Fingerprinting IoT Devices Using Latent Physical Side-Channels

JUSTIN FENG, UCLA, USA
TIANYI ZHAO, UCLA, USA
SHAMIK SARKAR, UCLA, USA
DOMINIC KONRAD, UCLA, USA
TIMOTHY JACQUES, UCLA, USA
DANIJELA CABRIC, UCLA, USA
NADER SEHATBAKHSH, UCLA, USA

The proliferation of low-end low-power internet-of-things (IoT) devices in "smart" environments necessitates secure identification and authentication of these devices via low-overhead *fingerprinting* methods. Previous work typically utilizes characteristics of the device's wireless modulation (WiFi, BLE, etc.) in the spectrum, or more recently, electromagnetic emanations from the device's DRAM to perform fingerprinting. The problem is that many devices, especially low-end IoT/embedded systems, may not have transmitter modules, DRAM, or other complex components, therefore making fingerprinting infeasible or challenging. To address this concern, we utilize electromagnetic emanations derived from the processor's clock to fingerprint. We present Digitus, an emanations-based fingerprinting system that can authenticate IoT devices at range. The advantage of Digitus is that we can authenticate low-power IoT devices using features intrinsic to their normal operation without the need for additional transmitters and/or other complex components such as DRAM. Our experiments demonstrate that we achieve ≥ 95% accuracy on average, applicability in a wide range of IoT scenarios (range ≥ 5m, non-line-of-sight, etc.), as well as support for IoT applications such as finding hidden devices. Digitus represents a low-overhead solution for the authentication of low-end IoT devices.

CCS Concepts: • **Computer systems organization** → **Embedded hardware**; • **Security and privacy** → **Authentication**.

Additional Key Words and Phrases: physical side-channels, fingerprinting, internet-of-things

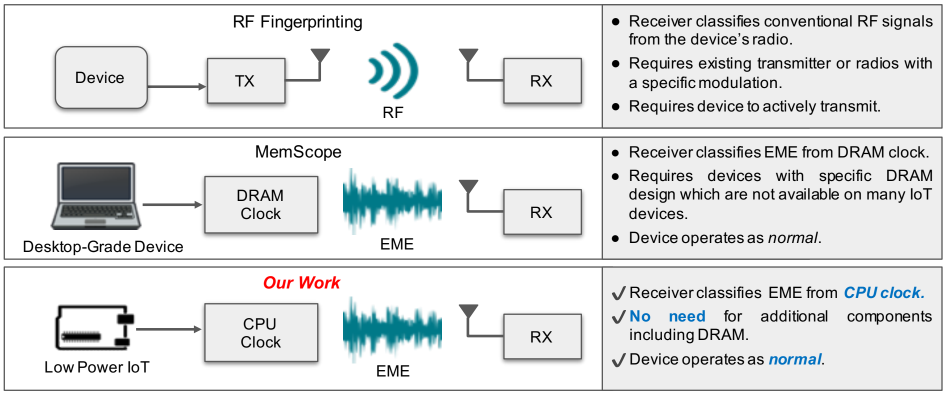
ACM Reference Format:
Justin Feng, Tianyi Zhao, Shamik Sarkar, Dominic Konrad, Timothy Jacques, Danijela Cabric, and Nader Sehatbakhsh. 2023. Fingerprinting IoT Devices Using Latent Physical Side-Channels. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 7, 2, Article 54 (June 2023), 26 pages. https://doi.org/10.1145/3596247

1 INTRODUCTION

In the rapidly growing world of IoT and embedded devices, especially low-cost, resource-constrained devices, the demand for low-overhead secure identification and authentication methods has become increasingly important. Among various solutions, *device fingerprinting* has received particular interest due to its advantages including being external and non-invasive, low-power, and (almost) zero-overhead, making it suitable for detecting and authenticating low-end and low-power IoT/embedded devices.

Authors' addresses: Justin Feng, jfeng10@ucla.edu, UCLA, Los Angeles, California, USA; Tianyi Zhao, zhaotianyi@ucla.edu, UCLA, Los Angeles, California, USA; Shamik Sarkar, shamiksarkar@ucla.edu, UCLA, Los Angeles, California, USA; Dominic Konrad, djkonrad@ucla.edu, UCLA, Los Angeles, California, USA; Timothy Jacques, tjacques888@ucla.edu, UCLA, Los Angeles, California, USA; Danijela Cabric, danijela@ee.ucla.edu, UCLA, Los Angeles, California, USA; Nader Sehatbakhsh, nsehat@ee.ucla.edu, UCLA, Los Angeles, California, USA.

2474-9567/2023/6-ART54
https://doi.org/10.1145/3596247



- Receiver classifies conventional RF signals from the device's radio.
- Requires existing transmitter or radios with a specific modulation.
- Requires device to actively transmit.

- Receiver classifies EME from DRAM clock.
- Requires devices with specific DRAM design which are not available on many IoT devices.
- Device operates as *normal*.

✓ Receiver classifies EME from ***CPU clock.***
✓ **No need** for additional components including DRAM.
✓ Device operates as ***normal***.

Fig. 1. Fingerprinting devices under various scenarios. Conventional RF fingerprinting methods use various RF modalities, such as WiFi packets, to fingerprint devices. Recently, Memscope [47] leverages electromagnetic emanations (EME) from the memory's clock to fingerprint devices. Our work, Digitus, leverages EMEs from the processor clock for fingerprinting, which makes it more suitable for low-power resource-constrained IoT devices.

The fingerprinting approaches have been extensively used for authenticating known IoT devices in a network, while also being used for detecting intruders and malicious (unknown) devices in the same network [30, 35, 48, 52]. More recently, fingerprinting has been also used for *privacy protection* where a fingerprinting framework is used to detect hidden devices (e.g., a hidden camera, a network eavesdropper, etc.) in an environment (e.g., home, office, etc.) [12, 29, 40, 45, 46].

To achieve the above goals for fingerprinting, existing approaches leverage different signal modalities created by the target devices, including RF signals, electromagnetic radiations, and even magnetic signals [15, 22, 53]. The main shortcoming of the state-of-the-art, however, is the lack of a generic approach that is applicable to low-end IoT devices *with or without* radios and/or other complex components such as DRAM, while being robust against various attacks and operating in a reasonable distance (few meters away). Given the growing popularity of "smart" environments filled with low-end and low-power IoT and embedded devices, such a generic yet effective approach is much needed.

To address this pressing need, we propose a new method called *Digitus* that is suitable for fingerprinting low-end and resource-constrained IoT and embedded devices. Digitus is a framework that can authenticate devices at range using latent electromagnetic (EM) emanations radiated from the embedded/IoT devices. *The key idea is to leverage the unique features of the processor's clock and its various modulations, which are widely available across a large variety of devices including low-end embedded and IoT systems, to fingerprint devices.* To achieve high accuracy, Digitus employs features engineering and deep neural networks to extract features and then uses various machine learning classifiers to make a decision.

In comparison with the state-of-the-art, this work offers several advantages, which are described below and summarized in Fig. 1.

- **Support for Low-End IoT and Embedded Devices.** While leveraging electromagnetic emanations for fingerprinting is not new, our approach improves the current state-of-the-art, particularly Memscope [47], by leveraging *only* the EM signals and features created in the processor – i.e., unlike state-of-the-art, our approach uses *only* the processor and its clock and doesn't rely on devices having other components like DRAM, which are *not* available in low-end IoT devices.

- **Fingerprinting at Range.** Many existing works in EM fingerprinting are only applicable in near-field scenarios. This work extends fingerprinting to long-range (≥ 5m) scenarios, which fits a more realistic setup for "smart" IoT environment scenarios.
- **Broad Applicability.** Digitus only leverages electromagnetic emanations from the processor, therefore, as will be shown in Section 7, is applicable to a wide range of IoT devices *with and without radios*.
- **Multi-device Fingerprinting.** Digitus improves the state-of-the-art by fingerprinting multiple devices simultaneously in the presence of other devices and interference. Existing approaches that leverage EM emanations for fingerprinting are very limited in fingerprinting devices in the presence of other interferences, including signals from other (similar) devices.

To realize our method, there are several challenges that need to be addressed. First, we need to find features that can be used to fingerprint various devices. This requires one to discover characteristics in the spectrum that are unique to each device and consistent over time. Secondly, we need to test Digitus in a wide range of scenarios to ensure its usefulness in practical scenarios. Third, we need to show that Digitus is robust against adversarial attacks in order to ensure that devices in our system cannot be spoofed easily. We describe our design in Sections 3 and 4 and present our results in Sections 6 and 7.

Digitus is well-suited for internet-of-things environments[1] such as smart homes, offices, hospitals, etc., where many devices and sensors may operate. These devices may be resource-constrained or low-power, so utilizing features that are common among them that do not rely on various complex components (e.g., DRAM, radio) is needed. As a result, Digitus relies on *unique* features related to the clock that are widely available in *many devices* and are reliable and robust for fingerprinting.

Specifically, the contributions in this work are:

- We propose a system that utilizes electromagnetic emanations from IoT processors to enable fingerprinting of devices that may not have radios or DRAM at range.
- We design and implement Digitus which extracts features from EM signals and uses various signal processing and machine learning techniques to fingerprint devices. Additionally, Digitus can also leverage deep-learning methods for implicit feature extraction and fingerprinting where needed.
- We evaluate our system on IoT devices in a wide range of scenarios to demonstrate the effectiveness and robustness of our approach.
- We analyze our system under two different real-world application scenarios and show its reliability and robustness against various adversaries.

The remainder of this paper is organized as follows. In Section 2, we review the current state-of-the-art and describe the challenges and needs. Section 3 describes why features from an IoT device's processor clock are unique and can be used for fingerprinting. We then present the details of our design in Section 4. The setup for our evaluations is explained in Section 5, and the main results are presented in Section 6. Two application scenarios are studied in 7. We briefly discuss future works in Section 8 and review other related works in Section 9, and present our conclusions in Section 10.

2 MOTIVATION: WHY A NEW FINGERPRINTING METHOD IS NEEDED?

The increasing presence of IoT and embedded devices in various settings, especially indoor "smart" environments such as homes, offices, and factory lines, has raised several security and privacy issues.

[1] When referring to IoT devices, we refer to embedded systems with or without transmitter modules. The benefit of our approach is that our system makes no assumptions on the supposed activity or inactivity (or lack of) the transmitter module. This provides more flexibility towards when our system can fingerprint a device.

An important issue that needs to be addressed in such settings is to properly identify the IoT devices in the environment and to authenticate them (e.g., for sending and receiving commands and/or data) when needed. Further, accurate identification is also needed to detect unauthorized devices that may interfere with the network.

Another critical concern arising from the prevalence of these computing devices is the presence of an eavesdropper, such as a camera, voice recorder, etc., in an environment. These *stealthy* devices may or may not be connected to the network, or may not even have a radio. Instead, these devices are designed to stay hidden and record sensitive information (e.g., voice, video, etc.) internally.

An effective fingerprinting method is one that can successfully and robustly cover *both* security (authentication) and privacy concerns. This means that the *fingerprinting method should be able to handle various devices with different communication technologies*, and more importantly, should be able to fingerprint devices *that don't have radios and/or other advanced and high overhead components*.

These requirements suggest that conventional RF fingerprinting methods that leverage unique hardware features of radio for identification are not broadly applicable to smart environments, although they have been quite successful in accurately fingerprinting devices with radios in various settings [6, 9, 24, 25, 32].

