows the packet rates measured on 79 subchannels. Low packet rates are observed on bad subchannels subjected to interference. The bottom figure shows the probability scores ρ_i for each subchannel i calculated using Eq. (1). As shown in the figure, the packet-based classifier reliably identifies bad subchannels despite the significant variation of packet rates across different applications.

Discussion: By classifying subchannels based on packet rates, packet-based classifier offers two advantages: *(i)* it works efficiently across different Bluetooth devices despite vendor-dependent subchannel classification methods, *(ii)* the classification is not affected by the disparity between the spectrum contexts of the target and BlueEar. However, packet-based classifier is less responsive in dynamic spectrum context because a subchannel cannot be classified as good or bad before overhearing a sufficient number of packets. As a result, packet-based classifier may perform poorly when subchannel status changes fast with time-varying interference.

2) Spectrum Sensing-Based Classifier: Design: Since Bluetooth piconet classifies subchannel based on interference conditions, subchannel i is likely bad if strong interference is measured on i. Driven by this observation, spectrum sensing-based classifier infers subchannel status based on interference measurements. When hopping on the basic channel, the scout measures interference power on each subchannel. The interference condition on a given subchannel is characterized using the probability density of interference power. Fig. 6 illustrates two examples measured by the scout on clean and dirty subchannels. On both subchannels, the environment noise floor is found at -95 dBm. An interference source whose signal power ranges from -45 dBm

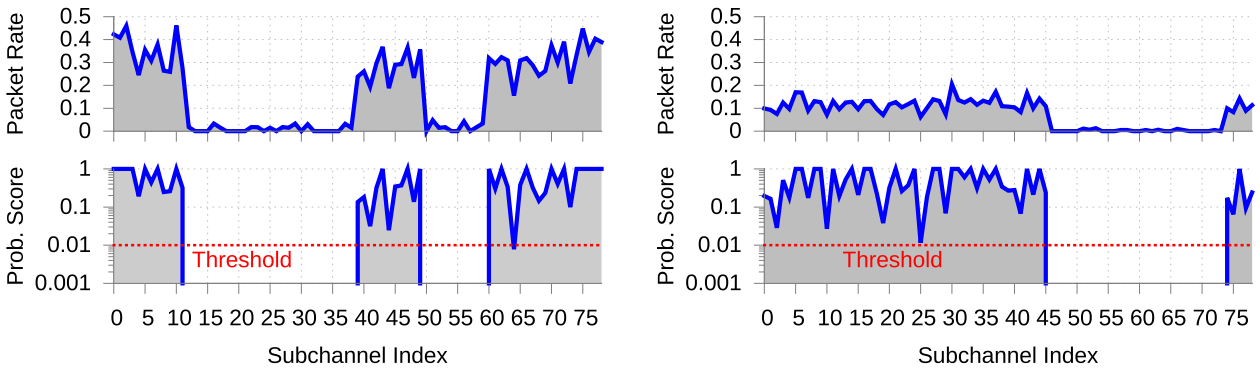


Fig. 5. Running examples of packet-based subchannel classification for data and voice traffic under in different spectrum contexts. The packet-based classifier calculates a probability score based on Eq. (1) to determine subchannel status. A subchannel is classified as bad if the probability score is below the pre-defined threshold. (a) Data. (b) Audio.

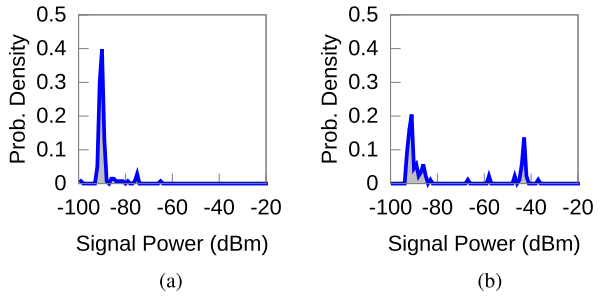


Fig. 6. Probability densities of interference power measured on clean and dirty subchannels. (a) Clean subchannel. (b) Dirty subchannel.

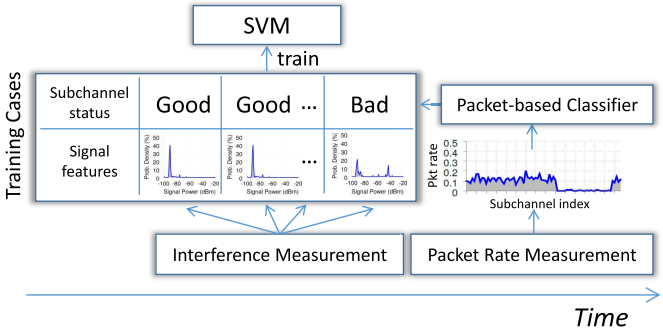


Fig. 7. Time-domain illustration of run-time training of the spectrum sensing-based classifier.

and -40 dBm can be observed in Fig. 6(b). The interference source is active in about 15% of time.

Based on interference measurement, the spectrum sensing based classifier employs an SVM to determine subchannel status. The SVM takes as input a feature vector obtained by discretizing the probability density of interference power based on $\mathcal{X}_i = \{x_{-100,i}, x_{-99,i}..., x_{-20,i}\}$, where $x_{s,i}$ is the probability of observing an interfering signal power of s dBm on subchannel i. $\mathcal{X}_i$ characterizes interference condition between -100 dBm and -20 dBm, which is sufficient to capture the activities of all interfering sources in practice.

Discussion: Although spectrum sensing-based classifier is responsive to time-varying interference conditions, achieving satisfactory accuracy is difficult because *(i)* depending on the location of interference source, the spectrum measured by the scout may differ from the one observed by the target; *(ii)* the subchannel classification method adopted by the target is vendor-dependent and may differ across different devices. To address these limitations, the spectrum sensing-based classifier must be trained at run-time.

3) Hybrid Classifier: For accurate and responsive classification of subchannel status, BlueEar employs a hybrid classifier that combines the complementary packet- and spectrum sensing-based classifiers. At run-time, the hybrid classifier uses packet-based classification results to train a spectrum sensing-based classifier, which learns the vendor-dependent subchannel classification method of the target. After training, BlueEar fuses the outputs of packet- and spectrum sensing-based classifiers to infer subchannel status.

