DeMiCPU: Device Fingerprinting with Magnetic Signals Radiated by CPU

Yushi Cheng
Zhejiang University
yushicheng@zju.edu.cn

Xiaoyu Ji*
Zhejiang University
xji@zju.edu.cn

Juchuan Zhang
Zhejiang University
juchuanzhang@zju.edu.cn

Wenyuan Xu
Zhejiang University
wyxu@zju.edu.cn

Yi-Chao Chen
University of Texas at Austin
yichao@utexas.edu

ABSTRACT

With the widespread use of smart devices, device authentication has received much attention. One popular method for device authentication is to utilize internally-measured device fingerprints, such as device ID, software or hardware-based characteristics. In this paper, we propose DeMiCPU, a stimulation-response-based device fingerprinting technique that relies on externally-measured information, i.e., magnetic induction (MI) signals emitted from the CPU module that consists of the CPU chip and its affiliated power supply circuits. The key insight of DeMiCPU is that hardware discrepancies essentially exist among CPU modules and thus the corresponding MI signals make promising device fingerprints, which are difficult to be modified or mimicked. We design a stimulation and a discrepancy extraction scheme and evaluate them with 90 mobile devices, including 70 laptops (among which 30 are of totally identical CPU and operating system) and 20 smartphones. The results show that DeMiCPU can achieve 99.1% precision and recall on average, and 98.6% precision and recall for the 30 identical devices, with a fingerprinting time of 0.6 *s*. In addition, the performance can be further improved to 99.9% with multi-round fingerprinting.

CCS CONCEPTS

• **Security and privacy → Security services**.

KEYWORDS

Device Fingerprinting; Electromagnetic Radiation; CPU; Smart Devices.

ACM Reference Format:
Yushi Cheng, Xiaoyu Ji, Juchuan Zhang, Wenyuan Xu, and Yi-Chao Chen. 2019. DeMiCPU: Device Fingerprinting with Magnetic Signals Radiated by CPU. In *2019 ACM SIGSAC Conference on Computer and Communications Security (CCS '19), November 11–15, 2019, London, United Kingdom.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3319535.3339810

*Corresponding faculty author.

CCS '19, November 11–15, 2019, London, United Kingdom

ACM ISBN 978-1-4503-6747-9/19/11…$15.00
https://doi.org/10.1145/3319535.3339810

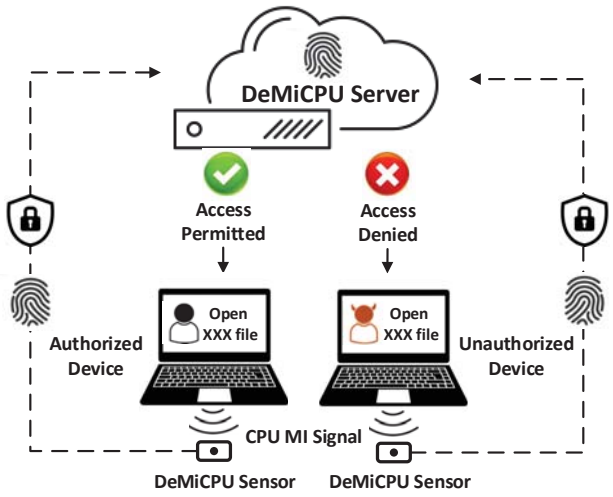


Figure 1: Based on CPU fingerprints, DeMiCPU provides the ability to fingerprint devices for software and applications.

1 INTRODUCTION

Mobile devices have emerged as the most popular platforms to assist daily activities and exchange information over the Internet. According to Gartner [16], there are more than 11 billion phones, tablets and laptops by the end of 2018. Along with the rapid growth is the rising demand of *device authentication*: it is useful for applications to recognize whether they are executing on the same device as the previously registered one, e.g., during payments, to ensure the safety of personal privacy or cyber assets.

One of the strategies for device authentication is *device fingerprinting*. Existing device fingerprinting solutions are mainly based on *internal* device information (e.g., IMEI (device ID), serial numbers of laptops), or built out of software or hardware characteristics. Software-based fingerprints utilize wireless traffic patterns [33], browser properties [46], and etc., while hardware-based fingerprints utilize hardware characteristics such as clock skews [26, 34], accelerometers [13], gyroscopes [2], microphones [11], cameras [14, 29], and Bluetooth implementation [1].

