SecurityHub: Electromagnetic Fingerprinting USB Peripherals using Backscatter-assisted Commodity Hardware

Si Liao
SIST, ShanghaiTech University
liaosi@shanghaitech.edu.cn

Huangxun Chen
IoT Thrust, Information Hub, HKUST (GZ)
huangxunchen@hkust-gz.edu.cn

Zhice Yang
SIST, ShanghaiTech University
yangzhc@shanghaitech.edu.cn

***Abstract*—In this paper, we propose an innovative method for fingerprinting USB peripherals. While USB technology has made significant progress in data transfer efficiency, by default, the USB host device trusts any connected peripherals. This ignorance has led to several security risks in practice. Existing protection practices mitigate these risks by verifying the specific type or even the identity of the peripheral before data transfer. However, these methods often depend on expensive and specialized hardware, such as software-defined radios, limiting their applicability in everyday scenarios.**

To address this issue, we propose profiling the electromagnetic radiation (EMR) generated by a USB peripheral as its unique identifier. Our method utilizes a low-cost backscatter unit and the host's WiFi network card for the profiling and verification process. The backscatter unit is integrated into a USB hub. It shifts the USB EMR signal into the frequency domain within the WiFi sensing range, and transforms EMR into an interpretable form suitable for feature extraction. Subsequently, we employ a hybrid classification method, combining heuristic and neural network clues, to recognize the identity of the EMR's source. We evaluate the effectiveness of our method on an extensive dataset collected from USB peripherals of various types and vendors.

1. Introduction

Universal Serial Bus (USB) serve as the predominant interface for linking diverse electronic peripherals, *e.g.*, keyboards, portable hard drives, network interface cards, to host computers. However, it lacks native security primitives to verify the identity of the connected peripheral. Consequently, malicious USB peripherals can be engineered to spoof arbitrary device information such as device ID and type. Once such peripherals happen to be connected to the host, it can be used for, *e.g.*, covertly stealing confidential data [30], [33], monitoring activities [40], compromising the host system [26], or even permanently cause physical damages to the host hardware [12].

Identifying USB peripheral information (device type, manufacturer, and specific characteristics) is considered an effective way to mitigate the aforementioned security risks.

This work was done while Si Liao was a visiting student at HKUST (GZ).

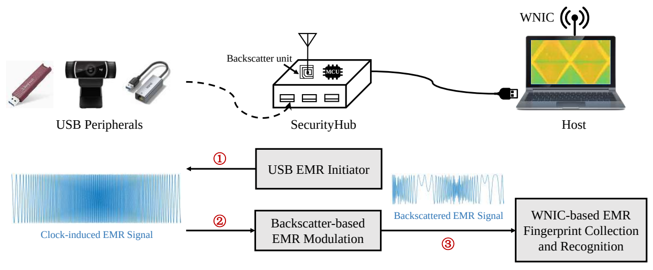


Figure 1: Workflow of SecurityHub.

Recent research has explored several methods. One approach is to add verifiable features to USB peripherals, such as DeviceVeil [44], which utilizes Physical Unclonable Functions (PUF) on integrated circuits (IC) and implements a hypervisor for pre-OS authentication. However, this approach necessitates additional IC costs on *every* USB peripheral.

Another branch of methods is to profile the physical characteristics of USB peripherals to create hardware fingerprints. For example, Time-Print [22] uses the access latency to distinguish USB peripherals, but it is limited to flash storage devices. More general methods rely on physical signals like electrical [42] and magnetic signals [31] to create hardware fingerprints. However, these methods usually require very fine-grained signals for feature extraction, necessitating high-speed analog-to-digital converters (ADC) or radio front ends for digitization. Some works use oscilloscopes costing thousands of dollars, while others employ low-end software-defined radios (SDR) priced at a few hundred dollars. Yet, they are still too expensive for consumer products. The costs are not solely due to the hardware but also the computational resources required on the host machine to process the intensive SDR data streams.

The above reasons limit the widespread deployment of existing methods in practical settings. This naturally raises the question: *Is it possible to implement USB peripheral fingerprinting using low-cost, widely accessible hardware?* We give a positive answer by proposing SecurityHub.

Our design leverages the widespread commercial WiFi network interface cards (WNIC) in host computer to detect the electromagnetic radiation (EMR) emitted by the USB peripheral for identification/authentication. Modern WNICs rely on processing radio signals from the frequency perspective. This capability also agilely allows WNICs to capture and monitor arbitrary signals, including USB EMR, in the frequency domain.

However, implementing this idea faces three challenges. Firstly, there is a frequency mismatch issue: WiFi bands typically operate around 2.4 GHz, whereas the major source of device EMR, the USB data clock, is around 2.5 GHz. This discrepancy largely limits the EMR information that can be captured. Secondly, USB devices use a spread-spectrum clocking (SSC) technique to reduce its EMR interference. This technique rapidly alters the frequencies the EMR signals and blurs the measurable information. Thirdly, while prior research has demonstrated the feasibility of utilizing SDRs or high-end oscilloscopes to capture device EMR as fingerprints, it remains unclear whether the information captured by commercial WNICs, which have significantly lower time and frequency resolutions, can be similarly applied.

To overcome the aforementioned challenges, we propose the SecurityHub system with a workflow depicted in Figure 1. As a USB hub, the SecurityHub device expands USB ports and connects peripherals to the host. Additionally, it renders peripheral identification to the host.

The core idea of SecurityHub is to incorporate a low-cost backscatter unit to modulate USB EMR signals that enables the host to capture and analyze them with high precision and cost-effectiveness. The backscatter unit is integrated into the SecurityHub device and can reflect the USB EMR signals emitted by the connected peripheral into the surrounding environment. According to existing research [34], [47], backscatter operations can impose a frequency shift on the incident signals. This property allows us to:

- Mirror a copy of USB EMR signals to the WiFi band. This enables the host computer to capture them directly with a commercial WNIC.
- Demodulate USB EMR signals in the air using a modulation method opposite to the USB SSC. This slows down the USB EMR pattern containing features, ensuring that signals captured at low rates by the WNIC still exhibits good distinguishability.

Additionally, by analyzing the EMR signals collected using the aforementioned method, we find that USB peripherals of the same model exhibit slight differences in their EMR signals. This is mainly because: 1. slight physical difference of on-board oscillators result in frequency offsets in various EMR attributes; 2. subtle implementation differences in analog SSC circuits lead to nuanced variations in the SSC patterns. Based on these findings, we design a frequency feature-based classifiers on the host to identify and verify connected peripherals. Only after confirming the identity of a connected peripheral will SecurityHub permit data transfers.

In summary, our contributions are as follows:

- To the best of our knowledge, we are the first to implement electromagnetic fingerprinting for general USB peripherals using low-cost and widely accessible hardware. Specifically, our system primarily utilizes standard WiFi network cards to capture the distinctive EMR signals produced by USB clock signals, thereby eliminating the need for specialized and expensive signal collection and processing equipment.
- We design a backscatter-based scheme that enables cost-effective USB EMR profiling. This scheme allows backscattered EMR signals to fall within the WNIC's spectral monitoring range and reveals the inherent EMR features even at a low sampling rate.
- We implement the SecurityHub device along with a classifier on the host. We evaluate the effectiveness of the system using an extensive dataset consisting of 34 USB peripherals (4 types and 17 distinct brands). The results show over 97% accuracy in distinguishing different types and over 84% accuracy for same-type peripherals from different brands, demonstrating its capability to identify unauthorized peripherals.

2. Threat Model

We assume the attacker intends to use malicious USB peripherals to launch attacks, such as data theft [40], [38], [33], system compromise [26], [48], or Trojan injection [20], against the user's (victim's) host system. Our security goal is to help the host system is to defend against malicious USB peripherals.

We assume that the attacker has the capability to modify or even craft any type of USB peripheral. We refer to its produced USB peripherals as malicious USB peripherals. They are assumed to be capable of executing arbitrary local code and generating arbitrary digital logic signals on the USB cable at the attacker's will.

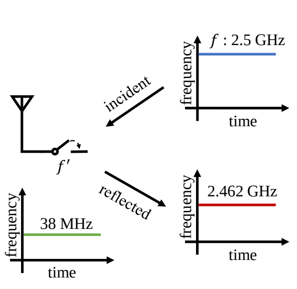
We assume that the attacker needs to plug a malicious USB peripheral into the host's USB port to launch the attack. They can disguise the malicious peripheral as a device familiar to the user, such as a USB flash drive, power bank, or keyboard, to induce the user to connect it. Alternatively, the attacker might connect it themselves. However, due to contextual constraints, such as monitored workspaces and server rooms, we assume that the attacker has only a limited amount of time and capability when in close proximity to the host. For example, they cannot physically dismantle the case to disable the host's security hardware or other functions.

3. Preliminaries and Design Rationale

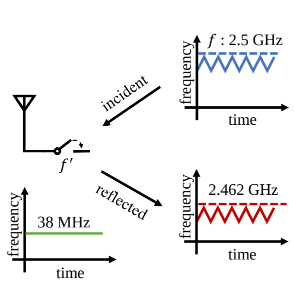
In this section, we introduce the basics of USB and backscatter technology. Then, we illustrate the design rationale of SecurityHub.

3.1. USB Technology

USB is a widely-used protocol for wired data and power transfer between peripheral devices and a host. USB 3.x represents the current mainstream USB version. It defines three physical layers (PHY): Gen1, Gen2, and Dual-Lane Operation. This paper is based on USB 3.2 Gen1, as newer versions are backward compatible. For simplicity, we use USB 3.0 to refer to USB 3.2 Gen1.