To address this, more recently, methods based on leveraging physical signals, created and radiated from the device, have been proposed for fingerprinting. The key advantage of these approaches is that they eliminate the need for radio frequency signals, as any electronic device creates these *unintentional* physical signals (often called physical side-channel signals).

For example, in DeMiCPU [15], the authors propose a fingerprinting technique that identifies CPUs via their magnetic induction signals. One limitation of DeMiCPU is that these signals can only be measured close by with a magnetic probe (<16mm). Thus, this approach will not be applicable in IoT scenarios where a device will need to be measured at a greater distance with an antenna.

More recently, Shen et al. proposed Memscope [47], a fingerprinting framework that leverages electromagnetic signals created by the device's DRAM memory unit. The key advantage of Memscope [47] over previous methods is that it is applicable to devices with and without radios and can fingerprint these devices from large distances. Further, compared to previous methods that leverage physical signals for fingerprinting, Memscope was the first method that could achieve fingerprinting from far-field (i.e., $\geq 1m$).

The major shortcoming of Memscope [47], however, is that it relies on EM signals from the complex DRAM memory system on the device. While effective in many classes of devices, including laptops and access points, the applicability of methods like Memscope is put in serious jeopardy when it comes to low-end IoT and embedded systems (e.g., simple microcontrollers, sensors, etc.) as these devices ***lack the necessary setup for creating the desired electromagnetic emanations***. Simple IoT and embedded systems often *only* have a simple low-end microcontroller with a small internal memory and an external (off-chip) flash memory. They typically don't support advanced memory hierarchy technologies nor other techniques such as spread spectrum clocking (CSS) which is the foundation of Memscope and other methods [55].

These observations motivate the need for designing a system that is holistic and more applicable to various classes of devices including low-end devices by leveraging the electromagnetic emanations generated by the systems' processor for fingerprinting. Unlike DRAM and radio, the processor clock is widely available in any microcontroller and/or embedded system.

Fig. 2 summarizes the various scenarios and how each method could be applicable. Table 1 compares existing works in fingerprinting. Overall, Digitus is applicable to a more generic set of devices (with or without radio and/or DRAM) making it more suitable for heterogeneous scenarios. Compared to other work, however, the limitation is a lower range. In the next section, we describe how the digital clock can be used for fingerprinting.

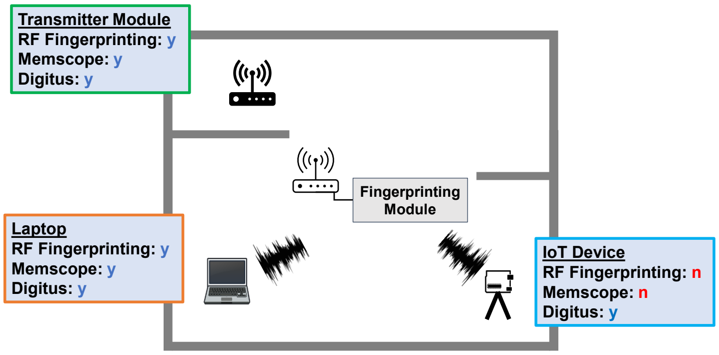


Fig. 2. A "smart" environment scenario may have many types of devices, such as a device with an RF transmitter module or access point, a laptop or a high-end IoT device, and a low-end IoT device. Digitus is the only system that satisfies the constraints of an IoT device with no added hardware.

Table 1. Comparison between Digitus and existing works. RF fingerprinting (devices with transmitter modules), near-field device fingerprinting, network-based fingerprinting, and fingerprinting of devices with complex memory are the categories. Note: F.B. = Feature-Based. D.L. = Deep Learning Based. Stat. = Statistically Based.

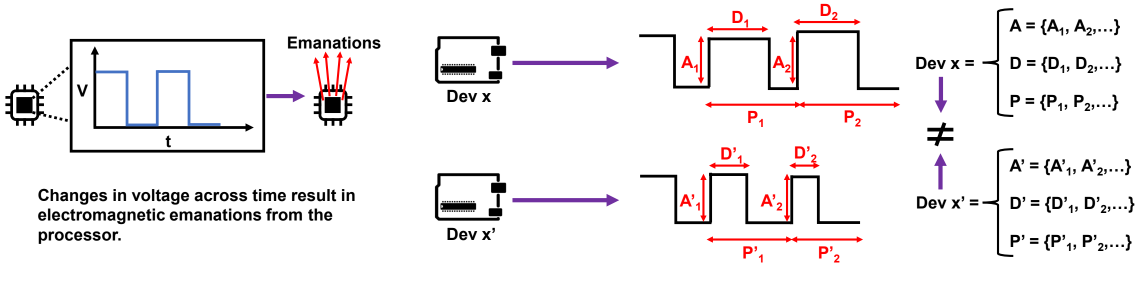
Approach	Range	Property	Limitation	Modality	Classifier
RF [6, 9, 24, 32]	~45m	TX Imperfections	TX Module	IEEE 802.11	F.B., D.L., Stat.
DeMiCPU [15]	0.02m	CPU	Near-Field	Mag. Induction	F.B.
IotFinder [37]	~45m	DNS	Same Network	IP4, IP6	Stat.
Memscope [47]	Up to 30m	Memory	Devices with DRAM	Memory Clock	F.B.
Digitus (Our Work)	Up to 7.5m	CPU	Limited Range	CPU Clock	F.B., D.L.

3 A PRIMER ON DIGITUS

Electronic devices unintentionally leak information to the outside world via the phenomena called *physical side-channels*. These signals can occur in many forms, such as electromagnetic, power, and magnetic side-channels [16, 39, 55]. In this work, we will be focusing on electromagnetic (EM) side-channels.

Computing devices leak electromagnetic emanations (sometimes called radiations) naturally during operation. The processing chip on the device and the underlying transistors and capacitors in the circuit experience voltage (and consequently current) oscillations over time, resulting in emanations, as shown in Fig. 3a. Traditionally, these emanations have been used to discover encryption keys [18, 56] or other sensitive information [44]. However, as shown more recently, these emanations can also be leveraged for non-malicious purposes [34, 42], including fingerprinting [2, 31].

Among various components that could create emanations, the processor's clock (and its circuitry) creates signals that can be measured at a distance [10, 41]. Specifically, the processor runs at the frequency of the clock, which causes switching of logic and circuit components that produces electromagnetic emanations at the clock frequency. While the usage of emanations (processor or DRAM) for malicious and non-malicious use cases have



(a)

(b)

Fig. 3. (a): Electromagnetic emanations from a processor are a direct consequence of the change in voltage and current, experienced by the hardware over time. (b): Each device has unique clock characteristics, i.e., having different *distributions* of various characteristics, such as an envelope of amplitudes (A), duty cycles (D), and periods (P). Given their uniqueness, these characteristics can be used as features for fingerprinting.

been extensively explored in the past [11, 43, 55], *the* ***new observation*** *in this work is that the EM emanations related to the processor's clock are unique to each device.*

Devices have different clocks with different characteristics, as shown in Fig. 3b. While devices of the same type may have the same clock frequency, in practice, each device has a different set of clock characteristics due to manufacturing imperfections. This may be differences in *amplitude*, *duty cycle*, as well as *period*. As a result, each device exhibits a *unique distribution* of values for these characteristics that are specific to the device at hand. Thus, one device's emanations will be different from another device's emanations.

Such an approach is fundamentally different from well-established methods in RF fingerprinting. Work has been done to identify and authenticate wireless devices at range using RF signals [9, 22, 25, 32, 52]. The key insight is that hardware imperfections within each device's transmitter module create uniqueness in the transmitted signal. Digitus, however, is different because it does not rely on additional transmitter modules, as they may not be available and/or regularly active in many low-end IoT devices. Similarly, phase-locked loops (PLLs) have been analyzed in fingerprinting [6]. While a valid approach, not all devices have PLLs in either transmission or clock generation. Additionally, while clock imperfections could be measured, the processing capabilities required to measure these imperfections at distance would be significant and further information can be derived from EM emanations beyond a singular clock peak.

This, however, comes at a new and non-trivial challenge. In RF fingerprinting, one can capture a *strong* signal that is typically wideband for data analysis with relative ease as it relies on transmitter modules and specific wireless modulations. Digitus, on the other hand, relies on typically weak signals created by emanations. As a result, more processing work needs to be done to boost our signal and to extract useful and robust features especially when it is used at range.

4 DESIGNING THE FINGERPRINTING MODEL

To realize fingerprinting of IoT devices at range via emanations, we design a framework called Digitus to process data and extract useful information relevant to each device, and classify each signal using a classifier. The overview is shown in Fig. 4. In this section, we will first present the data collection and preprocessing steps. Then, we will explain our feature extraction approach. Finally, we will describe the machine learning classification step.

Throughout this section, we use a (small) setup consisting of three identical IoT devices for explaining various steps and numbers. In the next section, we present our extended setup and its details.

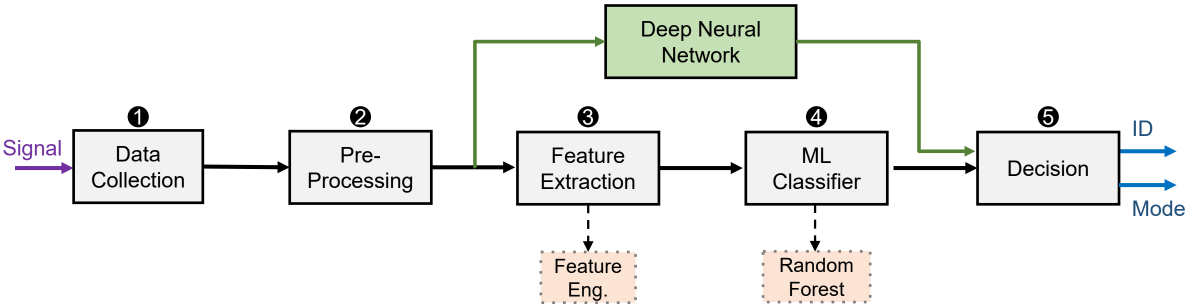


Fig. 4. This is an outline of Digitus's steps. It first collects data and prepares for classification. Digitus can use two alternative learning-based approaches for its decision-making. The primary approach consists of black arrows and gray numbered boxes. The alternate learning approach (deep-learning) that Digitus can use is shown using green arrows and the green box. When using the alternate approach, Digitus does not use the feature extraction and ML classifier boxes labeled as 3 and 4 in the figure. The final step is to classify the samples to make a decision about the device's ID and mode (i.e., fingerprinting).

We use three Arduino Uno devices as *target* devices that need to be fingerprinted, and an Ettus USRP software-defined radio (SDR) with a conventional indoor VHF/UHF antenna as the *receiver*, placed about 1 meter away from the target device.

4.1 Data Collection and Preprocessing

First, we need to collect data from our devices. To receive the emanations at far-field distances, an antenna, and an SDR are used. Different devices have different clock frequencies. Thus, it is desirable to pick a center frequency that, when chosen, allows for the capturing of each of the different possible frequencies we may observe. Additionally, due to the strong harmonic behavior of the clock signal (a rectangular-shaped signal), *multiple harmonics* of the same signal could be captured and used in our analysis.

We pick two harmonics separated by a frequency offset equivalent to a scalar multiple of the harmonic frequency (i.e., the clock frequency) such that Digitus can capture information across two bands. These two bands do not have to be consecutive harmonics, but one should pick two harmonics where the relative interference is minimal. For example, in our setup, we picked the 18th and 19th harmonics of the target device (Arduino Uno). Using more harmonics helps to minimize the potential error one may observe when calculating features, but has a tradeoff with processing complexity.