To train the spectrum sensing-based classifier, BlueEar labels interference conditions measured by the scout using the outputs of packet-based classification. Note that packet-based classifier infers subchannel status based on packet rates derived from the history of overheard packets, therefore its result may lag behind the true subchannel status. To compensate this delay, BlueEar composes training cases by labeling $\mathcal{X}_f$ using packet-based classification results obtained at a later time point. Fig. 7 illustrates this training procedure in time-domain.

For subchannel classification, the hybrid classifier fuses the outputs of packet- and spectrum sensing-based classifier based on the responsiveness and confidence of results. The soft-output of SVM is utilized to characterize the confidence of classification. The soft-output of SVM is a log-likelihood ratio computed as $\lambda_i = \log \frac{\rho_i}{1-\rho_i}$, where ρ_i is the probability that i is good, and $|\lambda_i|$ indicates confidence. Because spectrum sensing-based classifier is more responsive to dynamic spectrum context, the hybrid classifier adopts the output of SVM as the final classification result if its confidence $|\lambda_i|$ is higher than a predefined threshold. Otherwise, the output of packet-based classifier is adopted.

C. Selective Jamming

In the crowded 2.4 GHz frequency band, BlueEar is subjected to the interference of coexisting wireless devices, especially the prevalent 802.11 based WLANs. Unlike authorized

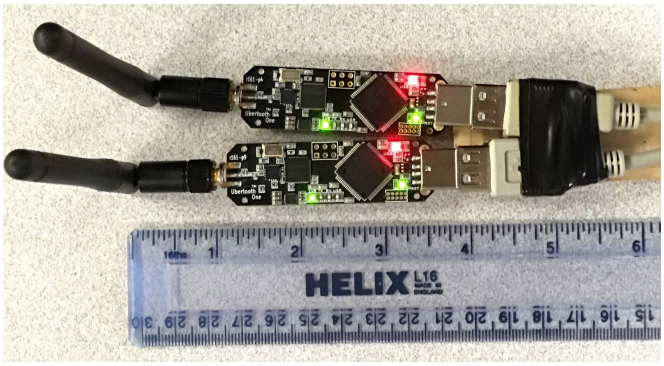


Fig. 8. BlueEar prototype that consists of two Ubertooths [13].

Bluetooth devices that can handle such interference by coordinating their hopping, designed as a passive packet sniffer, BlueEar cannot coordinate with the target, which may result in poor sniffing performance. BlueEar mitigates the impacts of such interference using a selective jamming algorithm. In the following, we present the algorithm design in detail, and then discuss the impact of jamming on 802.11 devices.

When the interference causes substantial packet corruptions on a subchannel i, the scout deliberately generates interference on i while the target visits i. Because of adaptive hopping, the target will be driven away from subchannels i, resulting in implicit coordination. To this end, BlueEar employs a loss detector to identify subchannels subjected to hidden interference. Whenever the scout overhears a packet, it checks packet integrity using CRC, and then sends the result to the loss detector. For each subchannel, the loss detector employs a moving window to compute the ratio of corrupted packets. The scout is commanded to jam a subchannel if the packet corruption ratio is higher than a predefined threshold. To effectively drive the target, a class one Bluetooth radio capable of high power transmission is employed to implement the scout.

Discussion: Despite deliberately generating interference in the 2.4 GHz band, the impacts of selective jamming on 802.11 devices is very limited because of two reasons. First, there are considerably low chances for collisions between 802.11-based frames and selective jamming frames. As explained earlier, selective jamming is equivalent to Bluetooth class 1 transmission, and hence, probability of collision with 802.11 WLAN is equivalent to that of Bluetooth. A previous study [26] highlights that in worst case scenario, where the three 802.11 non-overlapped channels are consistently occupied, the probability of collision with Bluetooth is less than 0.2. However, this probability considerably reduces if we consider the following factors: *(i)* the scenario of occupying the three 802.11 non-overlapped channels is rare;

(ii) the study [26] considers a low data rate (5.5Mbps) for 802.11 link. With adoption of OFDM higher data rates, like 48 Mbps, frame-in-air-time becomes shorter than that of 5.5Mbps rate. As a result, abundant white space between 802.11 transmissions is expected. A recent study [25] confirms this and shows that 802.11 traffic is highly bursty and frames are clustered together with short intervals of time (typically less than $1ms$), while periods between clusters are significantly longer; *(iii)* life time of a jamming session is 2 sec in average, where our experiments show that Bluetooth reacts to interference within 4 sec at most; and finally *(iv)* BlueEar only jams in the presence of hidden interference, which is a special scenario for BlueEar. Second, assuming the case of collision with 802.11-based frame, a previous study [27] shows that 802.11 WLAN is robust against narrow band, short-lived interference (selective jam occupies 1MHz). Accordingly, we conclude that selective jamming has very limited effect on 802.11-based WLAN.

VI. Implementation

In this section, we present the implementation of BlueEar in detail. As shown in Fig. 8, we employ two Ubertooths [13] to implement the scout and the snooper, and then interface them to a controller running on a Linux laptop. Computation intensive tasks like clock acquisition and subchannel classification are implemented on the laptop. Time-sensitive components like basic and adaptive hop selection are implemented by extending the firmware of Ubertooth. In addition, we identify critical issues in Ubertooth firmware that severely degrade its performance during frequency hopping, and present novel solutions to address these flaws. We note that although our current prototype is built based on Ubertooth, the design of BlueEar is platform-independent and can be easily ported to other systems.

A. Ubertooth-End Implementation

Ubertooth is an open source 2.4 GHz wireless development board that costs around $80 per unit [13]. Each Ubertooth is equipped with an LPC17xx microcontroller, and a low-power Bluetooth-compliant CC2400 transceiver connected to a 4-inch 2.2 dBi antenna. Ubertooth is capable of transmitting at 22 dBm, which assures the effectiveness of selective jamming.

The original firmware of Ubertooth is implemented in 823 lines of C code, which implements DMA management, basic hop selection, and carrier sense, etc. Data in DMA buffer is framed into USB packets and forwarded to the host. However, the original firmware lacks support for adaptive hop selection and run-time clock synchronization. In addition, we find that the firmware is poorly optimized for real-time frequency hopping. In particular, because of resource contention among multiple tasks, subchannel switching may be improperly delayed (e.g., by USB packet streaming, which typically takes around $50\mu s$ according to our measurements). Such delay will break the hop synchronization between BlueEar and the target.