(a) fixed f, fixed f'



(b) varying $f(t)$, fixed f'

Figure 2: Frequency-shift Effect of Backscatter Operations. Opening and shorting the antenna circuit at frequency f' can frequency shift the incident signal to the reflected copies by a shift value of $\pm f'$.

When a USB peripheral is connected to a USB port, multiple wires within the USB cable establish connections to the host to facilitate data transfer, as illustrated in left part of Figure 4. The cable contains wires for power supply (VBUS and GND), USB 2.0 data transmission/reception (USB 2.0 TX/RX), and USB 3.0 data transmission/reception (USB 3.0 SSTX/SSRX). During data transfer, the USB 3.0 PHY converts digital bits into binary voltage levels (high and low) at each clock edge to drive the signals in the transmission (Tx) lines. Given the raw data rate of 5 Gbps, the corresponding electromagnetic radiation (EMR) is near 2.5 GHz.

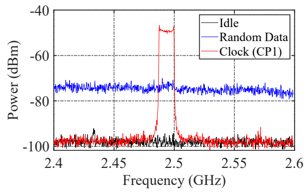
To minimize EMR interference with host wireless devices, USB 3.0 employs spread spectrum clocking (SSC). SSC rapidly sweeps the data clock frequency to spread the spectrum of the Tx signal to reduce its peak spectral power.

3.2. Backscatter Technology

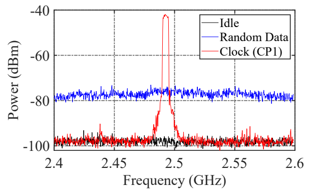
The operating principle of backscatter communication is depicted in Figure 2(a). A backscatter transmitter is simply an antenna of controllable impedance. It reflects an incident radio signal to transmit information, instead of generating radio signals on its own. This technology has been extensively discussed in the area of low-power wireless communications [15], [34], [25].

A backscatter transmitter circuit can be realized by switching the impedance of an antenna between two distinct states. Altering the antenna's impedance leads to a corresponding change in the amplitude of the reflected signal across these states. To formalize, assuming the frequency of impedance transition is f' and the incident signal is denoted as $w_{\text{in}}(t) = \sin(2\pi ft)$, The backscatter operation is like modulating $w_{\text{in}}(t)$ with a square wave of period $1/f'$, and if we approximate this square wave with its first harmonic component, the resulting signal can be expressed as:

$$\begin{aligned} w_{\text{in}}(t) \cdot w_{\text{switch}}(t) &= \alpha \cdot \sin(2\pi ft) \cdot \text{square}_{f'}(t) \\ &\approx \alpha \cdot \sin(2\pi ft) \cdot \sin(2\pi f't) \\ &= \frac{\alpha}{2} \cdot \cos(2\pi(f+f')t) + \frac{\alpha}{2} \cdot \cos(2\pi(f-f')t) \\ &= w_{\text{up}}(t) + w_{\text{down}}(t), \end{aligned} \tag{1}$$



(a) Portable Solid State Drive (SSD), Kingston



(b) Portable Hard Disk Drive (HDD), Western Digital

Figure 3: EMR Spectrum of Different USB Devices.

where α represents the reflection loss. Equation 1 can be interpreted as: the incident signal, upon modulated by the backcatter transmitter, gives rise to two backscattered mirrors characterized by their shifted frequency:

A backscatter transmitter switching at frequency f' introduces a consistent shift $\pm f'$ to the incident signal's frequency, i.e., $f(t) + f'$ and $f(t) - f'$.

Figure 2(a) visually explains the above equation by showing the $w_{\text{down}}(t)$ part.

3.3. Design Rationale: Backscatter-assisted USB EMR Fingerprinting

The feasibility of utilizing EMR as device fingerprints has been validated by existing studies [41], [31]. However, due to the necessity for specialized equipment like SDRs for measurements, both cost and accessibility pose challenges for widespread adoption. Consequently, we are exploring alternative solutions. An intriguing fact is that the USB 3.0 data clock frequency is 2.5 GHz (§3.1), which is close to the 2.4 GHz WiFi operating bands. This suggests the possibility to combine two accessible technologies for our purposes.

To validate this possibility, in Figure 3, we employ a spectrum analyzer to measure the power spectrum around USB 3.0 devices during data transmission. The results indicate noticeable EMR energy increases around 2.5 GHz. Specifically, when the transmitted data alternates between "01"s, *i.e.*, a clock signal, a distinct peak occurs at 2.5 GHz. An important observation is that when comparing experimental results among different peripherals, subtle differences can be found in their EMR peaks, *e.g.*, see the clock peaks of Figure 3(a) and (b).

Further investigation suggests the subtle electromagnetic variances are inherent and distinctive. They stem from the differences in the manufacturing process and physical attributes of the device clock oscillators [41], [31]. This motivates our approach to utilize the EMR signals related to the clock oscillator for device fingerprinting.

The following question is how to employ readily available commercial hardware, such as WNICs, to reveal such differences. WiFi operates at 2.4 GHz, whereas the EMR signals center around 2.5 GHz. To facilitate the WNIC's capability to capture EMR signals and extract detailed features, our solution incorporates an additional, cost-effective backscatter module. This module fulfills one critical role: it assists in transposing the frequency of the clock-induced USB EMR signal into the WiFi band.

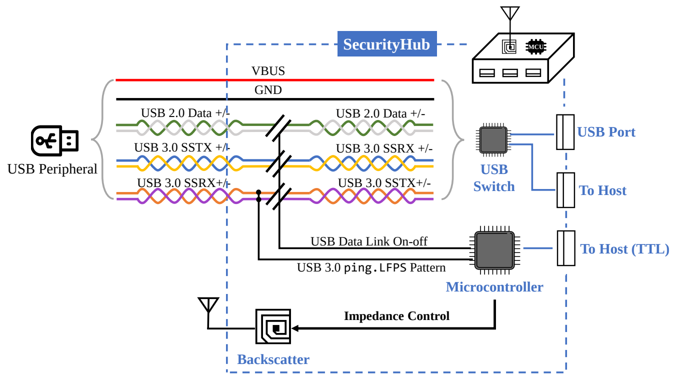


Figure 4: Internal Structure of SecurityHub.

The method is based on the frequency shift property shown in Figure 2. We employ the backscatter module to downshift the 2.5 GHz USB EMR signal to the 2.4 GHz WiFi band, which allows us to use the WNIC's spectrum scan function to analyze the frequency components of the USB EMR signals [9], [18].

However, the above frequency shift is too coarse-grained. As illustrated by $f(t)$ in Figure 2(b), the EMR from USB is also influenced by SSC modulation. The rapid frequency sweep induced by SSC result in intensive blurring of the collected information. In the following section (§4.1), we will introduce the methods for generating USB clock signals. Subsequently, we will discuss approaches to counteract SSC modulation (§4.2) and extract features from the collected data (§4.3).

4. SecurityHub System Design

Figure 1 presents an overview of the SecurityHub system. It consists of two parts. The first part is the *SecurityHub device*, an enhanced USB Hub. In addition to expanding the number of USB ports, the SecurityHub device incorporates two modifications to support the proposed security features. Firstly, upon peripheral insertion, it blocks the actual data transmission and initializes the peripheral's clock-like "01" Tx data signals until the access is guaranteed (§4.1). Secondly, it integrates a cost-effective backscatter unit to reflect the USB EMR signals to assist signal capturing (§4.2).

The second part is the *SecurityHub host application* (§4.3). It instructs the host's WNIC to capture the backscattered EMR signals and extracts features from them as the fingerprint of the inserted USB peripheral. The legitimacy of the peripheral can be determined by comparing its fingerprint with the enrolled fingerprint database. The detailed design of each module will be elaborated in the subsequent sections.

4.1. USB EMR Initiator

The primary role of this module is to stimulate the USB EMR signal for fingerprinting purposes prior to actual data transmission. As discussed in §3.1, USB peripherals emit EMR signals during data transfer. However, for our scenario, it is essential to verify peripheral identity prior to any data exchange. Therefore, we explore the USB protocol and identify a specific mode suitable for SecurityHub, namely, the *Compliance Mode*, as defined by the USB 3.x specification [11].

Pattern	Description	Frequency range
CP0	A pseudo-random data pattern	-
CP1	Nyquist frequency	2.5 GHz
CP2	Nyquist/2	1.25 GHz
CP4	The low frequency periodic signaling pattern	20-100 MHz
CP8	Repeating 50-250 1's and then 50-250 0's	10-50 MHz

TABLE 1: USB 3.x Compliance Pattern Examples

Compliance Mode (CP) is designed to meet the testing requirements of USB 3.x devices, and it is mandatory for both USB peripherals and hosts. When a USB peripheral is powered up, it will first issues a `polling` signal to probe the connection. Without receiving a corresponding response for a period, it automatically transitions into the compliance mode. In this mode, its PHY emits one of the predefined pattern sequences (CP0-CP16). The transmitter switches to the next CP patterns in a round-robin sequence upon receiving a Low Frequency Periodic Signaling (LFPS) signal: `ping.LFPS`.

From Table 1, CP patterns have two desired features. Firstly, they can be transmitted without entering the actual data transmission mode, which aligns with our security design objectives. Secondly, unlike typical USB data messages, the transmission of CP patterns is continuous, and certain CP patterns contain no data randomness, exhibiting favorable measurable properties. For example, CP1, consisting of a "01" sequence, directly reflects the USB data clock.