Another important design decision is to pick a sufficient bandwidth. On one hand, the bandwidth should be large enough such that it captures the range of the different clock frequencies as well as the relevant sideband activity from the different devices. On the other hand, the bandwidth should not be so large as to capture unwanted signals and increase processing time. For our setup, we use 500kHz of bandwidth.

After raw data is collected over-the-air from a device, we need to prepare the data for feature extraction (described in Section 4.2). The first step is to slice our collected data (n seconds in length) into smaller independent time slices for processing. We then compute the periodogram over portions of these slices and average over multiple windows to obtain a smoothed version of the periodogram, reducing noise.

For our deep-learning module, we compute the spectral density for each of the smaller length time slices and stack them up to form an image or a spectrogram. We compute two images for the two bands and concatenate them along the depth dimension.

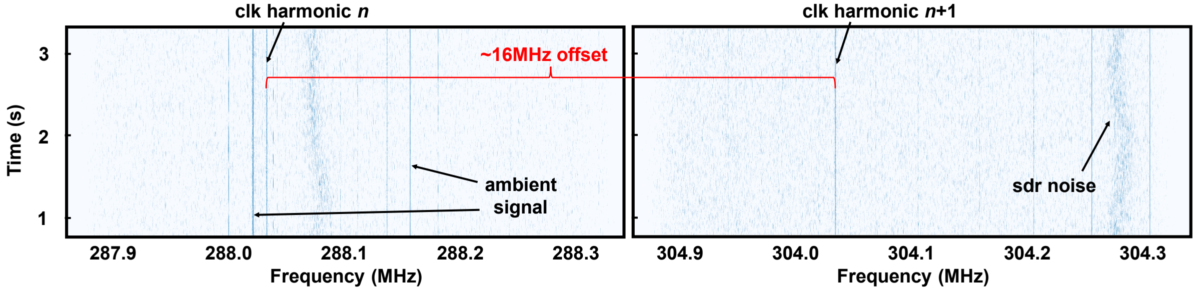


Fig. 5. Here are spectrograms of two different frequency bands with one device present. The band is sparse. The clock of the device is noted in each band, with an 16MHz offset denoting the clock harmonic frequency. The surrounding spectrum in either band is either ambient signal or noise.

4.2 Feature Extraction

4.2.1 Overview. To extract relevant features from the EM emanations created by the device's processor clock, ***we consider both (manual) feature engineering and deep learning methods***. When needed, we will compare the results for both methods and also describe how to choose between the two dynamically.

In the following, we explain our feature engineering method first, by introducing the features we find important in this problem and the methods for finding them. In Section 4.4, we will then explain our deep learning pipeline.

4.2.2 (Manual) Feature Engineering. The first step for extracting manual features is to distinguish between relevant signals and any noise and ambient signals, as shown in Fig. 5. To achieve this, we need to first find the location of the clock in the spectrum. One way we can find the clock in the spectrum is by measuring a device's clock signal in a high SNR setting in an offline phase. Once the clock frequency is found, we can reduce our search space in the spectrum by searching for peaks (energy) around the clock frequencies we found. Another way to find the clock during the fingerprinting process is to dynamically search for it. This can be done by collecting data across three or more bands to determine harmonic relations between peaks. The reason we need three or more harmonics is that we need two or more relations between peaks in order to determine a harmonic relationship and thus the clock frequency of a device.

In a more complex multi-device scenario, the algorithm is extended to detect and track multiple clocks. Details will be described in Section 7. Here, we focus on a single device only and use our feature engineering approach to extract features from the preprocessed EM emanations. We use the following features (shown as **F1**, **F2**, etc.):

F1) Clock Harmonic Frequency. The location of the device's clock harmonic within each band is the first feature used. This value, if localized correctly, is useful because each device has a unique clock frequency as same-type devices often have at least 10-100s Hz offset from their pre-defined frequency. Generally, for a given device with a clock period of *period* and a certain harmonic n, we have $Clock\ Frequency_{nth\ Harmonic} = \frac{1}{period} * n$.

Care should be taken for this feature, since an adversary could easily spoof it by creating a static tone at each frequency. Thus, this feature should not be used alone, but in tandem with other features (more details later).

F2) Clock Width. In addition to each device having a specific clock frequency, each clock has a unique response in the frequency domain. One characteristic relating to the shape of the peak in frequency is the clock width. The spike observed at the clock frequency has a particular *spread* across frequencies related to the phase noise.

For this feature to be valid, a sufficiently sized FFT should be used in relation to the bandwidth. The clock width will be more accurate when the resolution of the FFT is greater.

F3) Autocorrelation. Autocorrelation is a time-series-based feature that measures the relatedness of one series with its lagged version in time, as shown in Equation 1, where s and t refer to two different times, separated by lag h, and γ is the sample autocovariance.

$$\rho(s,t) = \frac{\gamma(s,t)}{\sqrt{\gamma(s,s)\gamma(t,t)}}, \tag{1}$$

Autocorrelation is beneficial because it provides a different viewpoint of the device (time domain instead of frequency domain). To choose the lag, we pick an offset where the autocorrelation for each device is still sufficiently different from the other devices.

If the bandwidth is large, the autocorrelation computed may include many irrelevant regions of the band. We apply a bandpass filter around the clock frequency in order to minimize the impact of irrelevant signals.

F4) Strength Ratio Between Clock Harmonics. Each device exhibits a unique frequency response across harmonics. Due to differences in the clock circuitry and the way the signals are emanated, not all harmonics exhibit the same signal strength. This feature captures the strength differences between two clock harmonics as this is unique to each device, with the calculation shown in Equation 2.

$$Strength\ Ratio = \frac{SNR\ Band\ 1}{SNR\ Band\ 2} * 100. \tag{2}$$

While using the amplitude of a singular clock could be useful, such a value is prone to errors due to different levels of attenuation or noise. By utilizing the strength ratio between two harmonics, our analysis will generally be less affected by attenuation and transient noise.

F5) Cyclic Frequency. We apply cyclostationary analysis, a concept typically used in signal processing [26], to fingerprinting for the purpose of studying higher-order characteristics of the signal. We apply the *spectral coherence* function to our data to capture cyclic relationships between different parts of the band. This is important as within a single band, there will be frequency information relevant to the target device throughout the band.

We loop over a set of cyclic frequencies, α (from $a1$ to $a2$), and obtain the spectral coherence function for each frequency. The spectral coherence function is computed by calculating the cyclic power spectrum, as shown in Equation 3, across the data. We utilize Welch's Method as provided by the Cyclic Spectral Analysis Toolbox [5]. y_n is the next window across y, and x_n is the next window across x, where y and x represent the mirror images of the same signal with center frequency f and cyclic frequency of interest α. The number of windows to compute over is defined by k.

$$Cyclic\ Power\ Spectrum = \frac{1}{k}\sum_{n=1}^{k} fft(y_n) * \overline{fft(x_n)}. \tag{3}$$

We pick the cyclic frequency α with the largest magnitude of coherence as a cyclic frequency of interest to our device, as a large value of coherence demonstrates a greater relevance to our signal.

There are a few reasons why we choose this collection of features but not any other. First, there is a combination of frequency-domain and time-domain analysis in our features. Features involving the clock (clock harmonic frequency, clock width, etc.) are in the frequency domain and capture information relevant to the frequency-domain peaks and the relationship between the peaks. There are other features that are indirectly relevant to the clock, so a time-domain feature like autocorrelation is used. We utilize cyclostationary analysis to find a feature that cannot be found by time-domain or frequency-domain analysis alone, such as cyclic frequencies.

Secondly, since we only want to focus on the clock and its relative features, other activities and parameters including those relevant to phase and/or other components, such as memory, are not included in our model.

Our feature engineering approach described above has several merits. However, for thoroughness, we use a complementary approach, specifically, a deep learning method that does not need explicit feature extraction. In

Table 2. Mean for the extracted features in three same-type devices used in our setup.

Device	Clk Harmonic Freq	Autocorr	Strength Ratio Clk Harmonics
Device 1	288 MHz + 13.484 kHz	-0.568	102.249
Device 2	288 MHz + 17.215 kHz	-0.487	128.558
Device 3	288 MHz + 13.835 kHz	-0.561	104.497

Table 3. Standard Deviation for the extracted features in three same-type devices used in our setup.

Device	Clk Harmonic Freq	Autocorr	Strength Ratio Clk Harmonics
Device 1	59.442 Hz	0.00139	29.823
Device 2	46.580 Hz	0.00106	95.355
Device 3	81.848 Hz	0.00315	32.306

our deep learning method, we feed the spectrograms corresponding to the two different bands (see the end of Section 4.1) as input to a neural network. The output of the neural network is the fingerprinting output. The architecture of the neural network used in our deep learning approach is presented later in this section.

4.3 Feature Robustness

Before explaining the design of Digitus' classifiers, we present our study on the robustness of the selected features using various metrics. Features need to be unique to a specific device and consistent over time and different setups in order to be effective for fingerprinting.

The first metric is ***time***. Features need to be consistent across time and this can be across times of day, across different days, and across different times of operation. We explore these factors throughout the remainder of this section. Second, features need to be consistent across ***different setups*** and ***environments***. For instance, Digitus should be able to fingerprint a device in a line-of-sight scenario and a non-line-of-sight scenario. Digitus should perform well when the board is closer to the receiver or farther away from the receiver. We explore these tests in our sensitivity analyses in §6.

To demonstrate these features are unique, we apply statistical measures such as mean and standard deviation to the three devices in our setup (Arduino Unos). We desire for the means of different features for each device to be separated sufficiently from other devices' features and that the spread of data we see (measured by standard deviation) to be sufficiently small. A few of the results are shown in Table 2 and Table 3.

For many features, we see a substantially different mean and standard deviation for each device in relation to the means and standard deviations of other devices. Some features such as autocorrelation and the clock harmonic frequency achieve greater separation, whereas others such as the strength ratio between clock harmonics achieve less separation. This does not mean this feature is not useful, but rather there is more variance observed across different samples and this feature needs to be utilized with other features. Note that although the autocorrelation values are negative, the autocorrelation for a specific device is unique and consistent over time.

We examine the effects of long-term usage on two devices of different types over the course of one month while being powered on the entire time. We measure the clock frequency and the temperature throughout the day (morning, afternoon, and evening) in order to capture day-to-day variations in clock drift and temperature. In addition to these two devices we test over a month, we also test the remaining devices for a single week for completeness in order to compare.

Table 4. Standard deviations of clock frequency and temperature over a month for two devices (one of each type), as well as over a week for the remaining devices. The devices are stable relative to drifts seen in our measurements.