We extend the firmware of Ubertooth using 400 lines of C code, which implement the following functions.

(i) Run-time clock synchronization. To hop following the basic and the adapted channel of the target, the scout and the snooper must synchronize their native clocks with the target's piconet clock, $i.e.$ their clocks must have the same value and tick at the same time. Run-time clock synchronization is imperative because the clocks of the scout, the snooper, and the target may have clock skews [28], which make them run at different rates, accumulating a drift that breaks hop synchronization.

The extended firmware accomplishes clock synchronization as follows. After clock acquisition, the firmware receives a piconet clock value from the clock acquisition component. The clock value is used as the initial value of a native clock, which is obtained by programming a 10MHz timer provided by LPC17xx into a 27-bit counter that ticks every hop. To assure that the native clock and the piconet clock tick at the same time, the extended firmware leverages the fact that Bluetooth packets are always transmitted immediately after clock ticks. Therefore, the receiving times of overheard packets can be utilized as a clock reference to correct clock drift. To avoid packet miss caused by remaining clock drift, the native clocks of the scout and the snooper are programmed to tick 1 μs earlier than the target.

(ii) Adaptive hop selection. The firmware of the snooper implements a standard-compliant adaptive hop selection kernel. The kernel takes three inputs, including the inferred subchannel map, the piconet address obtained from the controller, and the value of the native clock. The inferred subchannel map is updated every second.

(iii) Task scheduler. To assure real-time hopping performance, the extended firmware schedules tasks based on their time sensitivities. Hop selection and subchannel switching are given the highest priority to assure the right hop synchronization. USB packet streaming and carrier sense are given the second and lowest priority, respectively. Tasks are executed in the interval between subchannel switching in the order of their priorities.

B. Controller Implementation

The controller implements compute intensive tasks, including packet decoding, clock acquisition, subchannel classification, and jamming subchannel selection. In addition, it interacts with the scout and the snooper via high-speed USB. These tasks are implemented as multiple processes, which share a 3 KB of memory for coordination and parameter exchange. For packet-based subchannel classification, the controller implements the algorithm described in Section V-B1 in 53 lines of C code. A confidence level of 99% is used to assure accurate identification of bad subchannels. The spectrum sensing-based classifier is implemented based on SVM^{light}, which is an open-source computation-efficient SVM library [29]. The spectrum sensing-based classifier takes about 51.2 KB of memory at run-time. To compensate the delay of packet-based classification when training the SVM (as explained in Section V-B2 and Fig. 7), the controller uses packet-based classification results obtained in $t + 4s$ to label the signal features measured at t. This choice is motivated by our empirical measurements, which show that most Bluetooth devices update subchannel map every $4s$. The hybrid classifier chooses the result of SVM as output if the confidence of SVM is higher than 90%. Otherwise the output of packet-based classifier is adopted. The controller is responsible for decoding the raw bit stream received from Ubertooth. Packet integrity is examined by checking the received CRC. A subchannel is jammed if the ratio of corrupted packets is higher than 10%.

VII. BlueEar Performance

In this section, we present a thorough evaluation of BlueEar performance. In the following, we first introduce our experimental methodology, and then discuss experiment results in detail.

A. Experimental Methodology

We study BlueEar performance when sniffing data transfer and audio streaming, which are representative Bluetooth traffics that have distinct packet rates. Data traffic is generated by transferring data files between two laptops equipped with Broadcom dongles. Audio traffic is generated by playing an audio file on a laptop equipped with CSR dongle, and a Samsung Bluetooth headset is set as the audio sink. We conduct experiments in an office building under the interference of a large-scale 802.11 based WLAN, as well as in various controlled settings to benchmark BlueEar performance under specific interference patterns.

We evaluate the synchronization delay, the subchannel classification accuracy, and the packet capture rate of BlueEar. The synchronization delay is measured as the time needed to determine the correct piconet clock. To measure subchannel classification accuracy and packet capture rate, we log the groundtruth subchannel map and packet rates at the piconet master using a script written based on hcitool. The host of BlueEar is connected with the piconet master via an Ethernet link. The instantaneous readings of groundtruth subchannel map and packet rates are transferred to the BlueEar host using UDP. We compare BlueEar with a set of baselines. First, we compare the hybrid subchannel classifier proposed in Section V-B3 with pure packet-, and spectrum sensing-based classifiers (abbreviated as `Pkt' and `SS' in figures). The SVM of pure spectrum sensing-based classifier is trained offline in a controlled setting consisting of an 802.11 access point (AP) and a Broadcom piconet. During training, we tune the power and temporal pattern of 802.11 transmissions to introduce different interference conditions, which enables extensive profiling of the adaptive hopping behavior of the Broadcom device. We then use the trained classifier to predict the subchannel maps in data and audio tests, where the piconets are formed using different Bluetooth devices. Second, to evaluate the gain of selective jamming, we compare BlueEar with a baseline where the selective jamming is disabled. Third, we compare the packet capture rate of BlueEar with that of an existing Ubertooth-based sniffer [13], which operates in the basic hopping mode, and is oblivious to the adaptive hopping behavior and the impacts of interference.

B. Synchronization Delay

We first evaluate the delay incurred when synchronizing BlueEar with the target piconet. The dominant component of this delay is introduced by clock acquisition, during which the scout listens on a single subchannel until it captures enough packets to reverse the piconet clock. We benchmark clock acquisition delay in different spectrum contexts where the target exhibits diverse hopping behaviors. Our experiments are

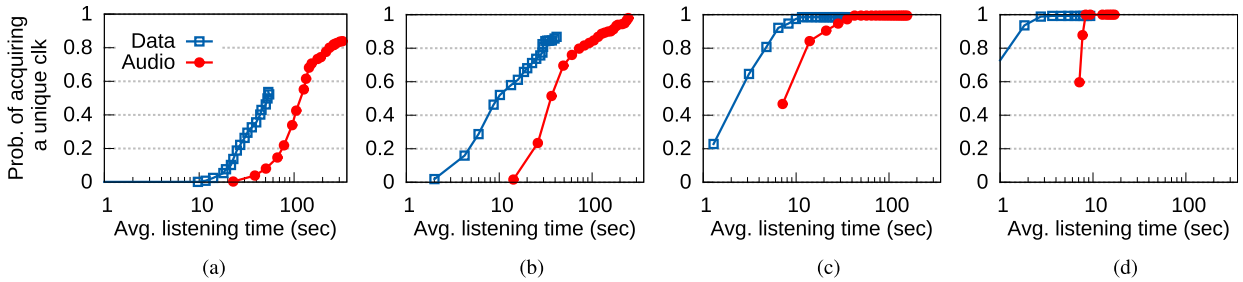


Fig. 9. Clock acquisition delay when sniffing data and audio traffics in different spectrum contexts (characterized by the percentage of bad subchannels at the target piconet). (a) No bad subchannels. (b) 25% bad subchannels. (c) 50% bad subchannels.