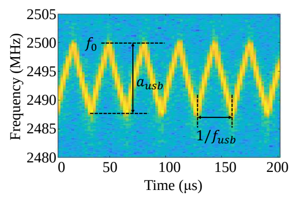
SecurityHub utilizes this mode to accomplish its objectives. As shown in Figure 4, the SecurityHub device initiates the connected USB peripheral to generate CP patterns with the follows steps:

- It disconnects the data links of the connected USB peripheral by default. This isolates the peripheral to the host and also forces the peripheral entering the compliance mode.
- To command the USB peripheral to transmit a specific CP pattern, it stores an `ping.LSPF` pattern and replays it to the SSRX data cables[1].

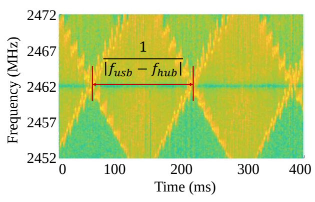
The above procedures can stimulate the USB peripheral to transmit a 2.5 GHz "01" sequence on the TX lines. Then, the ideal EMR signal is a square wave. Based on its Fourier series expansion, its power predominantly resides in its first harmonic component, *i.e.*, a single-tone wave at 2.5 GHz. This is the clock EMR we observed in Figure 3.

The SecurityHub device collaborates with the host application through a control channel. Upon successful verification of the peripheral's EMR signals by the host application, the hub will physically bridge the data lines and transition the peripheral to the normal mode to start data transfer with the host.

1. LFPS patterns are PHY messages consisting of a sequence of pulse bursts, characterized by the distinct electrical and temporal parameters [11], *e.g.*, signal period, burst duration, *etc.*, we use two GPIOs of the microcontroller to generate the pattern and use resisters to tune the voltage.



(a) Original USB EMR



(b) Backscattered EMR

Figure 5: Comparison of the Time-frequency Spectrograms of the Original (captured with spectrum analyzer) and Flattened Backscattered (captured with WNIC) USB EMR Signals.

4.2. Backscatter-based EMR Modulation

While the clock EMR obtained by the previous subsection already exhibits stable and prominent energy, its frequency is still rapidly varying due to SSC. To allow effective capturing by the WNICs. The SecurityHub device employs a backscatter unit to precisely manipulate and stabilize the EMR frequency.

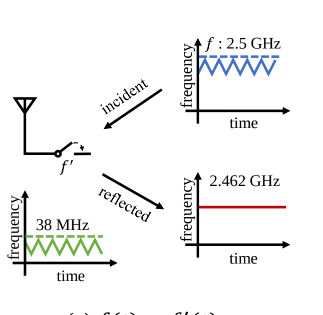
4.2.1. Spread Spectrum Clocking. In accordance with the specifications, all the USB devices have to implement the spread spectrum clocking (SSC) in their transceivers. SSC's primary purpose is to diminish the peak power of EMR by dispersing its spectrum across a broad frequency range within a brief timeframe.

To visualize the effect of SSC, we use an spectrum analyzer to record the USB EMR spectrum over a period of time. As illustrated in Figure 5(a), the base data clock is at f_0=2.5 GHz. Consistent with the specification [11], SSC introduces a linear frequency changing of up to $2a_{\text{usb}} = 0.5\% f_0 = 12.5$ MHz, with a changing frequency of $f_{\text{usb}} = 31.5$ kHz. Specifically, SSC modulates the frequency of EMR signal to oscillate between f_0 and $f_0 - 2a_{\text{usb}}$, with a periodicity of $1/f_{\text{usb}}$. Therefore, at any given time t, the instantaneous frequency $F_{\text{USB}}(t)$ of the EMR signal can be modeled as:

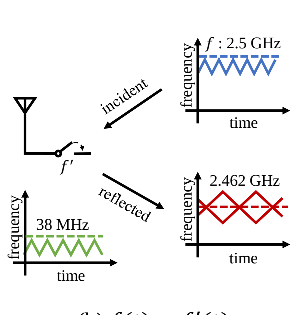
$$F_{\text{USB}}(t) = f_0 + a_{\text{usb}} \left(\text{tri}\left(2\pi f_{\text{usb}} t + \phi_{\text{usb}}\right) - 1\right), \qquad (2)$$

where ϕ_{usb} denotes the initial offset phase, tri$(\cdot)$ denotes the canonical triangular wave with a unit amplitude and a period of 2π.

One consequence of SSC's high-frequency modulation is that it makes the WNIC's low-rate sampling measurements vague and indistinguishable. For measuring environmental wireless signals, WNICs rely on the spectral scan function. It uses Fast Fourier Transform (FFT) to quantify the signal power across frequency bins [14]. The scan output is a time series that includes timestamps and the signal power of the frequency bins. By stacking these results over time, we can form a spectrogram similar to that produced by a spectrum analyzer. The spectral scan's frequency resolution is approximately 80 kHz, derived from a 256-point FFT across a 20 MHz bandwidth. At the same time, the time resolution is approximately around 4 kHz. In contrast, when



(a) $f(t) = f'(t)$



(b) $f(t) \approx f'(t)$

Figure 6: Reducing the Impact of SSC in the Backscattered EMR Signal. This can be done by applying a varying, instead of constant, frequency shift $f'(t)$, identical (a) or similar (b) to the EMR signal's varying frequency $f(t)$.

a USB peripheral employs SSC, its clock-induced EMR experiences frequency oscillation, around a rate of 31.5 kHz. This indicates that the time resolution of the WNIC is insufficient to capture the frequency variations of the EMR signal, potentially causing it to miss fine-grained features.

4.2.2. Flattening EMR through Backscatter Modulation. Given the consistent periodic nature of the SSC modulation, in addition to shifting a fixed frequency (§3.3 and Figure 2(b)), we incorporate additional backscatter modulation to ”flatten” the EMR signal's frequency oscillation. This allows WNICs to capture subtle EMR features directly. The concept is explained in Figure 6, by closely mimicking the behavior of SSC and introducing a similar oscillation into the backscatter's switching frequency, the effect of SSC can be canceled out (a) or largely reduced (b).

Formally, recall the EMR signal's frequency is $F_{\text{USB}}(t)$ in Equation 2. If we apply an inverted tri$(\cdot)$ part to shift $F_{\text{USB}}(t)$, it will become a constant, *e.g.*, the center frequency F_{ch} of a WiFi band:

$$F_{\text{USB}}(t) - F_{\text{HUB}}(t) = F_{\text{USB}}(t) - (F_{\text{USB}}(t) - F_{\text{ch}}) = F_{\text{ch}},$$

where $F_{\text{HUB}}(t)$ is the time-varying frequency shift induced by backscatter operations. Figure 6(a) depicts the above situation.

However, in reality, we cannot generate two entirely identical signals (which is unnecessary, as long as the rate of EMR frequency variation slows down, the WNIC can capture sufficiently rich information). When we apply a tri$(\cdot)$ with a frequency f_{hub}, slightly different from f_{usb},

$$F_{\text{HUB}}(t) = f_0 - F_{\text{ch}} + a_{\text{hub}} \left(\text{tri}\left(2\pi f_{\text{hub}} t + \phi_{\text{hub}}\right) - 1\right), \quad (3)$$

the frequency of the actual reflected EMR signal will still vary, but at a much slower rate. $F_{\text{USB}}(t) - F_{\text{HUB}}(t)$ becomes:

$$F_{\text{USB}}(t) - F_{\text{HUB}}(t) = \underbrace{F_{\text{ch}} + a_{\text{hub}} - a_{\text{usb}}}_{\text{time-invariant}} + \underbrace{a_{\text{usb}}\text{tri}\left(2\pi f_{\text{usb}} t + \phi_{\text{usb}}\right) - a_{\text{hub}}\text{tri}\left(2\pi f_{\text{hub}} t + \phi_{\text{hub}}\right)}_{\text{enveloped by a triangular wave of frequency}\,|f_{\text{hub}} - f_{\text{usb}}|}. \qquad (4)$$

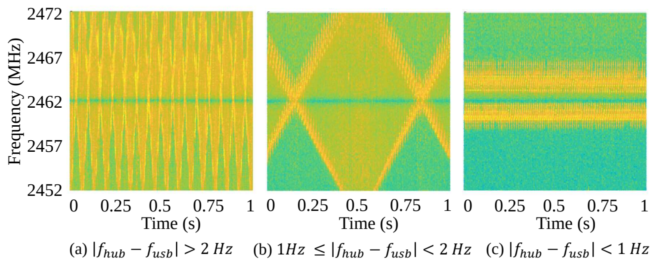


Figure 7: Backscattered EMR Signals with Different Backscatter Parameters.

In the above equation, the time-invariant components contribute to a constant frequency offset of $F_{\text{ch}} + a_{\text{hub}} - a_{\text{usb}}$. The time-varying components in the latter part is the superposition of two triangular waves with similar frequencies. If we use sine waves to approximate the triangular waves, it can be deduced that the envelope of the superimposed result is a triangular wave with a frequency of $|f_{\text{hub}} - f_{\text{usb}}|$. Figure 6(b) illustrates the above analysis.

The above implies that the frequency of the backscattered EMR signal changes to fluctuate at a rate of $|f_{\text{usb}} - f_{\text{hub}}|$ instead of f_{usb}. Note that $|f_{\text{usb}} - f_{\text{hub}}|$ is controllable by tuning f_{hub}, and could be much slower than the sampling rate of the WNIC. After applying such backscatter operations with an appropriate f_{hub}, the WNIC-captured spectrogram in Figure 5(b) exhibits intriguing features.

4.3. WNIC-based EMR Fingerprint Collection and Identification

The backscatter unit enables the host to effectively capture the SSC-modulated EMR signal using the WNIC. In this subsection, we address two remaining questions: how to collect EMR samples with sufficient distinctiveness and how to extract features to differentiate them. These tasks are primarily carried out by the SecurityHub host application.