	One Month		**One Week**	
Device	**Clock Frequency**	**Temperature**	**Clock Frequency**	**Temperature**
Device Type A	58.2 Hz	0.52 °C	42.8 Hz	0.46 °C
Device Type B	116 Hz	0.50 °C	142 Hz	0.55 °C

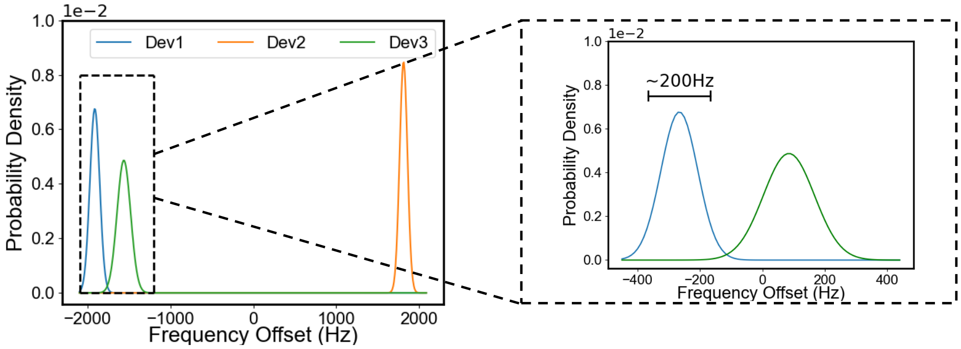


Fig. 6. This figure shows the normal distribution of measured center clock frequencies measured for Dev1, Dev2, and Dev3 across our set of measurements. Dev1 and Dev3 are close to each other in frequency, whereas Dev2 is at the opposite end of the band. The right side is a zoomed-in version of Dev1 and Dev 3. The expected frequency drift of ~200Hz fits within the normal distribution of each device's clock frequency.

Table 4 shows the standard deviation of clock frequency and temperature for both device types across one week and one month. We see that the clock frequency and temperature are both relatively stable when the device is plugged in for an extended time period. Additionally, there was a negligible change in clock frequency from the first day to the last day. We further examine the effects of long-term usage in Section 6.

To demonstrate the clock frequency measurements for each device are separated in the spectrum, we measure the center clock frequency of each device repeatedly to obtain a distribution of each device's clock. The reason we obtain a distribution is due to each clock's tendency to drift over time. Fig. 6 showcases the clock frequency distributions for three devices, with the graph on the right side being a zoomed-in version of the dotted region on the left side. For Dev 1 (Device Type A, one month), the expected drift of clock frequency (~200Hz) is within the normal distribution of observed clock frequencies during our measurements. This is true for the other devices in our experiments as well.

4.4 Machine Learning Models for Classification and Feature Extraction

The last step in Digitus is to classify the device and make decisions about its ID and mode. The ID reveals which device this is, where the numbers could be anything between 0 to N, where N represents all the devices *seen* during the training. Alternatively, the device may not be part of the training set (e.g., a new device that is added to the set). This is explored in Section 7. The other output of the classifier is the device's *mode* (e.g., whether it is transmitting or idle). We will also explain why this would be needed in Section 7.

To achieve the best accuracy, we implement seven different classifiers: *Logistic Regression, Support-Vector Machine, K-Nearest Neighbors, Decision Tree, Extra Trees, Random Forest, and (deep) Neural Network.* Among the

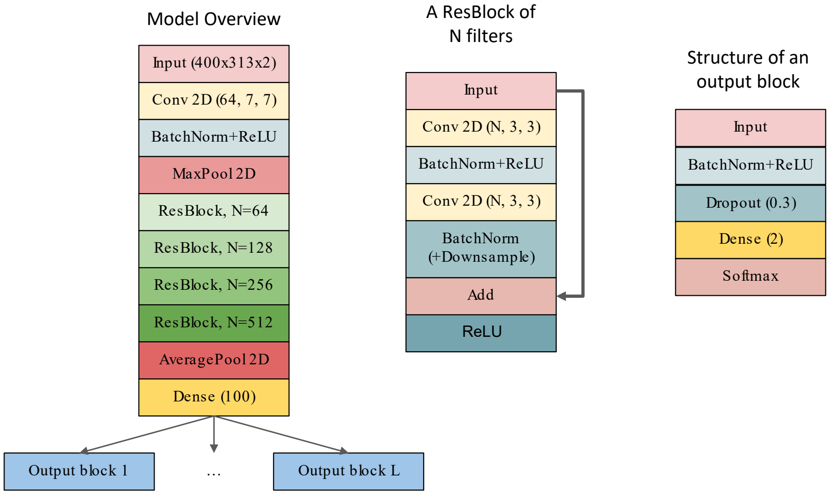


Fig. 7. The architecture of the neural network used in Digitus. The input is the pre-processed EM signals (not the features). The number of output blocks, L, varies according to the task and will be specified for different experiments in Section 7.

non-neural network models, Random Forest (max depth = 15 and the number of trees = 100) performs the best and is used throughout the evaluations in the next sections.

We will also compare the performance of the Random Forest classifier with our customized (deep) neural network classifier in Section 7.

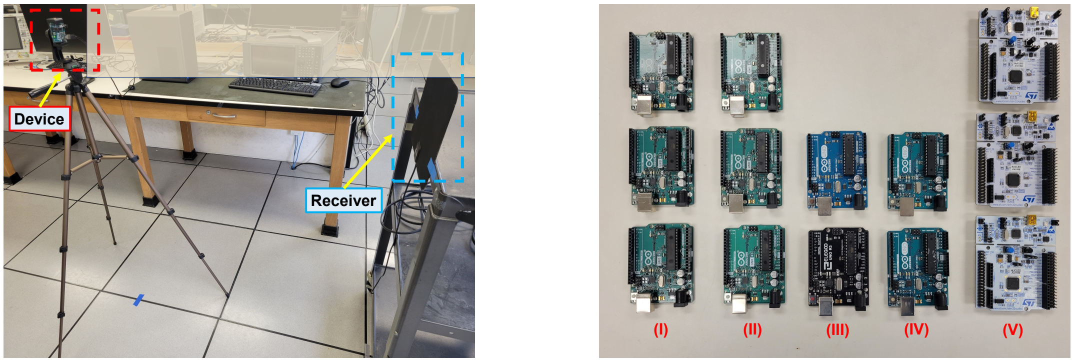
The architecture of the neural network that we use in Digitus is shown in Fig. 7. We use a convolutional neural network (CNN) architecture as the input to the neural network is an image and CNNs are known to work well with images by exploiting their spatial correlation. We use a ResNet-like architecture to train a deep model without overfitting. For pre-processing of the input to our deep learning approach, we use n = 0.5 seconds (capture length of an example) from a single band, and we take 400 points FFT on the smaller time slices. Then for each of two consecutive smaller time slices, we retain the maximum value per FFT bin. This way, we end up with 313 smaller time slices (our sampling rate is 500 kHz) and we get the input of our neural network to be 400 × 313 × 2. The last dimension, i.e., 2, is due to the use of two bands. (Note that the deep learning module takes the preprocessed samples as the input and not the features explained before.)

For each test, to minimize overfitting, we make sure to obtain data across different times, days, and environments to diversify our training set. We also use grid search on data separate from the test data to tune each model's hyperparameters to the best of our ability.

It is important to highlight that we envision the classifier (i.e., ML or deep learning) being implemented on a gateway where one device controls and monitors all devices in a room/space. As a result, the target devices (i.e., devices that are to be fingerprinted) won't be impacted by the choice of the classifier as it is external to each.

5 EVALUATION SETUP

To test our feature-based fingerprinting approach, we design a wide range of experiments to gauge performance. Fingerprinting needs to be tested in several different setups and environments in order to showcase the robustness of our features.



(a)

(b)

Fig. 8. (a) This is the baseline test experimental setup. The distance between the device and the receiver is 1m. (b) These are the devices used in Digitus. Columns (I) and (V) are used in the baseline test and sensitivity analyses. Columns (I-IV) are used in the intra-device test.

The various measurements are conducted inside a lab room in a building within a metropolitan area with a high chance of interference. No adjustments or optimizations are made to increase the signal strength of the device being tested in order to realistically demonstrate the feasibility of our approach.

The basic setup is shown in Fig. 8a. The device under test is attached to a tripod. The receiver's antenna is attached to a moveable cart placed away from the test device. This setup allows for easy reconfiguration for different experiments. The antenna used is a COTS VHF/UHF indoor TV antenna [4], which is connected to a USRP B205mini-i software-defined radio [7].

The devices we use are two commonly used microcontrollers: Arduino Uno [51] and STM 32 [1]. ***We specifically choose these devices since they don't have radios and/or DRAM; thus, none of the existing methods would be able to fingerprint them from range (i.e., a meter away or more).*** The collection of boards utilized in the experiments is shown in Fig. 8b. Both types of devices have a 16MHz clock on board. No changes were made to the hardware for each device.

Each device is measured in both an "idle" state where the board is in power-saving mode (i.e., it is on but does not actively execute any instructions) as well as an "active" state where the board is running a program.

For our data collection, we use 288MHz and 304MHz as harmonic frequencies, with a bandwidth of 500kHz. 288MHz and 304MHz are derived based on the base clock frequency of 16MHz (18th and 19th harmonic, respectively) as well as the relative strength of the clock and minimal ambient signal in these bands. 500kHz bandwidth is experimentally determined as sufficient bandwidth to capture the relevant signals.

6 RESULTS

The goal of this section is to demonstrate the robustness of Digitus in different realistic scenarios [2]. We hope to answer these basic questions:

(1) **Baseline Test.** Are emanations created by a device's processor clock indeed unique and useful across multiple similar (same type) and different devices?
(2) **Temporal Consistency.** Is Digitus stable across time? By measuring fingerprints at different times of the day, we should be able to showcase each device's stability across different measurements as well as robustness to changes in the channel.

[2] The code and data for this section are available online (See Section 10).

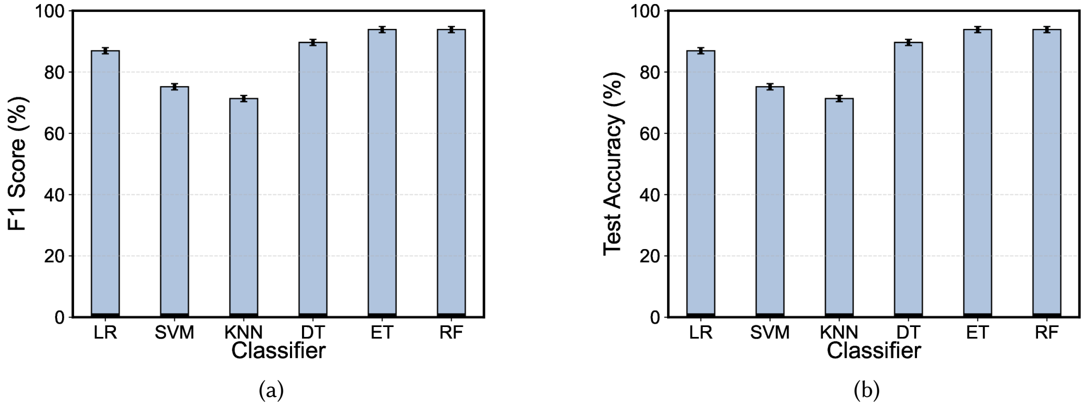


Fig. 9. Comparison of the F1 score (a) and test accuracy (b) for 6 different feature-based classifiers (Logistic Regression, Support-Vector Machine, K-Nearest Neighbors, Decision Tree, Extra Trees, and Random Forest) on the baseline test (3x Arduino Unos, 3x STM 32's, distance of 1m). Random Forest has the highest F1 score and test accuracy.

(3) **Clock Drift and Temperature Effects.** Clock drift and temperature are prominent factors that affect the operation of a device. In this section, we demonstrate the effects of clock drift and temperature on Digitus.
(4) **Spatial Consistency.** Is Digitus stable across different setups? Digitus should work at longer distances (i.e., attenuation) and even in non-line-of-sight scenarios. In this way, we hope to demonstrate Digitus's applicability to real-world scenarios.
(5) **Scalability Within Device.** Is Digitus applicable to a larger set of same-type devices? In a fingerprint dataset, there may be multiples of the same-type devices (we call it an intra-device test). By testing ten versions of the same device (Arduino Uno) in one set, we hope to show that Digitus can distinguish between all of them.
(6) **Spoofing Resistance.** In order for Digitus to provide reliable authentication, we need to demonstrate resistance to spoofing. Adversaries may try to mimic the fingerprints of the devices. Thus, we use our best judgment to spoof our devices and see how our model performs.