In this paper, we propose to fingerprint devices exploiting the featured electromagnetic interference (EMI) signals radiated by CPU modules on devices, which we call *CPU fingerprints*. The advantage of such a CPU fingerprint is that it can be measured *externally*

rather than *internally* by the operating system (OS), which could be a useful feature for applications on external devices to authenticate the devices. In addition, a CPU module is indispensable for almost all mobile or smart devices, and thus the CPU fingerprint is likely to be more universal compared with aforementioned built-in sensor based approaches.

Based on it, we design DeMiCPU, a device fingerprinting scheme consisting of a trusted DeMiCPU server, a stimulation program on the target device, and a trusted stand-alone DeMiCPU capturing module with a built-in magnetic sensor (in short DeMiCPU sensor), as shown in Fig. 1, and it works as follows. Once an application requests for device fingerprinting, DeMiCPU starts the stimulation program, and the DeMiCPU sensor measures and packages the measurements with protection and uploads the packaged measurements to the DeMiCPU server for fingerprint matching. An attacker may try to impersonate a target device by emulating the EMI radiated by its CPU module, but it is almost impossible to produce an EMI pattern close enough to that of the target device, as analyzed in Sec. 7.

DeMiCPU is promising yet challenging. First, EMI spans a wide spectrum, including high frequency that may produce data at the rate of *Gbps*. Such computation and communication costs are unacceptable, especially for the DeMiCPU sensor. Second, all electronic components inside a device emit EMI and their operation status affects the level of EMI. It is difficult, if ever possible, to control the status of each component across various attempts of measurement. Besides, it is unclear whether the EMI radiated from the same device at various time instants or locations is consistent and the ones from different devices are distinct. Last but not least, the EMI radiation may contain a large amount of noise and how to extract fingerprints efficiently out of the noisy EMI radiation is nontrivial. This paper addresses aforementioned challenges and validates the feasibility of CPU fingerprint.

Which frequency to measure and how to measure? After careful analysis and experimental validation, we choose low-frequency magnetic induction (MI) signals (< 100 kHz). EMI generated by electronic components includes both electromagnetic radiation (EMR) in the far field (> two wavelengths) and magnetic induction (MI) in the near field (< a wavelength). Since EMR is the main cause that affects interoperability of devices, it is suppressed for electromagnetic compatibility [18]. Yet MI signals dominate the near field and do not propagate as far as EMR. Being less a concern of interference, MI signals are not intentionally suppressed and serve as an excellent candidate for extracting hardware fingerprints.

How to induce consistent MI? It is almost impossible to control the status of each component, and thus we focus on controlling the one that emits the majority of MI signals, i.e., the CPU module that consists of the CPU chip and its affiliated power supply circuits. In this way, MI signals contributed by other components on the motherboard can be neglected. CPU fingerprints are made possible because even for devices of the same model, CPU modules are discrepant due to hardware diversities introduced during the manufacturing process. However, various applications may lead to various MI signals of the CPU module (as our experiments confirmed). To ensure that the CPU load and operation status are similar across measurements, we analyze the cause and influencing factors of the emitted MI signals and design a set of instructions

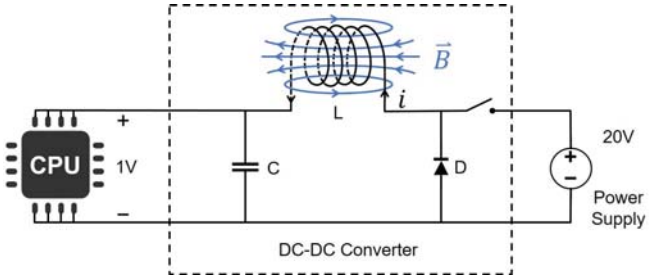


Figure 2: An illustration of a simplified CPU module. A DC/DC converter is connected to the CPU chip for voltage conversion. The inductor in the DC/DC converter can produce strong MI signals when large currents flow through it.

to generate an identical 100% utilization stimulation to the CPU module.