4.3.1. Distinctive Fingerprint Searching. Equation 4 implies that the discrepancy between f_{usb} and f_{hub} governs the width of a single diamond envelope depicted in Figure 5(b). Note that the f_{usb} values for different USB peripherals vary, typically ranging from 30 kHz to 33 kHz. In an experimental setup involving three distinct USB peripherals with the same f_{hub} setting, the resultant backscattered EMR signals are displayed in Figure 7. Figure 7(a) showcases a scenario where the SSC frequency delta $|f_{\text{usb}} - f_{\text{hub}}|$ is approximately 15 Hz, resulting in about 15 complete diamond patterns in one second. Conversely, Figure 7(c) illustrates a situation where the SSC frequency difference $|f_{\text{usb}} - f_{\text{hub}}|$ is substantially below 1 Hz, such that a 1-second timeframe fails to encompass a full diamond pattern. Figure 7(b) represents an intermediate case between (a) and (c), with a frequency difference ranging from 1 Hz to 2 Hz, thereby including a full diamond pattern.

This experimental observation gives us two important insights. Firstly, the presence of a single complete diamond in the collected spectrogram suggests that the configured f_{hub} of the backscatter module, is within a 1 Hz - 2 Hz offset from the SSC frequency, f_{usb}, of the EMR signals. Secondly, the occurrence signifies an optimal condition for amplifying the intricate characteristics of the USB EMR signal, thereby enhancing the efficacy of device identification. The presence of either an excessive number of diamonds or an incomplete one within a specific timeframe does not meet our requirement for accurate fingerprinting.

Algorithm 1 Sweeping-based Frequency Searching

```
Input: $f_{\min}$, $f_{\max}$, $f_{\text{fast\_step}}$, $t_{\text{fast\_scan}}$, $f_{\text{step}}$, $t_{\text{scan}}$
Output: EMR Fingerprint $S_{\text{EMR}}$, $f_{\text{hub}}$
 1: $v_{\text{queue}} = []$
 2: for $f_{\text{hub}}$ in range($f_{\min}$, $f_{\max}$, $f_{\text{fast\_step}}$) do
 3:     $S_{\text{EMR}} \leftarrow$ WNIC_scanning($t_{\text{fast\_scan}}, f_{\text{hub}}$)
 4:     $t_{\text{cycle}}, Err_{\text{fit}} \leftarrow$ cycle_calculation($S_{\text{EMR}}, t_{\text{fast\_scan}}/2$)
 5:     $v_{\text{queue}}$.append([$f_{\text{hub}}$, $t_{\text{cycle}}$, $Err_{\text{fit}}$])
 6: end for
 7: $f_{\text{hub}}, t_{\text{cycle}}$ = get_min_err($v_{\text{queue}}$)
 8: $f_{\text{usb\_candi}} = [f_{\text{hub}} + 1/t_{\text{cycle}}, f_{\text{hub}} - 1/t_{\text{cycle}}]$
 9: for $f_{\text{hub}}$ in round($f_{\text{usb\_candi}} + 1/t_{\text{scan}}, f_{\text{step}}$) do
10:     $S_{\text{EMR}} \leftarrow$ WNIC_scanning($t_{\text{scan}}, f_{\text{hub}}$)
11:     $t_{\text{cycle}}, Err_{\text{fit}} \leftarrow$ cycle_calculation($S_{\text{EMR}}, t_{\text{scan}}/2$)
12:     if $1/t_{\text{scan}} \le 1/t_{\text{cycle}} < 4/t_{\text{scan}}$ then
13:         return $S_{\text{EMR}}$, $f_{\text{hub}}$
14:     end if
15: end for
```

To identify the optimal frequency for f_{hub} to capture the desired EMR fingerprints as similar to Figure 7(b), we develop a sweeping-based frequency searching algorithm outlined in Algorithm 1. The algorithm's frequency search space for f_{hub} is described by $f_{\min}$ and $f_{\max}$, set to 30 kHz and 33 kHz, respectively. Our basic strategy involves executing a fast scanning phase to approximately determine f_{usb} (Line 2 to Line 7 in Algorithm 1), followed by a detailed scanning phase to refine the result (Line 8 to Line 15 in Algorithm 1).

During the fast scanning phase, f_{hub} sweeps from $f_{\min}$ to $f_{\max}$ in step of $f_{\text{fast\_step}} = 100$ Hz (Line 2). For each f_{hub}, the `WNIC_scanning` function is invoked for EMR measurements over a brief duration, $t_{\text{fast\_scan}} = 10$ ms (Line 3). Subsequently, the captured signal S_{EMR} is processed through the `cycle_calculation` function to estimate its cycle period of frequency variation (Line 4).

The `cycle_calculation()` function is detailed in Algorithm 2. This function is pivotal for analyzing the time-frequency diagram of the backscattered EMR signals and determining the cycle period of the diamond patterns. The essence of cycle calculation involves approximating the variation trend of the major frequency components, *i.e.*, the diamond pattern, in the EMR signal S_{EMR} with two symmetrical triangular waveforms characterized by their cycle t_{cycle} (Line 8 in Algorithm 2). Nevertheless, the optimization solver `fminsearch` [4] is notably influenced by the choice of initial parameters. To mitigate this, a self-correlation

Algorithm 2 `cycle_calculation`

```
Input: $S_{\text{EMR}}$, $t_{\text{template}}$
Output: $t_{\text{cycle}}$, $Err_{\text{fit}}$
1: $S_{\text{template}} \leftarrow S_{\text{EMR}}[0 : t_{\text{template}}]$
2: for $i$ in len($S_{\text{EMR}}$) $- t_{\text{template}}$ do
3:     corr($i$) $\leftarrow$ sum($S_{\text{EMR}}(i : i + t_{\text{template}}) \cdot S_{\text{template}}$)
4: end for
5: peakIndexArr $\leftarrow$ find(islocalmax(corr)==1)
6: $t_{\text{cycle}} \leftarrow f_{\text{WNIC}}$/average(diff(peakIndexArr))
7: $S_{\text{tri}} \leftarrow$ triangular_wave_generator($t_{\text{cycle}}/2$))
8: $t'_{\text{cycle}}, Err_{\text{fit}} \leftarrow$ fminsearch(distance($S_{\text{tri}}, S_{\text{EMR}}$))
9: return $t'_{\text{cycle}}$, $Err_{\text{fit}}$
```

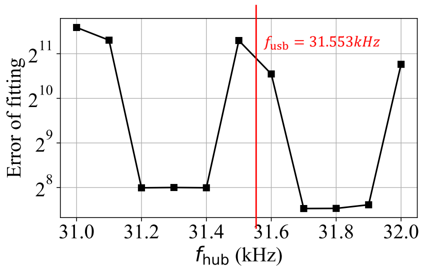


Figure 8: A Demonstration Example Shows the Variation of the Fitting Error Err_{fit} across Multiple f_{hub} Values during the Fast Scanning Phase in Algorithm 1.

step is performed on S_{EMR} to identify possible periodicities (Line 1-4 in Algorithm 2). The correlation window is on purpose set to half of the WNIC scanning duration (Line 4) to ensure reliable assertion confidence. The local peaks of self-correlation are considered as a preliminary estimate of t_{cycle}. This estimate then serves as the initial value for `fminsearch` to derive a refined estimation of t_{cycle}.

Figure 8 plots the fitting error Err_{fit} with respect to possible f_{hub} choices. In this particular instance, the actual f_{usb} is 31.553 kHz, so the f_{hub} = 31.7 kHz attempt yields the lowest fitting error. Their difference is determined by $1/t_{\text{cycle}}$. The figure also shows that when their difference is smaller, the fitting error Err_{fit} actually becomes larger. This is because, in such cases, there is no complete cycle of the diamond pattern in S_{EMR}. By compensating for $1/t_{\text{cycle}}$ in f_{hub}, an estimate of f_{usb} can be obtained, *i.e.*, $f_{\text{usb\_candi}}$, with an error level of a few Hz.

Upon completion of the fast scanning phase, the next step is the detailed scanning. This phase determines the final f_{hub} used for capturing EMR fingerprints. There are two main considerations: First, since the estimated f_{usb} contains a certain degree of randomness, to ensure that fingerprint samples from different capturing attempts are comparable, we round f_{usb} to coarser frequency points in step of f_{step}=10 Hz (Line 9 in Algorithm 1). This way, f_{hub} obtained from different measurements will be consistent. Second, the fingerprint sample needs to include at least one diamond pattern, but not too many (we set the upper limit to 4, Line 12 in Algorithm 1), so f_{hub} is shifted by one sampling period t_{scan}=0.3 s from f_{usb} (Line 9 in

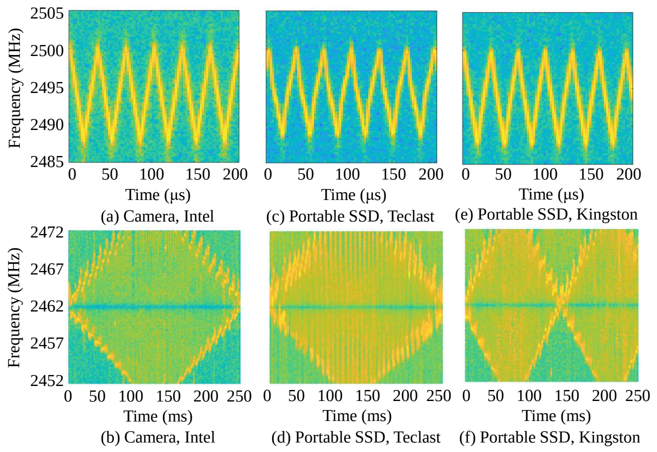


Figure 9: Spectrgrams of USB EMR (up, captured with spectrum analyzer) and the Backscattered EMR (down, captured with WNIC) from Different USB Peripherals.