The following tests are independent of each other (the percentages in Table 5 are not additive). However, these experiments overlap in coverage. For example, the Attenuation and Non-Line-of-Sight measurements both measure reduced signal strength. Thus, we would expect the worst-case test accuracy to be around the worst-performing scenario outside of spoofing (Attenuation 7.5m has a test accuracy of 89.6%).

We also provide the deep learning results alongside the feature-based results. Additionally, we will discuss the two approaches further in Section 8.

6.1 Baseline Test

For the baseline test, we test three Arduino Unos and three STM 32s. The three Arduinos come from the same shipment, as well as the STM 32s. The distance between each device and the receiver is 1m. *The goal of this test is to analyze whether the features from the processor's clock are indeed unique and useful.*

A total of five rounds of data are taken. Three rounds of data are from one day (one in the morning, one at noon, and one in the afternoon), whereas the other two rounds are from separate days separated by *more than one month* from the former rounds. For each round, we make sure to turn on our devices sufficiently long enough to stabilize and be within the distribution found in Fig. 6. We combine all these rounds of data and divide into a train-test split. For the baseline test, we use feature engineering (see Section 4.2) and six different classifiers.

Table 5. Baseline Test experimental metrics as well as sensitivity experiments beyond the baseline test (temporal, long-term, temperature, attenuation 3m, attenuation 7.5m, non-line-of-sight, intra-device, and spoofing). Test accuracy is displayed for both our feature-based and deep learning models for each scenario.

	Feature-Based	**Deep Learning**
Experiment	**Test Accuracy**	**Test Accuracy**
Baseline Test	95.1%	97.5%
Temporal Test	91.5%	94.6%
Long-Term Test	96.3%	95.6%
Temperature	90.2%	91.2%
Attenuation (3m)	99.4%	93.9%
Attenuation (7.5m)	89.6%	80.1%
Non-Line-of-Sight	92.7%	94.8%
Intra-Device	96.1%	97.0%
Spoofing HackRF	96.3%	88.1%
Spoofing USRP	80.5%	87.2%
Spoofing (w/ training)	96.0%	97.2%

Results are shown in Fig. 9, which shows the F1 scores and test accuracies achieved for these classifiers. The best test accuracy achieved is 95.1% from the Random Forest classifier. To provide more insight, in Fig. 10a, we depict the confusion matrix for the Random Forest classifier. We see that the classifier has the most difficulty with stm1 and stm3 devices. This makes sense as stm1 and stm3's features are visually similar to each other. *This test demonstrates that the features are unique and useful and fingerprints measured are consistent across different times of day and across various devices.*

6.2 Temporal Test

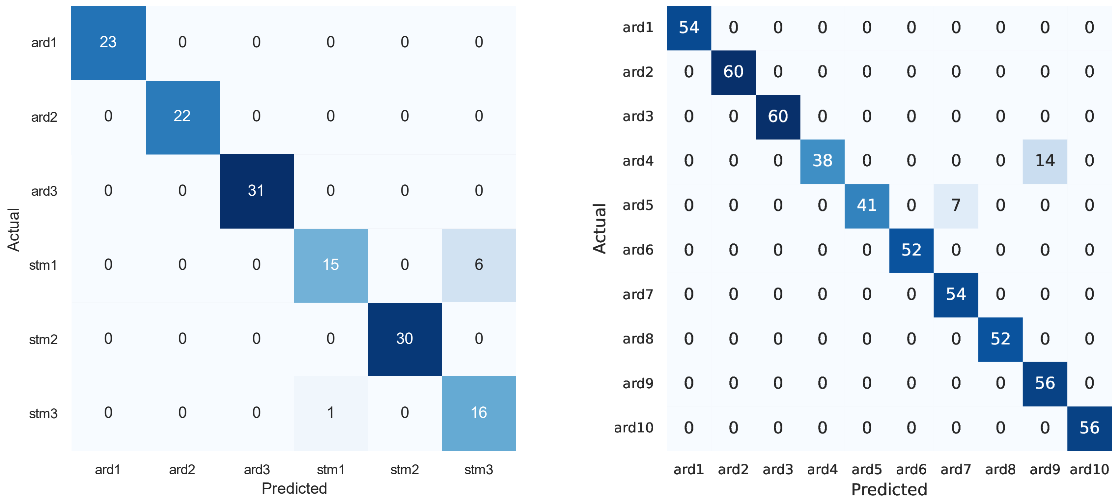
We further test our features' *temporal consistency*. For this test, we analyze the same three Arduino Unos and three STM 32s as before. Using the data from the baseline test in 6.1 as the training set, we obtain another set of data from **a completely separate day** as the test set to see how our model performs (i.e., unlike the first test, the training set does not contain any samples from this new day/measurement). Note that, ***this is in contrast with the state-of-the-art.*** To the best of our knowledge, while Memscope [47] conducted measurements across a month to test temporal consistency, they did not attempt to completely separate rounds of data for test purposes. We, however, argue that this should have been considered since it is a more realistic (training happens in a completely different time frame), but harder problem.

For the feature-based method, *for this test and all further experiments* (including sensitivity analyses and applications), we use the baseline test as training data and use the data collected for the given experiment as the test set in order to demonstrate Digitus' ability to generalize.

In Table 5, we see that the test accuracy is 91.5%. While the test accuracy has decreased from the baseline test, our model is still able to maintain a reasonable level of accuracy, thus showing *its ability to generalize to new data recorded at a completely different time.*

6.3 Long-Term Study

For the long-term study, we classify the two devices that were tested over a month in Section 4. The two devices picked were representative of the distribution of devices (Arduino 1 and STM 1). We collect data at the beginning,



(a) (b)

Fig. 10. (a) Confusion matrix for a random forest classifier with data from 3 Arduino Unos and 3 STM 32s, with a distance of 1m. The numbers represent the number of examples per bin. (b) Confusion matrix for a random forest classifier with data from ten Arduino Unos, with a distance of 1m.

middle, and end of the month. The goal of this test is to see the effect of clock drift from long-term usages on Digitus.

In Table 5, the test accuracy for the long-term study is 96.3%. Additionally, there is no decrease in accuracy when comparing the model's test accuracy on data from the end of the month compared to data from the beginning of the month. This demonstrates our model's robustness to clock drift seen in long-term usages.

6.4 Temperature Effects

For the temperature test, we expose our devices to cold and hot environmental temperatures (~10 degrees Celsius delta from room temperature in both directions) to examine the effect of temperature deviations on classification accuracy. By measuring the internal temperature of each device, we see the environmental temperature change induces a 5-10 degrees Celsius internal temperature change in the positive and negative direction for each device. In addition, this temperature difference causes clock drift to the system as well. In this way, we evaluate Digitus's robustness towards realistic temperature changes in the surrounding environment as well as further amounts of clock drift, which are all forms of temporal variation.

In Table 5, the resulting test accuracy is 90.2% (with similar results when hot and cold data are separate test sets), demonstrating that our model can withstand temperature variations and further clock drift changes.

6.5 Attenuation Test

For the attenuation test, we test the same three Arduino Unos and three STM 32s, but at two new distances on a different day: 3m and 7.5m (recall that the training set is still the baseline test setup). The 7.5m setup is the maximum distance possible in the lab. The setup is shown in Fig. 11a and Fig. 11b. The goal of this test is

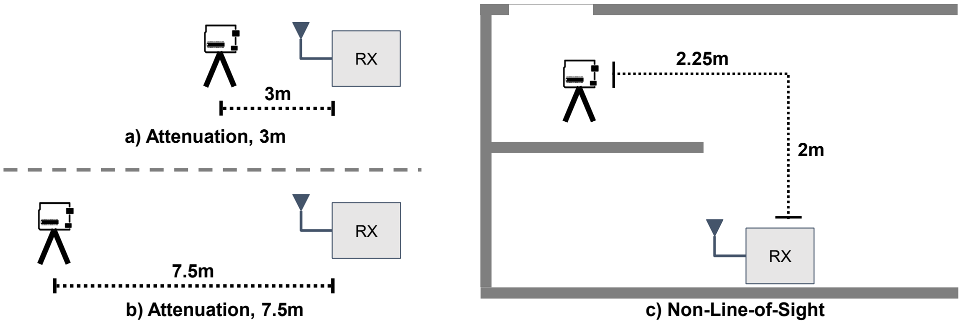


Fig. 11. (a): This is the attenuation test where the device is 3m away from the receiver. (b): This is the attenuation test where the device is 7.5m away from the receiver. (c): This is the non-line-of-sight test, where the device and receiver are separated by a point-to-point distance of around 3m.

to demonstrate the system's robustness to longer distances and increased levels of attenuation, even when the model is trained on a different setup (i.e., 1m and different days).

In Table 5, we see that at 3m, the test accuracy is 99.4%, whereas, at 7.5m, the test accuracy is 89.6%. *This demonstrates that the model performs comparatively better at closer distances, but still maintains reasonable performance even as the device is farther away*, a scenario one would find in a "smart room" IoT scenario as devices may find themselves in different positions in a room. Note that the accuracy at 3m is actually greater than at 1m. This indicates the uncertain effect of the channel and signal propagation paths on the strength of the emanations. A general rule of thumb, however, is the strength of the emanations will decrease as distance and/or attenuation levels increase.

6.6 Non-Line-of-Sight

We further examine the *spatial consistency* of Digitus by using the same three Arduino Unos and three STM 32s as before but this time in a non-line-of-sight (NLOS) setting. The point-to-point distance between the device and the receiver is 3m, and the sides of the triangle are 2.25m and 2m, as shown in Fig. 11c.

In Table 5, the test accuracy is 92.7%. Similar to the attenuation test, we see that *Digitus maintains reasonable performance even under changes of setup and environment.* In a real-world scenario, boards may not be directly facing the receiver or may be separated by a wall. Thus, it is important that we can maintain accuracy when these scenarios arise.

6.7 Intra-Device Test

For the intra-device (i.e., same-type devices) test, we evaluate ten Arduino Unos. The first three Arduino Unos are the same as the previous tests. Arduinos 4-6 come from another separate shipment. Arduino 7 is an individual board and did not come with any other Arduino Unos. Arduino 8 is from a different manufacturer. Arduinos 9-10 are from another separate shipment.

The distance between the device and the receiver is 1m. We collect two rounds of data on separate days and try two combinations of training and test sets using the two rounds of data. We compute our metrics as the average of the two combinations. The goal of this test is to scale up the number of boards, specifically of the same device type, to see if the system can distinguish between types well.

In Table 5, the test accuracy is 96.1%. Additionally, the confusion matrix in Fig. 10b shows that the model is able to distinguish between the different Arduinos, with slight difficulty with Arduino 4. *These results show that Digitus scales up with more devices of the same type.*

6.8 Device Spoofing

To model a spoofing attack and demonstrate spoofing resistance, we attempt to spoof our devices in the baseline test using our best judgment and effort and analyze how Digitus performs against these spoofed samples. The idea is that an adversary may try to mimic the fingerprints/signals of a known device. Thus, in order for Digitus to provide reliable authentication, we need to demonstrate resistance to spoofing.