How to extract fingerprints despite of noise? To distinguish the subtle discrepancies of CPU modules when the measurement of MI signals could be noisy, we remove the effects of the geomagnetic field and environmental noise in the pre-processing phase before extracting a set of 15 carefully-selected features, which serves as the fingerprint of the device. To further ensure high accuracy, reliability and usability in DeMiCPU, we compare 10 common classifiers to elect the appropriate classification algorithm. In summary, our contribution includes the following:

- We propose to fingerprint mobile devices by monitoring the MI signals emitted from the CPU module. To the best of our knowledge, this is the first work to attempt device fingerprinting based on the fingerprints of CPU modules.
- We design an efficient MI-based fingerprinting scheme consisting of identical stimulation generation, effective feature extraction and valid fingerprint matching, which can identify devices reliably and accurately.
- We validate DeMiCPU on 90 mobile devices, including 70 laptops and 20 smartphones. The results show that DeMiCPU can achieve 99.1% precision and recall on average, and 98.6% precision and recall for 30 identical devices, with a fingerprinting time of 0.6 s. Both precision and recall can be further improved to 99.9% with multi-round fingerprinting.

2 BACKGROUND

2.1 Magnetic Induction of Electronic Devices

All electronic components emit electromagnetic interference (EMI) when currents flow. EMI emitted from electronic components (e.g., CPUs, fans, GPUs) includes two types: high-frequency electromagnetic radiation (EMR) signals and low-frequency magnetic induction (MI) signals. EMR refers to electromagnetic waves that are synchronized oscillations of electric and magnetic fields and propagate at the speed of light. High-frequency EMR waves are mainly at an order of MHz or above, and are always effectively reduced or shielded [18] to eliminate interference with other electronic components or devices. By contrast, MI signals are non-radiative

waves generated by currents and are typically not intentionally suppressed. In addition, MI signals have a relatively larger strength and a lower frequency than EMR, and thus can be measured by low-frequency magnetic sensors. Therefore, MI signals are good representatives of EMI emitted from a device.

2.2 The CPU Module

The CPU module of a device refers to the CPU chip and its affiliated DC/DC converter. The computation-intensive nature of the CPU chip draws heavy currents from the DC/DC converter, which generate strong MI signals.

CPU. A CPU chip consists of hundreds of millions of CMOS (complementary metal oxide semiconductor) transistors arranged in a lattice form, which performs basic arithmetic, logical, control and input/output (I/O) operations. The CPU current depends on the power consumption of the CMOS circuits, which has three components: static power dissipation, short-circuit power dissipation, and dynamic power dissipation, mathematically denoted as follows [38]:

$$P_{cmos} = P_{static} + P_{short-circuit} + P_{dynamic} \quad (1)$$

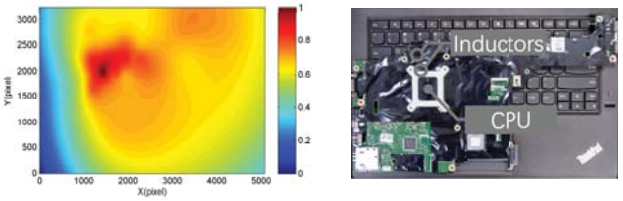
P_{static}, a.k.a., leakage power dissipation, is a steady and constant energy cost caused by the leakage currents of transistors. $P_{short-circuit}$ arises when two transistors in a CMOS gate are on at the same time, which creates a short circuit from the voltage supply to the ground and thus consumes energy. $P_{dynamic}$ is caused by the switching of CMOS gates. Energy consumption of a CPU mainly depends on the dynamic power dissipation of the CMOS lattice, which is roughly equal to the energy change in the output capacitance of CMOS transistors. Average power consumption of a multi-core CPU can be modeled as follows [39]:

$$P_{avg} = \sum_{i=1}^{N} \frac{C_i V(\alpha)^2 A F(\alpha)}{2} \quad (2)$$

where N is the number of CPU cores. C_i, A, V and F are influencing factors, with their meanings summarized in Tab. 1. V and F are further related to the CPU load α due to the power-management technique DVFS (dynamic voltage and frequency scaling) [28] applied by modern devices. DVFS decreases the clock frequency and allows a corresponding reduction in the supply voltage for energy saving. For example, for a ThinkPad T440p laptop, V and F are 0.899 V and 3095.95 MHz when the CPU load is 100%, and they drop to 0.668 V and 798.95 MHz when the CPU becomes idle (2−3% load on average). As all the four factors are hardware related and CMOS circuits are various across CPUs, those factors are distinct from device to device (detailed in Sec. 2.3).