Algorithm 1)[2]. Then, the algorithm will return both the obtained f_{hub}, a intentionally-shifted and rounded estimation of f_{usb}, and the EMR signals, S_{EMR}, collected at this f_{hub} with a duration of t_{scan} as the fingerprint sample for this USB peripheral.

The above computation is performed in the host computer. The host adjusts f_{hub} through the control channel with the hub. The execution time for `cycle_calculation()` function is approximately 200 ms without optimization. The total time needed to obtain a desired EMR fingerprint can be estimated as $(t_{\text{fast\_scan}} + t_{\text{cycle\_cal}}) \times \frac{f_{\text{max}} - f_{\text{min}}}{f_{\text{step}}} + (t_{\text{scan}} + t_{\text{cycle\_cal}}) \times 2$, around 7.3 s in our setting.

4.3.2. Feature Extraction and Identification. The upper three subfigures in Figure 9 display the EMR signals from three distinct peripherals, captured using a spectrum analyzer. These peripherals include a camera, an USB-network card, and a portable solid-state drive. Their original EMR signals exhibit observable differences under such a high time resolution. The corresponding bottom three subfigures in Figure 9 are captured with WNIC. The backscatter module has been configured with parameters returned by Algorithm 1 to reveal the clear diamond pattern. The f_{hub} values of the three peripherals happen to be the same. By comparing them, we have the following findings:

- Figure 9(a) produces a nearly perfect triangular wave in the frequency domain, whereas Figure 9(c) features an irregular triangular wave. These differences in their SSC signals are also reflected in the edge texture of the diamond pattern in Figure 9(b) and (d).
- The triangle frequency variation of Figure 9(a) and (e) are quite similar in terms of waveform shape. An minor difference comes from their SSC cycle. Note that their phase are aligned at the beginning but differ from

2. f_{step} and t_{scan} are not independent. When f_{step} is large, $1/t_{\text{cycle}}$ might also become large, so t_{scan} must be small enough to observe detailed diamond patterns, and vice versa. To observe 1 to 4 patterns, they satisfy the condition $f_{\text{step}} \cdot t_{\text{scan}}$ = 3.

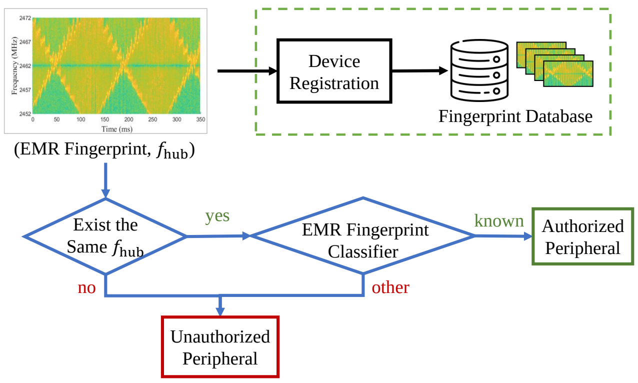


Figure 10: Procedures of Identifying EMR Fingerprint of USB Peripherals.

each other when the measurement is done. This subtle difference is magnified and also reflected in the cycle difference of the diamond pattern.

The above observation leads to our intuition for distinguishing peripheral fingerprints: First, f_{hub}, which is an estimation of the peripheral's oscillator frequency f_{usb}, is used for the initial differentiation. When the f_{hub} values fall into the indistinguishable range, we then use the image features in the EMR spectrogram for further differentiation: specifically, the edge shape of the diamond pattern and the periodic details. For this step, we use a neural network-based classifier.

Prior to training and using the classifier, we preprocess the raw EMR signals with the following two steps. Firstly, we apply time resampling. Due to the inherent scheduling mechanisms of the WNIC, the temporal intervals between consecutive spectrum scans may exhibit variability. To ensure temporal uniformity across the dataset, we resample the time axis of the collected data to achieve a consistent sampling rate. Secondly, we apply signal noise reduction and normalization. To mitigate the impact of interference within the WiFi band, we apply an averaging filter to the power measurements, and then discard any data points that fall beyond three standard deviations from the mean power level, effectively reducing the influence of outliers and noise. Besides, we normalize the power values to a common range to ensure the consistency across different environments. These preprocessing operations enhance the accuracy and robustness of subsequent fingerprint classification.

Figure 10 illustrates the process of fingerprint collection and classification. When a USB peripheral is plugged into the SecurityHub device, the host application instructs the device to use the frequency sweep method described in §4.3.1 to find an appropriate f_{hub} for the peripheral and collect its EMR fingerprint. Then, the host searches the local fingerprint database to determine whether it is a peripheral with the same f_{hub}. If there is no peripheral with the same f_{hub} in the fingerprint database, it will be directly classified as an unauthorized peripheral and no further classification is required. Otherwise, peripherals with the same f_{hub} are compared with the fingerprint and a matching probability is

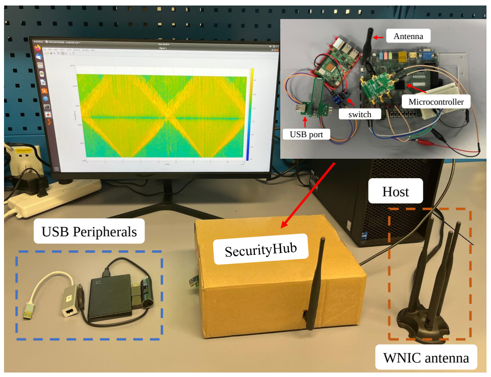


Figure 11: SecurityHub Prototype.

output for each known peripheral in the fingerprint database. The host then compares the maximum match probability to an unauthorized threshold, which is set to the minimum self-match probability for a known peripheral. If the maximum match probability is low, the test peripheral is classified as an unauthorized peripheral, and the host tell the SecurityHub device to reject the USB data connection. For the classification tasks, we utilize a classical two-dimensional convolutional neural network, *resnet18*, and it is worth mentioning that our focus is not to find the best classifier, but to validate the feasibility and effectiveness of EMR fingerprints.

5. Implementation

Our prototype is shown in Figure 11. It consists of the SecurityHub device and the host application.

SecurityHub Device: We use a Zynq-7000 FPGA to implement the backscatter logic, which controls the impedance of the antenna through an ADG902 RF switch [1]. We generate backscatter signal based on Equation 3, which are binarized to create the voltage signal needed to drive the antenna switch. The waveform parameters are configured according to control messages from the host application. Although we could reuse this FPGA to switch USB cables and receive host messages as well, for convenience, we use a Raspberry Pi 3 for these tasks. It controls an optocoupler to physically open and short the USB data lines and is also responsible for storing and replaying the `ping.LFPS` message.

Host Computer and SecurityHub Host Application: The host application runs on a PC with Intel i5 8500 processor and 8 GB RAM. The operating system is Ubuntu 16.04 LTS. We use COMPEX WLE1216VX NIC [13] with Qualcomm QCA9984 Wi-Fi chip inside as the WNIC. The WNIC can be configured to perform spectral scan through the `debugfs` of the `ath10k` driver [9]. The host uses a USB-to-TTL converter and pyserial [6] to send/receive messages to/from the hub. The algorithms to search SSC frequency is implemented in MATLAB. The fingerprint classifier is implemented using PyTorch [7] along with a pre-trained ResNet18 model.

Cost Analysis: We used the Zynq FPGA due to the popularity of its Zedboard hostboard. The backscatter logic

Type	Brand & Model × Number
Ethernet Adapter	Biaze ZH21 × 10, Llano LCN5100B × 1, Samzhe AR01S × 1 , Ugreen CR111 × 1, TP-Link TL-UG310 × 1
Portable SSD	Kingston DTMAXA × 4, Teclast NLI × 1, Sandisk CZ880 × 1, Thinkplus TU180Pro × 1
Portable HDD	Seagate SRD0NF1× 4, Western Digital S480G2G0A× 1, Orico 25PW1 × 1, Netac WH11 × 1, Lenovo HD25× 1, Ugreen US221 × 1,
Camera	Microsoft Azure Kinect DK × 3, Intel D4 Board V3 × 1

TABLE 2: USB Peripherals Used in the Evaluation.

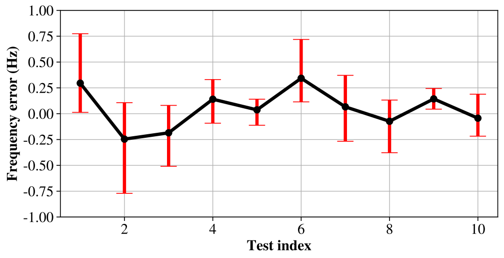


Figure 12: Estimation Error of Frequency Searching.

occupies only 1.5% of its resources. Its code is open-sourced at [8]. The logic can be ported a low-cost Microchip AGLN250V2-VQ100 FPGA costing \$20. Fabricating the logic as a chip could reduce costs further. Other components also have cheaper alternatives. The RF switch ADG902 costs less than \$5, and an STM32 microcontroller is around \$10. Although the host application could also be integrated into the SecurityHub device, the computational capabilities required by the classification algorithms would significantly increase the hub's cost. Therefore, we chose to utilize the host's network and computational resources already available to accommodate the implementation.