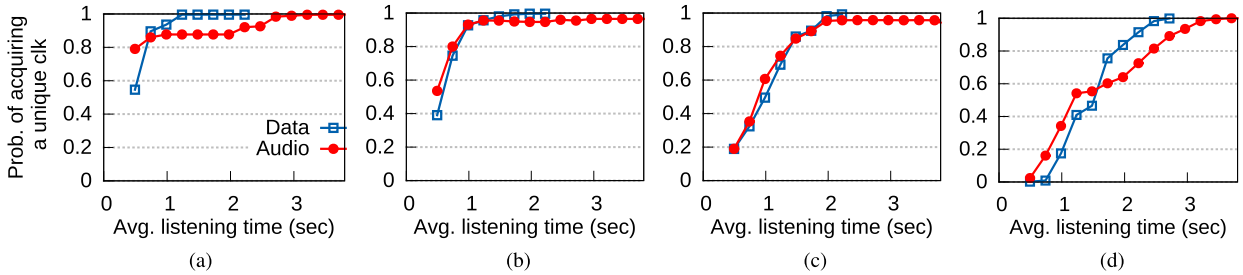


Fig. 10. Clock acquisition delay based on the ML algorithm, which is evaluated in different spectrum contexts (characterized by the percentage of bad subchannels at the target piconet). (a) No bad subchannels. (b) 25% bad subchannels. (c) 50% bad subchannels. (d) 75% bad subchannels.

conducted in a controlled setting where three 802.11 access points (APs) are deployed around the target. Each AP occupies one of the three non-overlapping 20 MHz channels. When all APs are active, they create a crowded spectrum where about 75% subchannels of the target piconet are bad.

Fig. 9 shows the probability of successfully determining the piconet clock, based on the PCM algorithm, as the listening time of the scout increases. We observe that clock acquisition delay when sniffing audio traffic is higher than that when sniffing data traffic. This is mainly because of the lower packet rate of audio traffic. Interestingly, the delay substantially reduces when the spectrum becomes more crowded. This is because, when more subchannels are occupied by 802.11 APs, the target piconet has to use fewer subchannels for packet transmissions, resulting in an increased packet rate on the subchannel monitored by the scout. Specifically, when 75% of subchannels are occupied by 802.11 APs, the clock acquisition delay is less than 10s in both data and audio tests. The result implies that Bluetooth traffic sniffers can substantially reduce its synchronization delay using deliberately planned interference.

Now we evaluate the ML algorithm performance. We consider clock acquisition delay as an evaluation metric. We compare the ML algorithm performance with that of the PCM algorithm when acts alone. Fig. 10 shows the probability of successfully estimating the piconet clock as the listening time increases. To compare, we observe that the ML algorithm outperforms the PCM algorithm, as in Fig. 9. The ML algorithm significantly reduces clock acquisition delay to less than $4s$. Similar to the PCM algorithm, we observe that clock acquisition delay when sniffing on audio traffic is higher than that when sniffing on data traffic. In contrast, the ML algorithm requires more time to estimate the piconet clock when more subchannels are occupied by 802.11 traffic, which in turn maximizes number of remapped packets. Specifically, when a set of observed packets P starts with a remapped packet, there is a low chance that the true clock is presented in the set of candidates C, i.e. all candidates are false. Alternatively, when P starts with a packet that belongs to the basic hopping sequence, the true clock must appear in C. As a result, the ML algorithm requires more packets, i.e. more listening time, if the number of remapped packets is increased. Based on this finding, the design of a countermeasure should consider increasing number of remapped packets in Bluetooth traffic to prevent passive attacker, like BlueEar, from sniffing on Bluetooth link.

C. Fast Varying Spectrum Context

We now evaluate BlueEar in dynamic spectrum contexts where the subchannel map of the target is modified frequently. The transmission of AP is turned on/off every a couple of seconds to create a fast varying spectrum context, which causes the target to modify its subchannel map every update period.

Fig. 11 evaluates subchannel classification accuracy based on false positive (FP) and false negative (FN) rates. As expected, the packet-based classifier performs the worst because of its poor responsiveness. In comparison, the spectrum sensing-based classifier offers better performance when sniffing data traffic, but fails to maintain its accuracy when sniffing audio. This is because the spectrum sensing-based classifier is trained offline against Broadcom devices, and it fails to predict the adaptive hopping of the CSR devices used in the audio test. We also observe that the hybrid classifier performs best among the three classifiers. In particular, the FP and FN rates are lower than 8% in both data and audio tests. Fig. 12 further compares the packet capture rates when BlueEar uses the three classifiers to predict adaptive hopping. Similar with the results shown in Fig. 11, the hybrid classifier is able to maintain the best packet capture rate, which is higher than 90% in both data and audio tests.

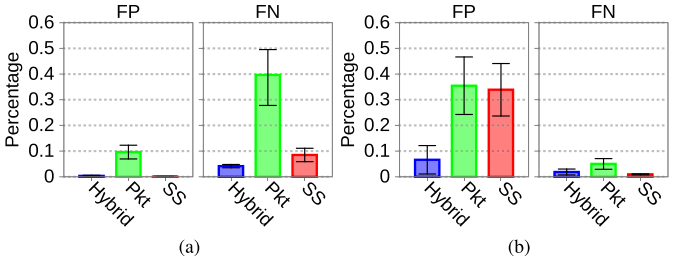


Fig. 11. Subchannel classification accuracy in fast varying spectrum context. (a) Data. (b) Audio.