To implement the attack, we use two software-defined radios: HackRF [17] and a USRP B205mini-i [7]. We use them to create RF signals similar to the expected EM emanations. The aim is to create frequency spikes at the clock frequencies of each board, as well as match the SNR and other characteristics as accurately as possible. The spoofing measurements were taken on completely separate days and times from the training data and had some feature variation. We repeat the experiment using both radios.

To enable detection of spoofing, we add the notion of *classification confidence* to Digitus, where a threshold, T, is used to identify a device, otherwise, it will be labeled as "unknown" or *spoofed*. We experimentally set the threshold to optimize the relationship between false positives and false negatives. We find via the validation set that $T = 85\%$ provides a good balance between fingerprinting accuracy and spoofing detection.

In Table 5, we see the test accuracies for thresholding on the HackRF data (96.3%) and thresholding on the USRP data (80.5%). *This demonstrates that Digitus can adequately handle spoofing and performs differently depending on the quality of the spoofer.* We also conduct an experiment where we split the spoofed samples (both HackRF and USRP) into a train and test split, where we add an additional class called "other". This results in a test accuracy of 96.0%, showing the value of training on spoofed samples to prepare our model for potential spoofers.

6.9 Performance of Deep Learning Approach

Along with our feature-based machine learning method, we also evaluate our deep learning approach for each of the experiments in Table 5. We observe that, in general, the performances for both of our approaches are comparable. However, for the attenuation experiments, the deep learning approach is slightly inferior to the feature-based method. We believe this is due to the fact that the signals collected in the attenuation experiments have a much lower signal-to-noise ratio. This affects the deep learning approach as it has difficulty differentiating between targeted signals and environmental interference and noise.

7 APPLICATIONS AND USECASES

7.1 Room Monitoring with Hidden Device Detection

The first scenario presented and evaluated here is leveraging Digitus to ensure privacy. Particularly, Digitus can be used to identify devices that are not part of the *authorized* list of devices that could operate in a room. Examples are "hidden" devices in a room such as a camera, voice recorder, etc. where they may or may not have a radio and their goal is to eavesdrop and/or steal private information.

While such detection capabilities has been explored in prior work [12, 14, 29, 40, 45, 46], the **key advantage** of our approach compared to the state-of-the-art is that we can detect a wider range of devices including tiny embedded devices that don't have radios and/or DRAM.

We outline the experimental setup in Fig. 12. In this experiment, we assume there are multiple authorized devices already present in the room (two devices in our experiment). A third device (i.e., the hidden device) not in the training set is activated in the room in idle mode (within the red dashed box in the figure, the target device). We test a total of five different configurations of the devices physically in the room to provide a more extensive evaluation. We then repeat the experiment this time with the third device in the training set (i.e., authorized). The first scenario is called *open set* while the second is *closed set*.

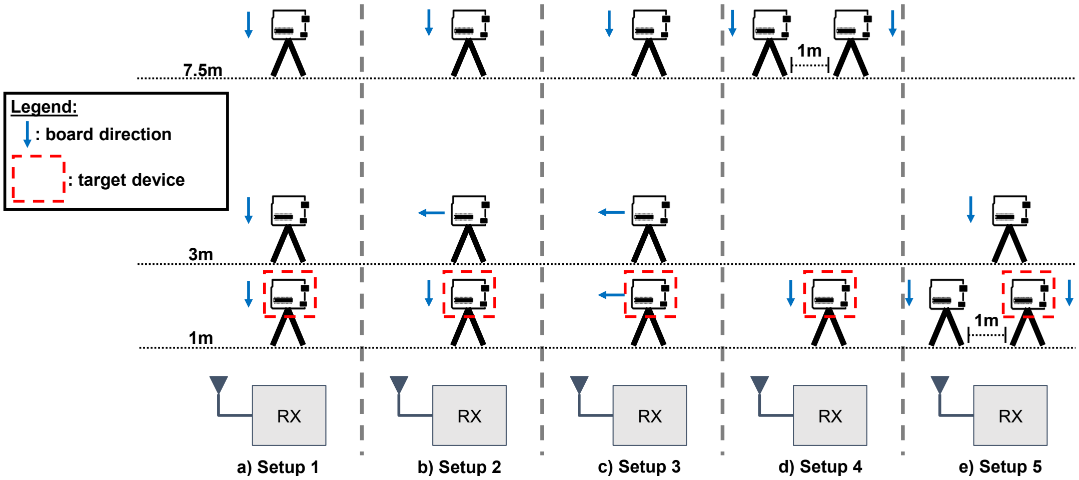


Fig. 12. This figure shows five different setups for the hidden device test and the determining device state test. The target device is surrounded by a red dashed box. The direction of the board in relation to the receiver is shown by the blue arrows (a horizontal blue arrow means the board is facing perpendicular to the receiver). There are three distances: 1m, 3m, and 7.5m. Additionally, in Setup 4 and Setup 5, the devices at the same distance are separated by 1m.

Table 6. This table shows results for the various applications and usecases: hidden devices and device state. For each experiment, we provide a description of the test as well as the feature-based (FB) test accuracy and the deep learning (DL) test accuracy. *Closed set* refers to a scenario where the target device was seen in the training set, while *open set* is the opposite.

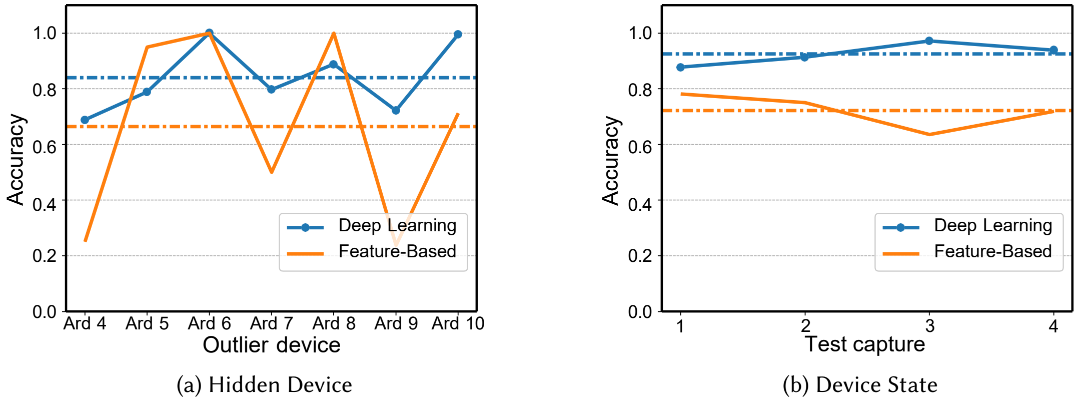
Experiment	Description	FB Test Accuracy	DL Test Accuracy
Hidden Devices	Closed Set	99.0%	98.9%
	Open Set	66.0%	84.0%
Device State	Singular Device	72.1%	92.5%
	Multiple Devices, Target Device	80.0%	≥99.9%

The goal for Digitus is to detect the presence of a new device AND to correctly identify *(1)* the target device as outside the set in the first scenario (true positive) OR *(2)* the target device as inside (i.e., authorized) in the second scenario (true negative). In both cases, two in-set (authorized) devices are present in the room.

We test seven combinations of devices across ten Arduinos. Arduinos 1 and 2 are always in the set, in the background of the room in idle mode. We obtain captures of each of Arduinos 4-10 as the target device. For our testing data, we take one of Arduinos 4-10 as out-of-the-set whereas the remaining Arduinos are deemed in the set. The training and validation data consist of only the devices in the set. We then repeat the experiment, this time while having all the devices in the training set in order to test true negatives.

Using our feature engineering method and a Random Forest classifier, we evaluate the accuracy of Digitus for this use case. The first step is to extend the algorithm described in Section 4.2 to extract multiple clocks and features. To achieve that, we utilize the same clock search, except we search for multiple clocks.

Once features are extracted, Digitus uses a Random Forest classifier to fingerprint devices. We use the validation data to select a threshold, where the threshold is defined as the lowest prediction probability of a correct prediction. When testing, any prediction that has a probability less than this threshold is deemed an outlier (i.e., a hidden device). We obtain an average accuracy of 83.0% (true positive of 66% and true negative of 99%) across all seven



(a) Hidden Device

(b) Device State

Fig. 13. (a) This plot shows the outlier (hidden) detection rate on different Arduinos with the two approaches of Digitus, where the dashed lines show the average for each. (b) This plot shows state classification accuracy across different captures (data set) with the two approaches of Digitus.

combinations of devices. Looking deeper into the individual test accuracies, we obtain 100% or near 100% test accuracies for Arduino 5, 6, and 8, but around 62% test accuracies for Arduino 4 and 9. A possible explanation for this varying performance is that Arduino 4 and 9's fingerprints appear more alike to existing fingerprints, whereas Arduino 5, 6, and 8 have more distinct fingerprints.

While the handcrafted features can achieve high accuracy in the true negative classification task, they encounter challenges in detecting unseen devices or outliers. Hence, for this task, we test Digitus with our deep learning model that can extract potentially more complicated yet subtle features from the signals without manual effort.

For the evaluation of the deep learning approach, we use the same sets of training and testing data, and also the same strategy to determine the threshold for rejecting an outlier device as done for the feature-based approach. As the model will learn from 7 and 6 known (authenticated) devices, for closed set and open set tasks, respectively, we have correspondingly $L = 7$ and $L = 6$ outputs from the model (refer to Fig. 7). Each output o_i predicts whether the input signal corresponds to device i or not. Then, for the decision, the model declares the device label to be j if the prediction confidence of o_j is maximum among all the outputs.

The deep learning approach obtains an average accuracy of 91.5% (true positive of 84.0% and true negative of 98.9%) across all seven combinations of devices. It can be observed that while the deep learning approach yields a true negative rate comparable to the feature-based approach, *it improves the true positive rate (outlier detection rate) significantly* comparable to the feature-based approach. A more detailed result is shown in Fig. 13 (a).

7.2 Determining Device State for Fingerprinting Transmitters

Another important application of Digitus is to determine a device's state or mode. There are a few interesting use cases for this. For instance, one may like to identify if a target device is actively completing a task whereas the other devices are idle in order to authenticate the target device. The other is to leverage EM emanations for fingerprinting devices with radios. The key idea is that the existing infrastructure for Digitus can be *reused* even for devices with radios and eliminate the need for an additional RF fingerprinting module. RF fingerprinting methods are often dependent on the characteristics of the RF signals (e.g., signaling protocol, modulation type, etc.) that are being transmitted. Digitus is a robust alternative to such RF fingerprinting approaches by virtue of being agnostic to the RF signals. For this test, we assume that the Arduino devices are using a LoRA module connected via I2C as the radio (Semtech SX1262 [50]).

To study this, first, we hope to demonstrate Digitus's ability to identify a singular device's state (i.e., idle vs. transmit/program) without the presence of other devices in the room. Using the data from Section 6.7, we attempt to identify both the device ID and the device state. We test four combinations of data and take an average for the test accuracy. For our feature-based approach, we obtain an average of 72.1% test accuracy (across both device ID and device state), with a standard deviation of 5.4%. For the deep learning approach, we have $L = 7 + 2 = 9$ outputs, where each output corresponds to a device ID (Ard 4-10) or a device state (idle or program), with a total of 9 output labels. Then, for the decision, the model selects two outputs, one for the device label and one for the state label. The model declares the device label to be j if it has the maximum confidence in the output among all the device outputs and the state k that has the maximum output confidence among all the state outputs. With the deep learning approach, we obtain an average accuracy of 92.5% with a standard deviation of 3.4%, when both ID and state are classified correctly. A more detailed result is shown in Fig. 13 (b).