6. Evaluation

We evaluate the SecurityHub in the following aspects:

- Can SecurityHub successfully search out f_{hub} suitable for extracting distinctive EMR fingerprint?
- Can SecurityHub accurately classify the type, brand model, and individual of USB peripherals?
- Does the impact factors like hub-host distance, WNIC sampling interval, and ambient interference affect the effectiveness of EMR fingerprints?

Our evaluation uses 34 USB peripherals of 4 different types from 17 different brands. Devices of the same brand are of the same model and from the same manufacture batch (we purchased in bulk). Table 2 details their information.

6.1. Effectiveness of Frequency Searching

As depicted in Figure 10, f_{hub} derived from the frequency searching results of Algorithm 1, is pivotal for device identification. Therefore, in this section, we evaluate whether f_{hub} is accurately determined. To achieve this, we select 10 different USB peripherals. We first utilize the USRP B210 SDR to capture their raw EMR signals and measured their SSC period f_{usb}, representing the ground truth values. Subsequently, we use the SecurityHub system to capture the backscattered EMR signals and calculate $f_{\text{usb\_candi}}$ using Algorithm 1. The expectation is that if the algorithm functions correctly, then the (smaller) difference between $f_{\text{usb\_candi}}$ and the ground truth, *i.e.*, frequency error, should be limited. If so, the f_{hub} calculated using $f_{\text{usb\_candi}}$ will also meet the design objectives. The results, displayed in Figure 12, show the frequency error is within 1 Hz, confirming the expectation.

Type	Acc	F1	Pr	Recall	Gm
Ethernet Adapter	0.98	0.98	1.00	0.97	0.98
Portable SSD	0.97	0.98	0.99	0.98	0.99
Portable HDD	0.98	0.99	0.98	1.00	0.99
Camera	0.99	0.99	1.00	0.98	0.99
Average	0.98	0.99	0.99	0.98	0.99
Standard Deviation	0.01	0.01	0.01	0.01	0.01

TABLE 3: Inter-type Classification Results (Scenario 1).

6.2. Effectiveness of Fingerprint Identification

In this section, we assess the effectiveness of USB peripheral identification with different granularity levels: inter-type, inter-model (brand), and intra-model (brand) classification.

6.2.1. Datasets. For each peripheral, 80 independent EMR fingerprint samples are captured at different times. With a resampling interval of 240 μs, an FFT bin size of 256, and a total sampling duration of 0.3 s, the EMR spectrogram matrix has dimensions of 1250×256, which is used as input for the classification model. We employ 4-fold cross-validation [10], using 60 samples for training and 20 samples for testing.

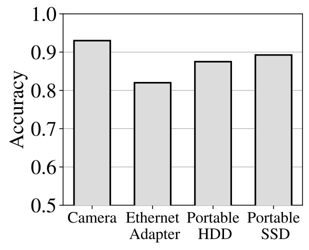
Among all peripherals listed in Table 2, a non-negligible portion of f_{hub} values differ from one another. For example, 6 out of a total of 17 models can be uniquely distinguished using their f_{hub}. To avoid discussing these trivial cases, the following tests only include fingerprint samples that cannot be differentiated by f_{hub} for training and testing.

6.2.2. Scenario 1: Inter-type Classification.. In this part, we aim to quantitatively assess whether the EMR fingerprints exemplified in Figure 9 can achieve accurate type classification. We examine four common types of USB peripherals: cameras, Ethernet adapters, portable solid-state drives (SSD), and portable hard disk drives (HDD). Table 3 presents the average accuracy, precision, recall, F1 score and G-mean. An average F1 score of 99% validates the effectiveness of EMR fingerprint features for type classification.

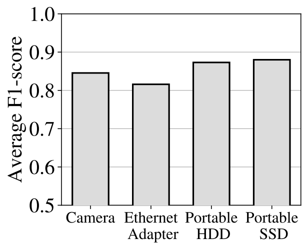
6.2.3. Scenario 2: Inter-model Classification.. In this part, we aim to explore finer classification beyond merely identifying the type. We intend to differentiate peripheral model

Model	Type	Acc	F1	Pr	Re	Gm
Samzhe AR01S	Ethernet Adapter	0.97	0.95	0.97	0.93	0.95
Ugreen CR111		0.97	0.96	0.93	1.00	0.96
Sandisk CZ880	Portable SSD	0.99	0.97	1.0	0.95	0.97
Teclast NLI		0.99	0.98	0.95	1.00	0.98
Orico 25PW1	Portable HDD	0.85	0.85	0.86	0.84	0.85
Netac WH11		0.84	0.83	0.84	0.82	0.83
Lenovo HD25		0.86	0.87	0.88	0.86	0.87
Average		0.93	0.92	0.92	0.91	0.92
Standard Deviation		0.07	0.06	0.06	0.07	0.06

TABLE 4: Inter-model Classification Results (Scenario 2).



(a) Accuracy



(b) F1-score

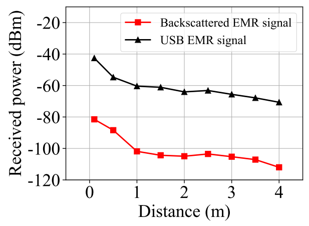
Figure 13: Intra-model Classification Results (Scenario 3).

within the same type. Accordingly, each brand model listed in Table 2 is treated as a separate class in this scenario. The test protocol is similar to the inter-type classification, and the results are presented in Table 4. Overall, the F1 score is 92% with a standard deviation of 6%, indicating a high level of accuracy. However, we observe that the accuracy the category of portable HDD is slightly lower, with an F1 score of approximately 85%. This may be because the control boards of these HDDs use similar circuit designs, leading to very similar diamond patterns in their fingerprints.

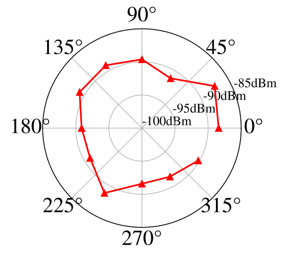
6.2.4. Scenario 3: Intra-model Classification. In this part, we aim to distinguish individual peripherals of the same type and model. This represents the most challenging scenario, as industrial products of the same type and model typically have highly similar components. To achieve this differentiation, we extend the sampling duration t_{scan} to 1.2 s and decrease the round step f_{step} in Algorithm 1 to 1 Hz. This adjustment aims to fine-tune f_{hub} to ensure that the EMR fingerprint contains exactly 1 to 2 diamond patterns, thereby amplifying the subtle features as much as possible. To maintain a consistent input dimension, the resampling interval is adjusted to 960 μs. We select models containing multiple peripherals, *e.g.*, Microsoft Azure Kinect DK in the Camera Type, and measure the classification accuracy within each model. The results are presented in Figure 13. Compared to the previous two scenarios, the F1 score is lower, indicating that the fingerprints are more similar. However, all values are still above 0.8, which suggests that the design has the capability to distinguish peripherals of the same model, though it requires a longer identification time.

6.3. Impacting Factors

6.3.1. Impact of Distance and Orientation between Hub and Host.. There is usually a separation between the host's WNIC and the USB hub. In this part, we will evaluate the impact of distance and the hub's orientation. We use 2 dBi omnidirectional antennas on the host's WNIC. In an open laboratory environment, the SecurityHub device is placed at different distances from the WNIC' antenna. We use the peak FFT power to indicate the received strength of the backscattered EMR signals. Additionally, we use a USRP B210 to measure the power of original USB EMR signals (near 2.5 GHz) for comparison. The resulting curves are shown in Figure 14(a). The received EMR power decreases as the distance increases. The WNIC can be used to observe EMR signals at distances up to 4 m. This is quite sufficient for the USB hub application, which is typically close to the host. Moreover, Figure 14(b) shows the received power in different horizontal directions at distance 0.5 m from the host is quite similar, which implies that there is no special requirements for the placement of the hub.

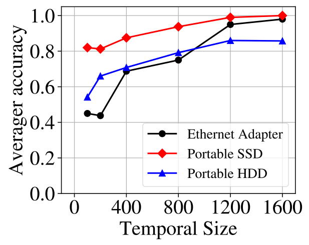


(a) Power v.s. Distance

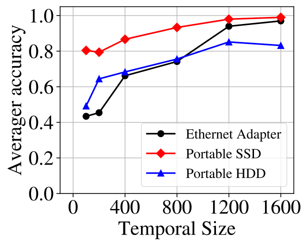


(b) Power v.s. Direction

Figure 14: Received Power of Backscattered EMR Signals v.s. Hub-Host Distance and Orientation.



(a) Accuracy



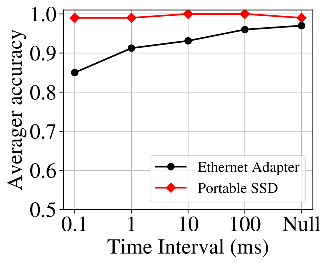
(b) F1-score

Figure 15: Classification Results v.s. Temporal Granularity.

6.3.2. Impact of Temporal Granularity.. With a given WNIC scanning duration t_{scan}, a fingerprint containing more temporal samples tends to capture finer-grained features. In this part, we investigate this factor by modifying the resampling interval on the raw EMR signals. Fingerprints of the same duration but different temporal sizes, resulting from this adjustment, are used for training and testing. As depicted in Figure 15, the classification performance on Section 6.2.3 Scenario 2 improves as temporal size increases. This is reasonable because denser samples preserve more information for the classifier. The trend levels off when the size is more than 1200. One reason is that this value is close to the density of raw WNIC samples. Another reason is that the additional information brought by denser data diminishes. Therefore, we choose a value around 1200 as the

Source	Clock (GHz)
DDR3	0.67, 0.8
DDR4	1,1.3
DDR5	0.8, 2.4, 3.2
GDDR5	2, 4
GDDR6	2, 2.5, 3.5, 7
USB3.x	**2.5**

(a) EMR Sources [49], [46]



(b) WiFi Inter-packet Interval

Figure 16: Impact of Interference. (a) shows that clock frequencies of other EMR sources have little overlap with USB EMR; (b) shows classification results under the interference of ambient WiFi transmission with various traffic intensity, *i.e.*, smaller inter-packet intervals indicate higher transmission rates. "Null" denotes no WiFi transmission.

default temporal size to achieve high classification accuracy and, consequently, enhanced security.