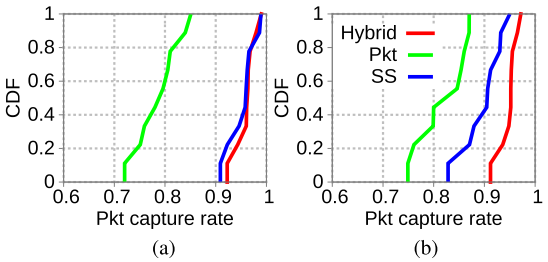


Fig. 12. Packet capture rate in fast varying spectrum context. (a) Data. (b) Audio.

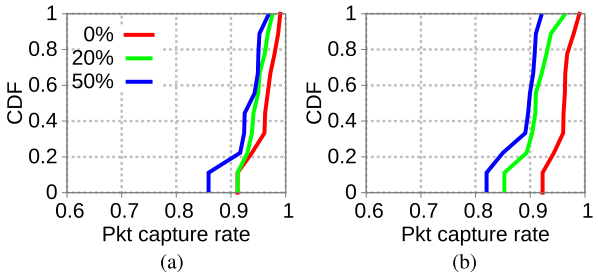


Fig. 13. Packet capture rates in crowded spectrum. (a) Data. (b) Audio.

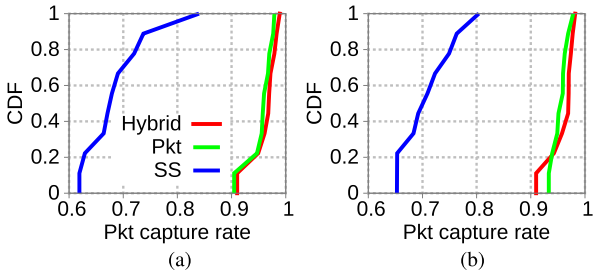


Fig. 14. Packet capture rates (exposed interference). (a) Data. (b) Audio.

D. Hidden and Exposed Interference

We now evaluate the packet capture rate of BlueEar in the presence of hidden and exposed interference. Fig. 19 evaluates the gain of selective jamming in the presence of hidden interference, where an RF signal does not interfere with the target, but collide with captured packets at BlueEar. The experiments are conducted in a controlled setting where an 802.11 device generates hidden interference starting from the 100-th second. When selective jamming is enabled, BlueEar is able to maintain high packet capture rates, despite a short period of performance drop before the target piconet reacts to the generated interference. In comparison, when selective jamming is disabled, BlueEar suffers substantial performance degradation, where the packet capture rate is reduced to about 60% from higher than 95%.

We further evaluate BlueEar in the presence of exposed interference, where an RF signal interferes the target, but is too weak to be measurable at the scout. Exposed interference results in significant disparity between the spectrum contexts at BlueEar and the target. We conduct experiments in a controlled setting where an 802.11 device is deployed to generate exposed interference. During our experiment, the 802.11 device keeps active, and interferes with 20 of 79 subchannels of the target piconet. Fig. 13 compares the subchannel classification accuracy of the hybrid, the packet-based, and the spectrum sensing-based classifiers. Different from what we observed in Fig. 11, the spectrum sensing-based classifier suffers high FP in both tests. This is because spectrum sensing-based classifier relies on the interference measurements of the scout to identify bad subchannels, which works poorly when the interference signal cannot be detected by the scout. In comparison, hybrid and packet-based classifiers are able to maintain extremely low FP and FN rates and high packet sniffing performance, as shown in Fig. 14.

E. Crowded Spectrum

We then evaluate the misclassification rate of the hybrid classifier in spectrum contexts with different levels of crowdedness. The FPs, FNs, and overall misclassification rates are shown in Fig. 15. We observe that the hybrid classifier maintains high accuracy despite the increasingly crowded spectrum. In particular, when 50% of subchannels are bad, the overall misclassification rate is below 8% in both data and audio tests. Fig. 16 shows the packet capture rates measured in the same experiment. As shown in the figure, BlueEar captures more than 90% packets in both data and audio tests.

F. Ambient Interference

We further evaluates the performance of BlueEar in an office building under the ambient interference from a large-scale 802.11 based WLAN. Fig. 17(a) shows the packet capture rates measured at four randomly selected locations, where BlueEar is deployed at 10m away from the target piconet. In all of the four locations, the number of active 802.11 APs is higher than 20 during our experiments. We compare BlueEar with an existing Ubertooth-based sniffer [13] that hops following the basic channel of the target. Because the basic hopping sniffer is oblivious to the adaptive hopping behavior, it suffers 50% to 25% packet misses. In comparison, BlueEar is able to maintain a packet capture rate higher than 95% at all of the four locations.

Fig. 17(b) evaluates the packet capture rate at site D when BlueEar is deployed at different distances from the target. The disparity in spectrum contexts is expected to increase as BlueEar moves away from the target. However, thanks to the high-performance subchannel classifier, BlueEar is able to capture more than 85% packets even when it is 27m away from the target.

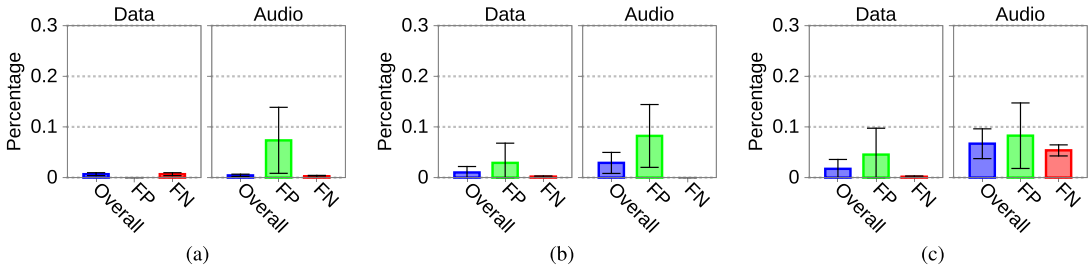


Fig. 15. Subchannel classification accuracy in crowded spectrum (characterized by the percentage of bad subchannels at the target piconet). (a) No bad subchannels. (b) 25% bad subchannels. (c) 50% bad subchannels.

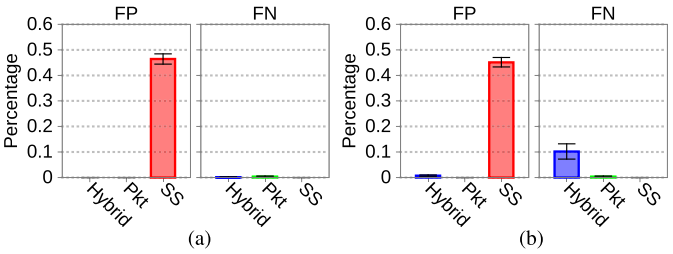


Fig. 16. Subchannel classification accuracy (exposed interference). (a) Data. (b) Audio.