In this section, we begin with the principle of magnetic signals, then elaborate how CPU modules can produce magnetic signals, and finally explain why magnetic signals from CPU modules are differentiated in nature.

DC/DC converter. Due to the difference of voltage levels between the CPU and the power supply system (either a battery or an external power source), a DC/DC converter is placed close to the CPU chip to convert a high voltage to a low one [10]. In Fig. 2, we show the key components of a DC/DC converter and its relationship with the CPU chip. In principle, the high-frequency switch



(a) Heatmap of MI signals from T440p. (b) Physical structure of T440p.

Figure 3: Investigation of MI signals emitted from the T440p laptop. (a) The heatmap of measured MI signal strength. (b) Physical structure of the laptop.

in the DC/DC converter works in a duty-cycle mode to generate a lower voltage. Electronic components including the capacitors, inductors, and diodes are utilized to make the output voltage smooth and continuous. The regulated voltage and currents are then fed into the CPU chip to satisfy its computation requirements.

In short, CPU chips nowadays exploit a reduced voltage for energy efficiency, but incur heavy currents when performing computation-intensive tasks. The heavy currents flowing through the CPU module generate strong MI signals, which are further amplified by the inductor inside the DC/DC converter, due to the effect of coils.

2.3 CPU Module Discrepancy

Hardware discrepancies exist among devices, or more precisely, their CPU modules. For CPUs of various models, all the four factors C_i, V, A and F that affect the CPU power consumption, can be different due to the discrepancies in hardware structure and specification. Even for CPUs of the same model, e.g., Intel Core i5-3210M for ThinkPad T440p laptops, discrepancies exist due to the imperfections introduced during the manufacturing process. As shown in Tab. 1, manufacture techniques have influence upon three factors C_i, V, and F, i.e., the transistor sizes, working voltages, and working frequencies of CPU chips can be distinct. Besides, the DC/DC converter of the CPU module further enlarges the differences. Therefore, MI signals from CPU modules of the same or various models are distinct due to the hardware discrepancies across devices.

In summary, MI signals from CPU modules are different in nature and can serve as a candidate of device fingerprints. In addition, CPU load α affects MI signals since it influences V and F. As a result, MI signals can be strengthened by increasing the CPU load. Thus, to maintain a stable observation of MI signals, the CPU load shall be accurately controlled.

Table 1: Impact factors of CPU power consumption.

P_{avg}	Factors		Meaning
	H	α	
C_i	✓		CMOS capacitance, related to the transistor size and the wire length
V	✓	✓	Supply voltage to CPU
A	✓		Average switching frequency of transistors
F	✓	✓	Clock frequency

H: Hardware related. α: CPU load.

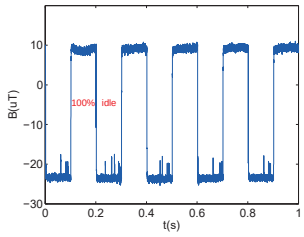


Figure 4: MI signal is highly related to the CPU working period.

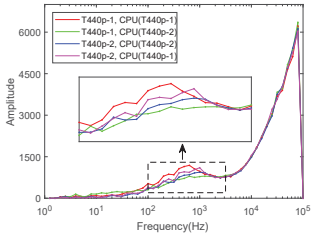


Figure 5: Histograms of MI signals before and after exchanging CPUs.

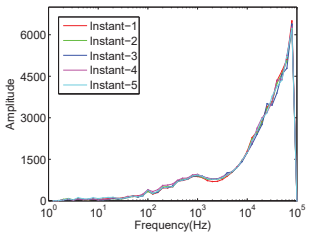


Figure 6: Histograms of MI signals at five instants.