6.3.3. Impact of Other EMR Signals.. The host computer also contains other EMR sources, but we did not observe interference from them in any of the experiments we conducted. There are two main reasons for this. First, these EMR sources do not overlap with 2.5 GHz. For example, existing research [49], [46] has measured the clock signals of double data rate (DDR) random-access memory (RAM) used by CPUs and GPUs. Some of these frequency points are listed in Figure 16(a). In most cases, the DDR EMR signals induced by read/write operations do not fall near 2.5 GHz. Even in cases of complete overlap, such as with certain GPUs using DDR6 (which we did not test) or actively in-using USB peripherals, we did not detect notable interference. This is because, for the USB peripherals under testing, we use the compliance mode for 01 transmissions, which intensively focuses the USB EMR spectrum. As shown in Figure 3, typical data transmissions are nearly random, leading to dispersed spectral power. Consequently, the presence of these sources primarily increases noise level.

6.3.4. Impact of Ambient WiFi Transmission.. The host uses a WNIC to collect EMR signals, which means it also picks up WiFi packets operating in the same frequency band. To evaluate this interference, we set up a controlled, aggressive WiFi transmission source with another computer to generate WiFi packets. This source is placed beside the host and is configured to operate on 2.4 GHz channel 11, sharing the same central frequency as the backscattered EMR signals. It uses the 20 MHz 802.11g protocol with a 6 Mbps physical rate to send 1000-bit packets. All other settings are left at their defaults. We reconduct the classification task of Section 6.2.3 Scenario 2 with different packet intervals to assess the impact.

As shown in Figure 16 (b), WiFi transmission has no significant impact on the of SSD types. However, under denser traffic conditions, the classification accuracy for Ethernet adapters drops to 85%. We attribute this performance difference to the fact that SSD fingerprints are primarily distinguished by the cycle of their diamond patterns, while Ethernet adapter fingerprints rely more on the edge features of these patterns, as their cycles are relatively similar. Consequently, denser WiFi signals introduce more frequency domain noise, which reduces the distinguishability of Ethernet adapter fingerprints. However, it is worth noting that in practical usage cases, traffic density is unlikely to reach such high levels, and interfering sources are not likely positioned close to the host.

7. Security Analysis

In this section, we provide a detailed analysis of the SecurityHub's security properties and identify its capabilities and limitations in defending against common and advanced USB attacks.

Tian et al. [45] have provided a comprehensive summary of the attack vectors within the USB ecosystem. Many attacks involving USB peripherals rely on establishing a data connection with the host and then exploiting various vulnerabilities for malicious purposes. For example, the operating system's trust-by-default strategy allows malicious peripherals to hide their program execution [5], [35], input/output [33], and other unexpected capabilities by imitating regular peripherals. Hidden files and protocol vulnerabilities can also be exploited to facilitate data theft [38] and code injection [40]. The design goal of SecurityHub is not to counter these specific attacks but rather to prevent data connections between the host and untrusted USB peripherals by verifying the identity of the peripherals. Since the attacker's peripherals are not registered, SecurityHub can defend against general attacks that depend on data connections. However, advanced attackers who are aware of SecurityHub's defense mechanisms may develop bypass methods as described below:

Twin Hardware Attack: The attacker uses a peripheral with a physical fingerprint similar to that of an authorized peripheral to carry out the attack. There are two main possible approaches: First, the attacker can use components of identical models (*e.g.*, by observing the brand and model of an authorized peripheral or purchasing a same-model peripheral) to fabricate the malicious peripheral. However, due to minor differences between hardware components (even of the same model), SecurityHub can still detect differences in their EMR fingerprints (Section 6.2.4). Alternatively, the attacker can directly replace the software of an authorized peripheral to its own, *e.g.*, by flashing the firmware [17], or even retain only the USB parts while replacing the rest of the circuit to its own. The modified peripheral will be identified by SecurityHub as authorized because its USB parts—and thus the fingerprint—remain unchanged. However, this approach requires obtaining and modifying an authorized peripheral, which might not always be feasible, and both software and hardware replacements are also challenging reverse-engineering tasks if the design information is proprietary.

Advanced Replay Attack: This method involves the attacker collecting an EMR trace of an authorized peripheral and then replaying these signals to the SecurityHub device during EMR signal collection to bypass the identification process. This attack relies on high-precision signal collection and replay, which makes the equipment cost relatively high. Additionally, the attacker must have the opportunity to collect signals in close proximity when an authorized USB peripheral is being authenticated. From a defense perspective, the host can intentionally introduce some randomness in the backscatter modulation that only the host can later remove, to limit the effectiveness of the replay attack.

Connection Spoofing Attack: The attacker can exploit the data connection of an authorized peripheral to bypass physical fingerprint checking. Some related work has demonstrated specific methods, such as using on-path sniffer [2], [3] or off-path traffic hijacking [24]. The former attack method typically targets wireless USB peripherals using unencrypted radio communication and the latter method requires first obtaining an authorized peripheral. SecurityHub currently cannot defend against these attacks because it cannot verify the source of USB signals once the connection is established. A simple improvement could be to regularly invoke the identification process. Additionally, physically fingerprinting the entire data transfer process is an intriguing problem that deserves further exploration.

Conectionless Attack: Some attacks based on USB peripherals do not rely on USB data connections. For example, some side-channel attacks use the USB interface merely to capture power-line EMR signals emitted by other connected peripherals to infer sensitive information [23], [39], [43]. Some signal injection attacks exploit the USB port to release high power to damage devices [16]. SecurityHub cannot effectively defend against these attacks, as some of them may not even necessarily rely on the USB interface.

8. Related Work

The authenticity of devices is an effective defensive measure against many attack vectors, making it a long-standing topic in the field. Previous research [21] shows that variations in the manufacturing process lead to subtle differences in every hardware component, such as chips, circuit boards, and oscillators. These variations cause these components to exhibit unique properties that are difficult to replicate. A significant portion of existing studies leverages such inherent variations as device identifier.

Physical Unclonable Functions (PUFs) are a hardware implementation of the above concept. PUFs can generate unique responses to inputs and are often integrated into chips with high security requirements as their identifiers [44]. However, ordinary chips may not include PUFs due to cost considerations, prompting further exploration of other types of hardware fingerprints. For example, variations in radio hardware can be reflected in the waveforms they produce, which have been proposed as hardware fingerprints for wireless devices [28]. In devices lacking active signal generation capabilities, some works use power analysis techniques to passively collect and extract features from power traces as fingerprints [42]. Similarly, in high-speed circuits, a portion of the signal within the line can be radiated and serve as EMR fingerprints [41], [19], [27]. Additionally, some studies utilize other function-dependent features, such as access latency in storage blocks [22] and timing discrepancies during communication [37], as fingerprints.

In the context of USB peripherals, all of the aforementioned fingerprinting approaches have been attempted. For instance, PowerID [42] utilizes USB power traces, while MAGNETO [31] leverages the unique EMR waveforms produced when USB peripherals are inserted into USB ports as fingerprints. However, these works require additional measurement tools, such as oscilloscopes or software-defined radio, for data collection and analysis. A recent study [22] pointed out that the access latency of USB storage devices exhibits distinguishability, indicating a promising USB identifier that can be obtained solely via software. Unlike these previous works, SecurityHub distinguishes USB peripherals using features within the EMR signals generated by USB data clocks. In terms of functionality, what sets SecurityHub apart is its use of commercial WNICs to collect EMR signals, differing from approaches that rely on dedicated equipment and overwhelming computing resources. Additionally, SecurityHub supports the identification of general USB 3.x peripherals beyond just storage devices.

Technically, SecurityHub employs backscatter technology, which has been widely discussed and developed in the wireless networking field [34]. Backscatter is typically used for data transmission [15], [29] and wireless localization [36], [32], and its application for demodulating EMR signals represents a unique technical innovation of SecurityHub.

9. Conclusion and Future Work

The security issues associated with USB peripherals, combined with the spectrum scanning capabilities of modern WNIC, motivate us to utilize WNIC to collect USB EMR signals as fingerprints to avoid connections with untrusted USB peripherals. We design, implement, and evaluate SecurityHub, a security-enhanced USB hub to achieve this. The core of SecurityHubis its use of a backscatter module in the hub to convert raw USB EMR signals into a format that can be captured by commercial WNICs and that also exhibits distinguishable features. Evaluations demonstrate its capability in in identifying peripheral types, brand models, and identities. Additionally, other high-speed data buses, such as memory [41] and CPUs [19], also emit EMR signals around 2.4 GHz and exhibit similar SSC patterns. Thus, SecurityHub can be adapted to fingerprint them and we foresee its potential applications in boarder settings.

10. Acknowledgement

We sincerely thank the anonymous reviewers and our shepherd for their valuable comments and suggestions, which have been instrumental in improving and refining this paper. This work is supported by the Guangdong Provincial Key Lab of Integrated Communication, Sensing and Computation for Ubiquitous Internet of Things (No.2023B1212010007) and NSFC 62002224.