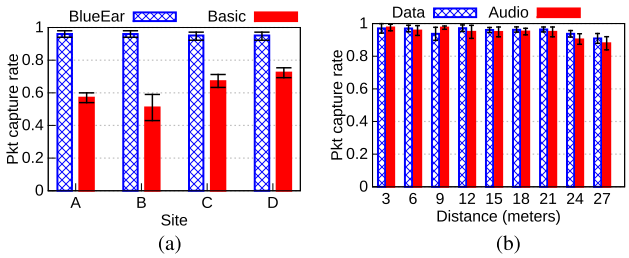


Fig. 17. Packet capture rates under the ambient interference: (a) different locations in an office building (interference of a large-scale 802.11 based WLAN); (b) different distances.

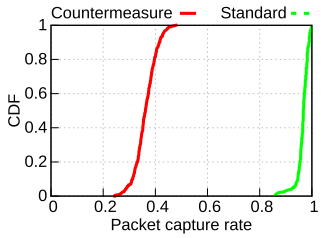


Fig. 18. BlueEar pkt capture rates: standard and countermeasure.

VIII. Implications of BlueEar

In this section, we discuss the privacy implications of BlueEar in detail.

A. Implications on BLE Privacy

Although the current prototype of BlueEar is developed for sniffing classic Bluetooth, its methodology has significant implications on the privacy of BLE. In the following, we first briefly introduce the hopping protocol of BLE, and elaborate on the impacts of BlueEar on BLE privacy breach. The hopping protocol of BLE defines a physical channel, that hops over 37 data subchannels in the open 2.4 GHz spectrum starting from 2.402 to 2.48 GHz. All subchannels are equally spaced with 2 MHz of bandwidth. The connection state of BLE can be characterized as a set of connection events. During the initialization of a connection, the master defines *(i)* connection interval –a multiple of $1.25ms$ ranging from $7.5ms$ to $4.0s$ that defines the event lifetime; *(ii)* transmission window size –a multiple of $1.25ms$ that defines the size of transmit window, i.e. packet size; and *(iii)* hop increment inc –a random value ranges from 5 to 16. The basic channel hopping is characterized by $\mathcal{K}(c, inc)$, where $\mathcal{K}(.)$ is the hop selection kernel, and c is the index of current subchannel. At the first connection event, the first subchannel is defined to be zero [2], the channel sequence repeats itself whenever subchannel zero is visited. Similar with Bluetooth classic, BLE adopts adaptive hopping mode, where the basic channel is modified to adapt spectrum use in the presence of ambient interference. The adaptive channel is defined by a subchannel map that classifies the data subchannels into good and bad. If the basic kernel $\mathcal{K}(c, inc)$ selects a bad subchannel, a remapping procedure is invoked to calculate a *remapped subchannel index*. The master maintains the subchannel map and it notifies slave(s) about any updates [2].

The hopping protocol of BLE is different from that of Bluetooth Classic in two aspects. First, basic channel sequence of BLE is characterized by a random value of the hop increment inc. In contrast, basic channel sequence of Bluetooth Classic is characterized by the piconet address. The second difference between BLE and Bluetooth Classic is hopping phase, which is not defined in BLE hopping protocol. Unlike the basic channel sequence of Bluetooth Classic, which repeats itself every about 23 hours, BLE basic sequence repeats itself whenever subchannel zero is visited. Due to power constraints, the hopping protocol of BLE is much simpler than that of Bluetooth Classic, making BLE basic sequence easier to compromise.

As BLE becomes pervasive, the privacy leakage of BLE devices is becoming an increasing concern. Although BlueEar is designed for Bluetooth Classic, it has significant impacts on the privacy leakage of BLE devices, which calls for further research to further investigate and enhance the privacy of BLE. In particular, the key components of BlueEar system, including subchannel classification and selective jamming, are independent of the hopping protocol. These techniques can be directly ported to BLE as well as other adaptive hopping systems without modifications. Unfortunately, the clock acquisition component, the hop selection subsystem, and the packet decoder of our prototype are specifically engineered for

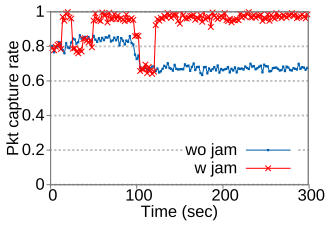


Fig. 19. The gain of selective jamming under hidden interference.

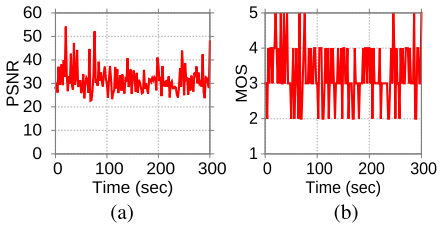


Fig. 20. Audio quality observed by BlueEar (without countermeasure). (a) PSNR. (b) MOS.

Bluetooth classic, which make the current version of BlueEar incompatible with BLE.

B. Impacts on Privacy Breach

Previous research has shown the possibilities of cracking Bluetooth encryption and compromising user privacy [5]–[11]. A prerequisite of these attacks is to passively sniff Bluetooth traffic. Existing attacks [7], [8] employ prohibitively expensive commodity sniffers, which limits their widespread distribution. The BlueEar system we demonstrated in this paper may unleash such attacks, making them a real issue for off-the-shelf Bluetooth devices.

To further understand the impacts of BlueEar on privacy leakage, we conduct the following two experiments. First, we study the impacts of BlueEar when successfully eavesdropping on a Bluetooth data link. To quantify such privacy leakage, we adopt the *seriousness of privacy leakage* model $S(P, L) = \Sigma w_i.p_i.l_i$, where p_i is the *privacy unit*, w_i is the weight of p_i, and $l_i = 1$ when p_i is leaked, and 0 otherwise [30]. As data packet carries potential sensitive information, rather than other packets, we define the components of privacy unit based on data packets and assign their weights as: data packet (weight=95%), and other types of packets (weight=5%).