A reason why deep learning could perform better than feature-based is that features for idle compared to program may look similar in a feature-based approach. In contrast, a deep learning approach could learn useful characteristics from different parts of the spectrum that our feature-based model cannot capture.

Next, we test the case where ***multiple*** devices are in the room at the same time. We utilize the experimental setup shown in Fig. 12. Arduino 1 and 2 are in the background in idle mode, and Arduino 3 is always the test device in transmit/program mode. Digitus needs to correctly identify the state of Arduino 3 with the presence of the interferences from Arduino 1 and 2. For training, we use the individual captures from Arduino 1 (idle), 2 (idle), and 3 (idle and program). Moreover, we augment our training dataset with one capture in the presence of real interfered signals for all the setups shown in Fig. 12. Then, we test on another unseen capture for the same setups in Fig. 12.

In this task, Digitus can correctly identify the state of the target device 80% and $\geq$ 99.9% of the time with the feature-based approach and the deep learning approach, respectively. In this case, the deep learning model has $L = 2$ outputs, corresponding to the idle and program states of Arduino 3, and the model will select the state with the higher confidence in the output.

We observe that our feature-based model often mispredicts the target device to be in idle mode. This is because the difference in features between idle and program for a specific device is small, whereas the deep learning model can learn those differences more accurately. For the surrounding idle devices, we note that the feature-based model has more trouble with the device that is further away from the receiver, which makes sense as the signal will be more attenuated. Generally speaking, our model is able to succeed in determining the device state of various devices that are together in the same room.

8 DISCUSSIONS

8.1 Feature-Based Versus Deep Learning

In Section 7, we demonstrate the usefulness of a feature-based approach as well as the merits of a deep learning approach. Feature-based is suitable for situations where we desire to fingerprint the device type, such as the results in Section 6 as well as closed set problems. Deep learning can thrive in scenarios where more complex information in the band is needed to separate between classifications, such as in open set problems or classification between device states. Consequently, the decision on which one to use is made dynamically based on the application. The computational resources available also need to be considered. If computational resources are available, such as in the cloud, deep learning may be a better option considering the training time and the complexity of the model. In scenarios where resources are limited and computation may need to be closer to the edge, feature-based may be better. We summarize the differences between feature-based compared to deep learning in Table 7. Despite

Table 7. Comparison between Feature-based (F.B.) and Deep Learning (D.L.) approaches. There exists a tradeoff between these two approaches that need to be considered when selecting the approach; however, both methods are viable options due to their similar performance in our different test cases.

Metric	F.B.	D.L.	Notes
Input Length	5 s	0.5 s	Deep Learning has more flexibility in terms of input length compared to feature-based.
Model Complexity	↓	↑	Feature-based is more efficient in terms of training and model size.
Environment Sensitivity	↓	↑	Feature-based is better suited to handle environmental changes. Deep learning requires more training data and/or retraining.
Simple Task	↑	↑	Both approaches perform comparably in the tasks shown in Section 6.
Complex Task	↔	↑	In more complex tasks, Deep Learning may perform better due to an ability to learn more complex information.

these differences, we believe that feature-based and deep-learning approaches are both viable options due to their comparable performance in many scenarios.

Future work could involve adding increasingly complex features to the feature-based approach or further systematizing the interaction between feature-based and deep learning.

8.2 Authentication Protocol

For Digitus to be effective, an authentication protocol needs to be in place. One potential method is for a device to signal they are ready to be authenticated by running a particular program. This assumption holds because the majority of devices should be in an idle state most of the time in an IoT scenario. In Section 7, we demonstrated Digitus' ability to identify a target device's ID and its program state in the presence of other, idle devices. In this way, the receiver can see the device is ready to be authenticated and the receiver can begin the fingerprinting process. If two or more devices are active at the same time, we can attempt to fingerprint all of them at once or can authenticate via a round-robin access protocol, an implementation left to future work.

A second potential method is to authenticate a device while it is sending data to the gateway. As the device will unintentionally emanate during transmission, the most efficient use of fingerprinting would be to authenticate the device during the transmission of data and bypass the need for the device to run an authentication program. In this way, the target device would be active for less time, increasing efficiency. We leave the actual implementation to future work.

Embedded IoT devices may be inactive for large periods of times or in low-power states. To obtain the fingerprint of a device in these scenarios, Digitus periodically scans expected frequency ranges of various device clocks to obtain the fingerprint of the device when the device is "on". Once this fingerprint is obtained, Digitus can handle scenarios where the device is off (the device cannot transmit in the off state, thus there is no need to fingerprint), as well as the "on" state (the fingerprint had been collected already). Additionally, in Section 6, we demonstrate Digitus's ability to classify devices in sleep mode.

8.3 Program-Agnostic Fingerprinting

To be more generic, Digitus needs to be effective when different types of programs are active on different devices. We attempted to capture some of those differences with our "program" and "idle" states and Digitus was able to accurately classify each device's mode (especially when using deep learning). In future work, we need to be able to handle different programs. Each device may have different tasks and our fingerprinting system should successfully authenticate the device no matter which state it is in. Additionally, it is possible that while we are fingerprinting, the target device may even switch between states (either between different programs or between program and idle). We need to guarantee that our system can handle these transitions as the fingerprinter cannot control the device state. Thus, a more robust analysis of the effect of different programs and the relationship between states should be studied.

9 RELATED WORK

Fingerprinting Near Field. Much existing work focuses on fingerprinting devices via their side-channels at near-field ranges [2, 53]. In EM-ID [53], the authors utilize a near-field antenna to measure emanations in order to replace RFID. A cosine similarity function is used to identify devices. Additional machine learning techniques can also be used to fingerprint. For example, sparse bayesian linear regression can be used to identify both the device and the software on the device at near-field ranges [31]. Feature-based approaches have been developed to identify devices such as laptops, phones, and USB flash drives [15, 23]. A neural network based industrial control flow monitoring system called Zeus has also been developed using near-field emanations [19].

Compared to existing work, Digitus is beneficial because we are able to extend the authentication range.

Fingerprinting RF Devices at Range. Fingerprinting RF devices with transmitter modules is a well known field. Many works attempt to identify devices within wireless networks using unique characteristics of the transmitted signal caused by transmitter module imperfections [9, 22, 25, 32, 52]. An approach has been developed to handle changes in the region of interest as well as fingerprinting automation [54]. The impact of channel [3] as well as differences in modulation [20] on identification is also important. Work has also been done on features for RF devices. Features such as modulation offset and I/Q offset can be used [36]. For ZigBee devices, work has been done to select features using multiple discriminant analysis [8]. A framework of choosing features per authentication device has also been developed [38].

Compared to existing work, Digitus is wireless modulation agnostic and transmitter module independent.

Authentication and Security Applications. One application of authentication is hidden device detection. In EarFisher [46], the authors distinguish eavesdroppers from legitimate devices by listening to the natural emissions from device memory accesses. In Ghostbuster, it is shown that even passive receivers that are eavesdropping can be identified via RF signal leakage [12]. Hidden device detection can also be done via observation of packet flow and network traffic [14, 29, 40, 45]. Hidden device detection calls into consideration the question of open set versus closed set, and work has been done to detect devices not in the set such as malicious devices [21, 35]. Work has been done to generally secure a collection of IoT devices on a network from malicious adversaries [13, 30, 33, 48, 49]. Authencation can also happen between two devices utilizing the ambient electromagnetic signals in the room, as shown in AEROKEY [27]. In an adversarial setting, DroneTrace was developed in order to track and identify malicious drones [28]. Millimeter wave signals are utilized to pierce the drone case.

Compared to existing work, our work can be applied to offline IoT devices without active stimulation of the devices nor radio (packet transmission) and/or memory activities.

10 CONCLUSIONS

Digitus is a fingerprinting system that utilizes emanations from the processor's clock to characterize IoT devices and authenticate at range. This system removes the reliance on transmitter modules, the specific wireless modulation in the spectrum, and the need for complex memory, approaches utilized in previous work. We demonstrate through various sensitivity analyses the applicability of Digitus in a wide range of IoT scenarios. Additionally, we show Digitus's applicability to several IoT applications such as hidden device detection. Digitus is a new authentication method that can provide a low-overhead solution for many low-power IoT devices in the rapidly growing world of IoT.

ACKNOWLEDGEMENTS

We thank the reviewers for their comments and guidance on this work. This work has been supported, in part, by NSF grant 2211301 and IARPA grant PO-IARPA 2021-21062400004. The views and findings in this paper are those of the authors and do not necessarily reflect the views of NSF and IARPA.

REFERENCES

[1] STM 32. [n. d.]. https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html. Accessed: 2022-11.

[2] Mosabbah Mushir Ahmed, David Hely, Etienne Perret, Nicolas Barbot, Romain Siragusa, Maxime Bernier, and Fredric Garet. 2018. Authentication of microcontroller board using non-invasive em emission technique. In *2018 IEEE 3rd International Verification and Security Workshop (IVSW)*. IEEE, 25–30.

[3] Amani Al-Shawabka, Francesco Restuccia, Salvatore D'Oro, Tong Jian, Bruno Costa Rendon, Nasim Soltani, Jennifer Dy, Stratis Ioannidis, Kaushik Chowdhury, and Tommaso Melodia. 2020. Exposing the fingerprint: Dissecting the impact of the wireless channel on radio fingerprinting. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 646–655.

[4] Indoor TV Antenna. [n. d.]. https://www.amazon.com/dp/B01FUB4ZG8?psc=1&ref=ppx_yo2ov_dt_b_product_details. Accessed: 2022-11.

[5] Jerome Antoni. [n. d.]. https://www.mathworks.com/matlabcentral/fileexchange/48909-cyclic-spectral-analysis/. Accessed: 2022-11.

[6] Mahzad Azarmehr, Ankit Mehta, and Rashid Rashidzadeh. 2017. Wireless device identification using oscillator control voltage as RF fingerprint. In *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*. IEEE, 1–4.

[7] USPR B205mini-i. [n. d.]. https://www.ettus.com/all-products/usrp-b205mini-i/. Accessed: 2022-11.

[8] Trevor J Bihl, Kenneth W Bauer, and Michael A Temple. 2016. Feature selection for RF fingerprinting with multiple discriminant analysis and using ZigBee device emissions. *IEEE Transactions on Information Forensics and Security* 11, 8 (2016), 1862–1874.

[9] Vladimir Brik, Suman Banerjee, Marco Gruteser, and Sangho Oh. 2008. Wireless device identification with radiometric signatures. In *Proceedings of the 14th ACM international conference on Mobile computing and networking*. 116–127.

[10] Robert Callan, Alenka Zajić, and Milos Prvulovic. 2015. FASE: Finding amplitude-modulated side-channel emanations. In *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 592–603.

[11] Giovanni Camurati, Aurélien Francillon, and François-Xavier Standaert. [n. d.]. Understanding screaming channels: From a detailed analysis to improved attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems* ([n. d.]).

[12] Anadi Chaman, Jiaming Wang, Jiachen Sun, Haitham Hassanieh, and Romit Roy Choudhury. 2018. Ghostbuster: Detecting the presence of hidden eavesdroppers. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*. 337–351.