References

[1] Adg902. https://www.analog.com/en/products/adg902.html.

[2] Bastille: Keysniffer. https://www.bastille.net/research/vulnerabilities/keysniffer-intro/.

[3] Bastille: Mousejack. https://www.bastille.net/research/vulnerabilities/mousejack/.

[4] fminsearch. https://www.mathworks.com/help/matlab/ref/fminsearch.html.

[5] K. nohl and j. lell, "badusb-on accessories that turn evil,". Black Hat USA, vol. 1, no. 9, pp. 1–22, 2014.

[6] pyserial. https://pyserial.readthedocs.io/en/latest/pyserial.html.

[7] Pytorch. https://pytorch.org/.

[8] Securityhub fpga project. https://github.com/zlab-pub/SecurityHub.

[9] Spectral scan. https://wireless.wiki.kernel.org/en/users/drivers/ath10k/spectral.

[10] Stratifiedkfold. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html.

[11] Usb 3.2 revision 1.1 - june 2022. https://www.usb.org/document-library/usb-32-revision-11-june-2022.

[12] Usbkill attack. https://usbkill.com/.

[13] Wle1216vx wi-fi nic. https://compex.com.sg/shop/wifi-module/wle1216vx-2/.

[14] Ieee standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements–part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications–amendment 5: Enhancements for high efficiency wlan, May 2021.

[15] Ali Abedi, Farzan Dehbashi, Mohammad Hossein Mazaheri, Omid Abari, and Tim Brecht. Witag: Seamless wifi backscatter communication. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pages 240–252, 2020.

[16] O. Angelopoulou, S. Pourmoafi, A. Jones, and G. Sharma. Killing your device via your usb port. In *Thirteenth International Symposium on Human Aspects of Information Security & Assurance*, pages 61–72. The Centre for Security, Communications and Network Research (CSCAN), July 2019.

[17] Matthew Brocker and Stephen Checkoway. iSeeYou: Disabling the MacBook webcam indicator LED. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 337–352, San Diego, CA, August 2014. USENIX Association.

[18] Ruirong Chen and Wei Gao. Transfi: emulating custom wireless physical layer from commodity wifi. In *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services*, pages 357–370, 2022.

[19] Yushi Cheng, Xiaoyu Ji, Juchuan Zhang, Wenyuan Xu, and Yi-Chao Chen. Demicpu: Device fingerprinting with magnetic signals radiated by cpu. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 1149–1170, New York, NY, USA, 2019. Association for Computing Machinery.

[20] John Clark, Sylvain Leblanc, and Scott Knight. Risks associated with usb hardware trojan devices used by insiders. In *2011 IEEE International Systems Conference*, pages 201–208, 2011.

[21] William E. Cobb, Eric D. Laspe, Rusty O. Baldwin, Michael A. Temple, and Yong C. Kim. Intrinsic physical-layer authentication of integrated circuits. *IEEE Transactions on Information Forensics and Security*, 7(1):14–24, 2012.

[22] Patrick Cronin, Xing Gao, Haining Wang, and Chase Cotton. Time-print: Authenticating usb flash drives with novel timing fingerprints. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1002–1017, 2022.

[23] Patrick Cronin, Xing Gao, Chengmo Yang, and Haining Wang. {Charger-Surfing}: Exploiting a power line {Side-Channel} for smartphone information leakage. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 681–698, 2021.

[24] Robert Dumitru, Daniel Genkin, Andrew Wabnitz, and Yuval Yarom. The impostor among us(b): Off-path injection attacks on usb communications. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 5863–5880, 2023.

[25] Manideep Dunna, Miao Meng, Po-Han Wang, Chi Zhang, Patrick Mercier, and Dinesh Bharadia. Syncscatter: Enabling wifi like synchronization and range for wifi backscatter communication. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, pages 923–937, 2021.

[26] Rezky Aulia Efendy, Ahmad Almaarif, Avon Budiono, Muhardi Saputra, Warih Puspitasari, and Edi Sutoyo. Exploring the possibility of usb based fork bomb attack on windows environment. In *2019 International Conference on ICT for Smart Society (ICISS)*, volume 7, pages 1–4, 2019.

[27] Justin Feng, Tianyi Zhao, Shamik Sarkar, Dominic Konrad, Timothy Jacques, Danijela Cabric, and Nader Sehatbakhsh. Fingerprinting iot devices using latent physical side-channels. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 7(2), jun 2023.

[28] Tianbo Gu and Prasant Mohapatra. Bf-iot: Securing the iot networks via fingerprinting-based device authentication. In *2018 IEEE 15th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pages 254–262, 2018.

[29] Xiuzhen Guo, Yuan He, Zihao Yu, Jiacheng Zhang, Yunhao Liu, and Longfei Shangguan. Rf-transformer: a unified backscatter radio hardware abstraction. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, MobiCom '22, page 446–458, New York, NY, USA, 2022. Association for Computing Machinery.

[30] Mordechai Guri, Matan Monitz, and Yuval Elovici. Usbee: Air-gap covert-channel via electromagnetic emission from usb. In *2016 14th Annual Conference on Privacy, Security and Trust (PST)*, pages 264–268. IEEE, 2016.

[31] Omar Adel Ibrahim, Savio Sciancalepore, Gabriele Oligeri, and Roberto Di Pietro. Magneto: Fingerprinting usb flash drives via unintentional magnetic emissions. *ACM Trans. Embed. Comput. Syst.*, 20(1), dec 2020.

[32] Jinyan Jiang, Jiliang Wang, Yijie Chen, Yihao Liu, and Yunhao Liu. Locra: Enable practical long-range backscatter localization for low-cost tags. In *Proceedings of the 21st Annual International Conference on Mobile Systems, Applications and Services*, MobiSys '23, page 317–329, New York, NY, USA, 2023. Association for Computing Machinery.

[33] Evangelos Karystinos, Antonios Andreatos, and Christos Douligeris. Spyduino: Arduino as a hid exploiting the badusb vulnerability. In *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 279–283, 2019.

[34] Bryce Kellogg, Vamsi Talla, Shyamnath Gollakota, and Joshua R Smith. Passive wi-fi: Bringing low power to wi-fi transmissions. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 151–164, 2016.

[35] Hongyi Lu, Yechang Wu, Shuqing Li, You Lin, Chaozu Zhang, and Fengwei Zhang. Badusb-c: Revisiting badusb with type-c. In *2021 IEEE Security and Privacy Workshops (SPW)*, pages 327–338, 2021.

[36] Zhihong Luo, Qiping Zhang, Yunfei Ma, Manish Singh, and Fadel Adib. 3d backscatter localization for fine-grained robotics. In *16th USENIX symposium on networked systems design and implementation (NSDI 19)*, pages 765–782, 2019.

[37] John V. Monaco. Device fingerprinting with peripheral timestamps. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1018–1033, 2022.

[38] Abdul Azies Muslim, Avon Budiono, and Ahmad Almaarif. Implementation and analysis of usb based password stealer using powershell in google chrome and mozilla firefox. In *2020 3rd International Conference on Computer and Informatics Engineering (IC2IE)*, pages 421–426, 2020.

[39] Tao Ni, Yongliang Chen, Weitao Xu, Lei Xue, and Qingchuan Zhao. Xporter: A study of the multi-port charger security on privacy leakage and voice injection. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, pages 1–15, 2023.

[40] Annisa Dwiayu Ramadhanty, Avon Budiono, and Ahmad Almaarif. Implementation and analysis of keyboard injection attack using usb devices in windows operating system. In *2020 3rd International Conference on Computer and Informatics Engineering (IC2IE)*, pages 449–454, 2020.

[41] Cheng Shen, Jun Huang, Guangyu Sun, and Jingshu Chen. Electromagnetic fingerprinting of memory heartbeats: System and applications. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 6(3), sep 2022.

[42] Riccardo Spolaor, Hao Liu, Federico Turrin, Mauro Conti, and Xiuzhen Cheng. Plug and power: Fingerprinting usb powered peripherals via power side-channel. In *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*, pages 1–10, 2023.

[43] Yang Su, Daniel Genkin, Damith Ranasinghe, and Yuval Yarom. {USB} snooping made easy: crosstalk leakage attacks on {USB} hubs. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1145–1161, 2017.

[44] Kuniyasu Suzaki, Yohei Hori, Kazukuni Kobara, and Mohammad Mannan. Deviceveil: Robust authentication for individual usb devices using physical unclonable functions. In *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 302–314, 2019.

[45] Jing Tian, Nolen Scaife, Deepak Kumar, Michael Bailey, Adam Bates, and Kevin Butler. Sok: "plug & pray" today – understanding usb insecurity in versions 1 through c. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 1032–1047, 2018.

[46] Zihao Zhan, Zhenkai Zhang, Sisheng Liang, Fan Yao, and Xenofon Koutsoukos. Graphics peeping unit: Exploiting em side-channel information of gpus to eavesdrop on your neighbors. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1440–1457, 2022.

[47] Pengyu Zhang, Dinesh Bharadia, Kiran Joshi, and Sachin Katti. Hitchhike: Practical backscatter using commodity wifi. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*, pages 259–271, 2016.

[48] Yipeng Zhang, Zhonghao Sun, Liqun Yang, Zhoujun Li, Qiang Zeng, Yueying He, and Xiaoming Zhang. All your plcs belong to me: Ics ransomware is realistic. In *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 502–509, 2020.

[49] Zhenkai Zhang, Zihao Zhan, Daniel Balasubramanian, Bo Li, Peter Volgyesi, and Xenofon Koutsoukos. Leveraging em side-channel information to detect rowhammer attacks. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 729–746, 2020.