Fig. 22 quantifies seriousness of privacy leakage, where S is calculated every 2 seconds based on number of packets captured by BlueEar.

Second, we study impacts of BlueEar when eavesdropping on audio traffic, like eavesdropping on a speech conversation, which is known to be challenging because audio streams are extremely sensitive to packet loss. The experiment proceeds as follows. *(i)* We generate audio traffic over a Bluetooth link and deployed BlueEar to sniff on traffic; *(ii)* as BlueEar collects real-time trace, it reports packet loss rates every 2 seconds and logs locations of missing packets; and *(iii)* we simulate the audio packet stream on a PC and replayed each missing packets with it's preceding one.

We quantify the quality of the simulated audio stream, which should be equivalent to the quality of the eavesdropped

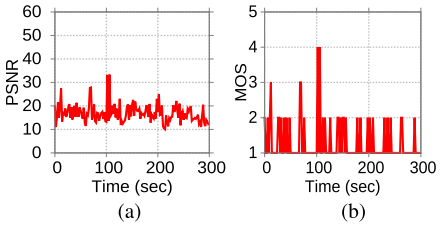


Fig. 21. Audio quality as the target implements countermeasure. (a) PSNR. (b) MOS.

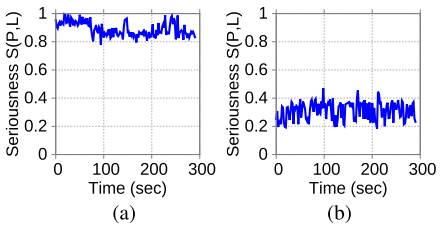


Fig. 22. Seriousness of privacy leakage evaluated: (a) standard; and (b) countermeasure.

audio, based on peak signal to noise ratio (PSNR) and mean opinion scores (MOS). To obtain the later, we map PSNR values into MOS, which is categorized as five voice qualities including, excellent, good, fair, poor, and bad as proposed in [31]. Fig. 20 evaluates audio quality based on PSNR, that is calculated every 2 seconds, and MOS. We observe that BlueEar maintains high audio quality, where average PSNR is 35 and MOS scores are higher than fair in 81% of the time.

C. Practical Countermeasure

To counteract sniffing systems, like BlueEar, we propose a practical countermeasure approach. We implement the proposed countermeasure as a user-space script on Bluetooth master and it requires no modifications to existing slaves. The key idea of the approach is to frequently flip status of randomly selected subchannels (good becomes bad, and vice versa). Such random flipping interferes the subchannel classifier, making it hard for the sniffer to learn the adaptive hopping sequence. Thus, the sniffer experiences poor data quality. To evaluate the effectiveness of the countermeasure, we run the following experiment. The countermeasure randomly selects $n - 20$ subchannels to be flipped, where n is the total number of good subchannels; this complies with the FCC rule, that requires at least 20 subchannels to be used by Bluetooth. We wrote a user-space script that updates the subchannel map is every $200ms$. The script interacts with BlueZ –the open source Bluetooth stack. We deploy BlueEar to eavesdrop on a data and audio traffic as in section VIII-B.

Fig. 18 shows significant drop in packet capture rates due to the countermeasure. Fig. 21(a) evaluates PSNR, where the average drops to 15. Further MOS scores drop to poor in 95% of the time, as shown in Fig. 21(b). This confirms that the sniffer experience poor audio quality due to high packet loss rate, which is mainly caused by the countermeasure. Similarly, Fig. 22 evaluates seriousness of eavesdropped data packets, where average seriousness drops to 40% in 95% of the time.

IX. Conclusion

This paper presents BlueEar, the first Bluetooth packet sniffer that only uses inexpensive, Bluetooth-compliant radios. BlueEar features a dual-radio architecture, where two radios are coordinated by a suite of novel algorithms to eavesdrop on an indiscoverable Bluetooth device, relieving the need of expensive specialized radios adopted by commodity sniffers. Extensive experiments show that BlueEar can maintain a high packet capture rate higher than 90% in dynamic settings. We discuss the privacy implications of BlueEar, and propose a practical countermeasure that can reduce the packet capture rate of the sniffer to 20%.

References

[1] W. Albazrqaoe, J. Huang, and G. Xing, "Practical Bluetooth traffic sniffing: Systems and privacy implications," in *Proc. 14th ACM Annu. Int. Conf. Mobile Syst., Appl., Services (MobiSys)*, 2016, pp. 333–345.

[2] *Bluetooth Technology*. Accessed: Aug. 6, 2017. [Online]. Available: https://www.bluetooth.com

[3] *Android Auto*. Accessed: Aug. 6, 2017. [Online]. Available: https://www.android.com/auto

[4] *Apple CarPlay*. Accessed: Aug. 8, 2017. [Online]. Available: http://www.apple.com/ios/carplay

[5] B. Zhang, C. Xu, and D. Feng, "Real time cryptanalysis of Bluetooth encryption with condition masking," in *Advances in Cryptology—CRYPTO*, R. Canetti and J. A. Garay, Eds. Berlin, Germany: Springer, 2013.

[6] B. Zhang, C. Xu, and D. Feng, "Practical cryptanalysis of Bluetooth encryption with condition masking," *J. Cryptol.*, vol. 31, no. 2, pp. 394–433, Apr. 2017, doi: 10.1007/s00145-017-9260-1.

[7] A. K. Das, P. H. Pathak, C.-N. Chuah, and P. Mohapatra, "Uncovering privacy leakage in BLE network traffic of wearable fitness trackers," in *Proc. 17th ACM Int. Workshop Mobile Comput. Syst. Appl. (HotMobile)*, 2016, pp. 99–104.

[8] X. Pan *et al.*, "How privacy leaks from Bluetooth mouse?" in *Proc. ACM Conf. Comput. Commun. Secur. (CCS)*, 2012, pp. 1013–1015.

[9] J. Padgette *et al.*, "Guide to Bluetooth security," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. 800-121 Rev 2, 2017, doi: 10.6028/NIST.SP.800-121r2.

[10] Y. Qu and P. Chan, "Assessing vulnerabilities in Bluetooth low energy (BLE) wireless network based IoT systems," in *Proc. IEEE 2nd Int. Conf. Big Data Secur. Cloud (BigDataSecurity), IEEE Int. Conf. High Perform. Smart Comput. (HPSC), IEEE Int. Conf. Intell. Data Secur. (IDS)*, Apr. 2016, pp. 42–48.