[13] Batyr Charyyev and Mehmet Hadi Gunes. 2020. Locality-sensitive iot network traffic fingerprinting for device identification. *IEEE Internet of Things Journal* 8, 3 (2020), 1272–1281.

[14] Yushi Cheng, Xiaoyu Ji, Tianyang Lu, and Wenyuan Xu. 2018. Dewicam: Detecting hidden wireless cameras via smartphones. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*. 1–13.

[15] Yushi Cheng, Xiaoyu Ji, Juchuan Zhang, Wenyuan Xu, and Yi-Chao Chen. 2019. Demicpu: Device fingerprinting with magnetic signals radiated by cpu. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 1149–1170.

[16] Shane S Clark, Benjamin Ransford, Amir Rahmati, Shane Guineau, Jacob Sorber, Wenyuan Xu, and Kevin Fu. 2013. {WattsUpDoc}: Power Side Channels to Nonintrusively Discover Untargeted Malware on Embedded Medical Devices. In *2013 USENIX Workshop on Health Information Technologies (HealthTech 13)*.

[17] Great Scott Gadgets. [n. d.]. https://greatscottgadgets.com/hackrf/. Accessed: 2022-11.

[18] Daniel Genkin, Lev Pachmanov, Itamar Pipman, and Eran Tromer. 2015. Stealing keys from PCs using a radio: Cheap electromagnetic attacks on windowed exponentiation. In *International workshop on cryptographic hardware and embedded systems*. Springer, 207–228.

[19] Yi Han, Sriharsha Etigowni, Hua Liu, Saman Zonouz, and Athina Petropulu. 2017. Watch me, but don't touch me! contactless control flow monitoring via electromagnetic emanations. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*. 1095–1108.

[20] Samer Hanna, Chris Dick, and Danijela Cabric. 2021. Signal Processing-Based Deep Learning for Blind Symbol Decoding and Modulation Classification. *IEEE Journal on Selected Areas in Communications* 40, 1 (2021), 82–96.

[21] Samer Hanna, Samurdhi Karunaratne, and Danijela Cabric. 2020. Open set wireless transmitter authorization: Deep learning approaches and dataset considerations. *IEEE Transactions on Cognitive Communications and Networking* 7, 1 (2020), 59–72.

[22] Samer S Hanna and Danijela Cabric. 2019. Deep learning based transmitter identification using power amplifier nonlinearity. In *2019 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 674–680.

[23] Omar Adel Ibrahim, Savio Sciancalepore, Gabriele Oligeri, and Roberto Di Pietro. 2020. MAGNETO: Fingerprinting USB Flash Drives via Unintentional Magnetic Emissions. *ACM Transactions on Embedded Computing Systems (TECS)* 20, 1 (2020), 1–26.

[24] Tong Jian, Yifan Gong, Zheng Zhan, Runbin Shi, Nasim Soltani, Zifeng Wang, Jennifer Dy, Kaushik Chowdhury, Yanzhi Wang, and Stratis Ioannidis. 2021. Radio frequency fingerprinting on the edge. *IEEE Transactions on Mobile Computing* 21, 11 (2021), 4078–4093.

[25] Tong Jian, Bruno Costa Rendon, Emmanuel Ojuba, Nasim Soltani, Zifeng Wang, Kunal Sankhe, Andrey Gritsenko, Jennifer Dy, Kaushik Chowdhury, and Stratis Ioannidis. 2020. Deep learning for RF fingerprinting: A massive experimental study. *IEEE Internet of Things Magazine* 3, 1 (2020), 50–57.

[26] Kyouwoong Kim, Ihsan A Akbar, Kyung K Bae, Jung-Sun Um, Chad M Spooner, and Jeffrey H Reed. 2007. Cyclostationary approaches to signal detection and classification in cognitive radio. In *2007 2nd ieee international symposium on new frontiers in dynamic spectrum access networks*. IEEE, 212–215.

[27] Kyuin Lee, Yucheng Yang, Omkar Prabhune, Aishwarya Lekshmi Chithra, Jack West, Kassem Fawaz, Neil Klingensmith, Suman Banerjee, and Younghyun Kim. 2022. AEROKEY: Using Ambient Electromagnetic Radiation for Secure and Usable Wireless Device Authentication. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 1 (2022), 1–29.

[28] Zhengxiong Li, Baicheng Chen, Xingyu Chen, Chenhan Xu, Yuyang Chen, Feng Lin, Changzhi Li, Karthik Dantu, Kui Ren, and Wenyao Xu. 2022. Reliable Digital Forensics in the Air: Exploring an RF-based Drone Identification System. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 2 (2022), 1–25.

[29] Tian Liu, Ziyu Liu, Jun Huang, Rui Tan, and Zhen Tan. 2018. Detecting wireless spy cameras via stimulating and probing. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*. 243–255.

[30] Eman Maali, David Boyle, and Hamed Haddadi. 2020. Towards identifying IoT traffic anomalies on the home gateway. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*. 735–736.

[31] Laura J Mariano, Alexander Aubuchon, Troy Lau, Onur Ozdemir, Tomo Lazovich, and John Coakley. 2019. Classification of electronic devices and software processes via unintentional electronic emissions with neural decoding algorithms. *IEEE Transactions on Electromagnetic Compatibility* 62, 2 (2019), 470–477.

[32] Kevin Merchant, Shauna Revay, George Stantchev, and Bryan Nousain. 2018. Deep learning for RF device fingerprinting in cognitive communication networks. *IEEE Journal of Selected Topics in Signal Processing* 12, 1 (2018), 160–167.

[33] Markus Miettinen, Samuel Marchal, Ibbad Hafeez, N Asokan, Ahmad-Reza Sadeghi, and Sasu Tarkoma. 2017. Iot sentinel: Automated device-type identification for security enforcement in iot. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2177–2184.

[34] Alireza Nazari, Nader Sehatbakhsh, Monjur Alam, Alenka Zajic, and Milos Prvulovic. 2017. Eddie: Em-based detection of deviations in program execution. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*. 333–346.

[35] Jorge Ortiz, Catherine Crawford, and Franck Le. 2019. DeviceMien: network device behavior modeling for identifying unknown IoT devices. In *Proceedings of the International Conference on Internet of Things Design and Implementation*. 106–117.

[36] Linning Peng, Aiqun Hu, Junqing Zhang, Yu Jiang, Jiabao Yu, and Yan Yan. 2018. Design of a hybrid RF fingerprint extraction and device classification scheme. *IEEE Internet of Things Journal* 6, 1 (2018), 349–360.

[37] Roberto Perdisci, Thomas Papastergiou, Omar Alrawi, and Manos Antonakakis. 2020. Iotfinder: Efficient large-scale identification of iot devices via passive dns traffic analysis. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 474–489.

[38] Donald Reising, Joseph Cancelleri, T Daniel Loveless, Farah Kandah, and Anthony Skjellum. 2020. Radio identity verification-based IoT security using RF-DNA fingerprints and SVM. *IEEE Internet of Things Journal* 8, 10 (2020), 8356–8371.

[39] Ulrich Rührmair, Xiaolin Xu, Jan Sölter, Ahmed Mahmoud, Mehrdad Majzoobi, Farinaz Koushanfar, and Wayne Burleson. 2014. Efficient power and timing side channels for physical unclonable functions. In *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 476–492.

[40] Said Jawad Saidi, Anna Maria Mandalari, Roman Kolcun, Hamed Haddadi, Daniel J Dubois, David Choffnes, Georgios Smaragdakis, and Anja Feldmann. 2020. A haystack full of needles: Scalable detection of iot devices in the wild. In *Proceedings of the ACM Internet Measurement Conference*. 87–100.

[41] Seun Sangodoyin, Frank T Werner, Baki B Yilmaz, Chia-Lin Cheng, Elvan M Ugurlu, Nader Sehatbakhsh, Milos Prvulović, and Alenka Zajic. 2020. Side-channel propagation measurements and modeling for hardware security in iot devices. *IEEE Transactions on Antennas and Propagation* 69, 6 (2020), 3470–3484.

[42] Nader Sehatbakhsh, Monjur Alam, Alireza Nazari, Alenka Zajic, and Milos Prvulovic. 2018. Syndrome: Spectral analysis for anomaly detection on medical iot and embedded devices. In *2018 IEEE international symposium on hardware oriented security and trust (HOST)*. IEEE, 1–8.

[43] Nader Sehatbakhsh, Alireza Nazari, Monjur Alam, Frank Werner, Yuanda Zhu, Alenka Zajic, and Milos Prvulovic. 2019. REMOTE: Robust external malware detection framework by using electromagnetic signals. *IEEE Trans. Comput.* 69, 3 (2019), 312–326.

[44] Nader Sehatbakhsh, Baki Berkay Yilmaz, Alenka Zajic, and Milos Prvulovic. 2020. A new side-channel vulnerability on modern computers by exploiting electromagnetic emanations from the power management unit. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*.

[45] Rahul Anand Sharma, Elahe Soltanaghaei, Anthony Rowe, and Vyas Sekar. 2022. Lumos: Identifying and Localizing Diverse Hidden {IoT} Devices in an Unfamiliar Environment. In *31st USENIX Security Symposium (USENIX Security 22)*. 1095–1112.

[46] Cheng Shen and Jun Huang. 2021. {EarFisher}: Detecting Wireless Eavesdroppers by Stimulating and Sensing Memory {EMR}. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*. 873–886.

[47] Cheng Shen, Jun Huang, Guangyu Sun, and Jingshu Chen. 2022. Electromagnetic Fingerprinting of Memory Heartbeats: System and Applications. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 3 (2022), 1–23.

[48] Akash Deep Singh, Luis Garcia, Joseph Noor, and Mani Srivastava. 2021. I Always Feel Like Somebody's Sensing Me! A Framework to Detect, Identify, and Localize Clandestine Wireless Sensors. In *30th USENIX Security Symposium (USENIX Security 21)*. 1829–1846.

[49] Arunan Sivanathan, Hassan Habibi Gharakheili, Franco Loi, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. 2018. Classifying IoT devices in smart environments using network traffic characteristics. *IEEE Transactions on Mobile Computing* 18, 8 (2018), 1745–1759.

[50] Semtech SX1262. [n. d.]. https://www.semtech.com/products/wireless-rf/lora-core/sx1262mb2cas/. Accessed: 2022-07.

[51] Arduino Uno. [n. d.]. https://docs.arduino.cc/hardware/uno-rev3/. Accessed: 2022-11.

[52] Qiang Xu, Rong Zheng, Walid Saad, and Zhu Han. 2015. Device fingerprinting in wireless networks: Challenges and opportunities. *IEEE Communications Surveys & Tutorials* 18, 1 (2015), 94–104.

[53] Chouchang Yang and Alanson P Sample. 2016. EM-ID: Tag-less identification of electrical devices via electromagnetic emissions. In *2016 IEEE International Conference on RFID (RFID)*. IEEE, 1–8.

[54] Jiabao Yu, Aiqun Hu, Guyue Li, and Linning Peng. 2019. A robust RF fingerprinting approach using multisampling convolutional neural network. *IEEE Internet of Things Journal* 6, 4 (2019), 6786–6799.

[55] Zihao Zhan, Zhenkai Zhang, and Xenofon Koutsoukos. 2020. Bitjabber: The world's fastest electromagnetic covert channel. In *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 35–45.

[56] Mark Zhao and G Edward Suh. 2018. FPGA-based remote power side-channel attacks. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 229–244.

APPENDICES

A CODE

The link below contains the feature extraction code and the Random Forest classifier used in Digitus.
https://github.com/ssysarch/digitus