[11] P. Cope, J. Campbell, and T. Hayajneh, "An investigation of Bluetooth security vulnerabilities," in *Proc. IEEE 7th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2017, pp. 1–7.

[12] *Frontline Test Equipments*. Accessed: Feb. 10, 2015. [Online]. Available: http://www.fte.com

[13] *Ubertooth*. Accessed: Jun. 5, 2017. [Online]. Available: http://ubertooth.sourceforge.net

[14] *GNU Radio*. Accessed: Mar. 1, 2015. [Online]. Available: https://gnuradio.org

[15] *Ettus Research*. Accessed: Jan. 11, 2015. [Online]. Available: https://www.ettus.com

[16] D. Spill and A. Bittau, "BlueSniff: Eve meets Alice and Bluetooth," in *Proc. 1st USENIX Workshop Offensive Technol. (WOOT)*. Berkeley, CA, USA: USENIX Association, 2007, pp. 1–10.

[17] M. Ryan, "Bluetooth: With low energy comes low security," in *Proc. 7th USENIX Conf. Offensive Technol. (WOOT)*, Berkeley, CA, USA: USENIX Association, 2013, pp. 1–7.

[18] Adafruit. *Bluefruit LE Sniffer*. Accessed: Jan. 8, 2016. [Online]. Available: https://www.adafruit.com/products/2269

[19] *Sniffer Firmware of CC2540*. Accessed: Jan. 8, 2016. [Online]. Available: https://e2e.ti.com/support/wireless_connectivity/f/538/t/197748

[20] M. Moser. *Busting the Bluetooth Myth—Getting Raw Access*. Accessed: Oct. 12, 2017. [Online]. Available: http://goo.gl/8qNjwM

[21] Logitech. *Logitech Advanced 2.4 GHz Technology*. Accessed: Feb. 10, 2016. [Online]. Available: http://goo.gl/svJ2g9

[22] S. Gollakota, F. Adib, D. Katabi, and S. Seshan, "Clearing the RF smog: Making 802.11n robust to cross-technology interference," in *Proc. ACM SIGCOMM Conf. (SIGCOMM)*, 2011, pp. 170–181.

[23] C.-J. M. Liang, N. B. Priyantha, J. Liu, and A. Terzis, "Surviving Wi-Fi interference in low power Zigbee networks," in *Proc. 8th ACM Conf. Embedded Netw. Sensor Syst.*, 2010, pp. 309–322.

[24] Y. Yubo *et al.*, "ZIMO: Building cross-technology MIMO to harmonize Zigbee smog with WiFi flash without intervention," in *Proc. ACM 19th Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, 2013, pp. 465–476.

[25] J. Huang, G. Xing, G. Zhou, and R. Zhou, "Beyond co-existence: Exploiting WiFi white space for Zigbee performance assurance," in *Proc. 18th IEEE Int. Conf. Netw. Protocols (ICNP)*, Oct. 2010, pp. 305–314.

[26] M. Song, S. Shetty, and D. Gopalpet, "Coexistence of IEEE 802.11b and Bluetooth: An integrated performance analysis," *Mobile Netw. Appl.*, vol. 12, nos. 5–6, pp. 450–459, Dec. 2007.

[27] A. Cidon, K. Nagaraj, S. Katti, and P. Viswanath, "Flashback: Decoupled lightweight wireless control," in *Proc. ACM SIGCOMM Conf. Appl., Technol., Architectures, Protocols Comput. Commun. (SIGCOMM)*, 2012, pp. 223–234.

[28] J. Huang, W. Albazrqaoe, and G. Xing, "BlueID: A practical system for Bluetooth device identification," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2014, pp. 2849–2857.

[29] T. Joachims, "Making large-scale SVM learning practical," in *Advances in Kernel Methods: Support Vector Learning*, B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds. Cambridge, MA, USA: MIT Press, 1999.

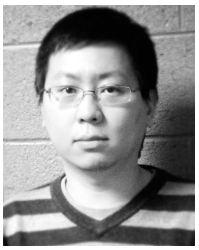
[30] N. Cheng, X. O. Wang, W. Cheng, P. Mohapatra, and A. Seneviratne, "Characterizing privacy leakage of public WiFi networks for users on travel," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 2769–2777.

[31] A.-N. Moldovan, I. Ghergulescu, and C. H. Muntean, "A novel methodology for mapping objective video quality metrics to the subjective MOS scale," in *Proc. IEEE Int. Symp. Broadband Multimedia Syst. Broadcast. (BMSB)*, Jun. 2014, pp. 1–7.

Wahhab Albazrqaoe received the B.S. degree in computer and software engineering from the University of Technology, Baghdad, Iraq, in 2002, and the M.S. and Ph.D. degrees from the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, USA, in 2011 and 2018, respectively.

He is currently with the College of Engineering, University of Kufa, Najaf, Iraq.

His research interests include wireless networks, mobile systems wireless security/privacy, and autonomous vehicles networks.



Jun Huang received the B.S. and M.S. degrees in computer science from Beihang University, China, in 2005 and 2008, respectively, and the Ph.D. degree in computer science and engineering from Michigan State University in 2013. He is currently an Assistant Professor with the Center for Energy Efficient Computing and Applications, School of EEC, Peking University. His research interests include wireless nets and mobile systems. He received the Best Paper Awards at the 18th IEEE International Conference on Network Protocols in 2010.

Guoliang Xing received the B.S. degree in electrical engineering and the M.S. degree in computer science from Xi'an Jiaotong University, China, in 1998 and 2001, respectively, and the M.S. and D.Sc. degrees in computer science and engineering from Washington University in St. Louis in 2003 and 2006, respectively.

He is currently a Professor with the Department of Information Engineering, The Chinese University of Hong Kong. Previously, he was a Faculty Member at the City University of Hong Kong and Michigan State University, East Lansing, MI, USA.

His research lies at the intersection between systems, embedded AI, data/information processing algorithms, and domain sciences, with a focus on interdisciplinary applications in health, environment, and energy.

Dr. Xing received the Faculty Early Career Development (CAREER) Award from the US National Science Foundation in 2010 and the Withrow Distinguished Scholar Award from the Michigan State University in 2014. His group received two Best Paper Awards and five Best Paper Nominations from the prestigious international conferences, including ICNP, IPSN, and PerCom.