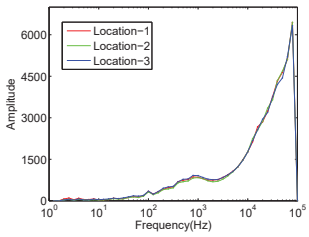


Figure 7: Histograms of MI signals at three locations.

3 PRELIMINARY ANALYSIS

In this section, we verify the feasibility of CPU fingerprints empirically. As shown in Fig. 10, we collect MI signals emitted from the CPU models with a magnetic-field sensor DRV425 [22] from Texas Instruments (TI), and conduct AD conversion with a data acquisition (DAQ) card U2541A [25] from Keysight at a sampling rate of 200 kHz. Each collection lasts for 1 s (0.5 s is shown to be sufficient to fingerprint a device in Sec. 6).

3.1 MI Signals from CPU Module

Does the CPU module produce the strongest MI? To verify whether the CPU module emits the strongest MI signals among all components, we execute a `while(1)` loop (in C++) to generate a CPU utilization of 100%, and measure the MI signal strength by placing the sensor on various spots (33 spots in total) of a Lenovo ThinkPad T440p laptop's surface (device No. 31 in Tab. 3). We plot the heatmap of the MI signals measured across the laptop's surface in Fig. 3(a), from which we can find that the strongest MI signals are observed at "S" and "D" keys. Dismantling the laptop reveals that two inductors of the DC/DC converter that powers the CPU chip are located right below these two keys, as shown in Fig. 3(b). This indicates that the CPU module, specifically the DC/DC converter, produces the strongest MI signals when the CPU is under a high load.

Does the CPU load affect the MI Signals? To understand whether the variation of the CPU load affects MI signals emitted from the CPU module, we force the CPU to work in a duty-cycle mode at a frequency of 5 Hz, i.e., alternating between a 100% utilization and an idle mode at an interval of 100 ms. Throughout the experiments, the sensor was placed above the CPU module, i.e., on S and D keys, to measure the emitted MI signals. The results shown in Fig. 4 confirm that the CPU load does affect the MI signals. Thus, it is important to create a consistent software stimulation to ensure the same CPU load such that the fingerprints generated from the CPU module are consistent for the same device.

Do other components affect the MI Signals? Modifying the status of other computer components may lead to variation of the MI signals. However, MI signals generated by others attenuate rapidly with distance due to the near field effect. We observe no noticeable difference between the MI signals collected right above the CPU module when the fan was turned on and off. As a result, `DeMiCPU` does not control other components during device fingerprinting.

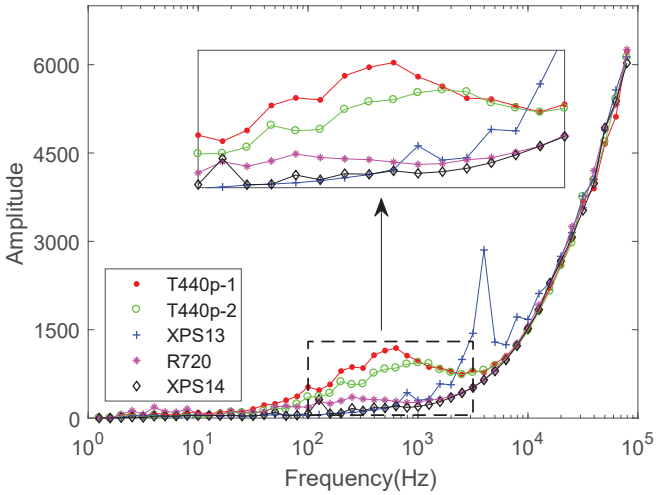


Figure 8: Histograms of MI signals from 5 laptops. Even for the two laptops of the same model, i.e., T440p-1 and T440p-2, the MI signals show discrepancies.

3.2 Evidence of CPU Fingerprint

To explore the existence of CPU fingerprint, we conduct an experiment with 5 laptops, which are two Lenovo ThinkPad T440p (T440p-1 and T440p-2, for short), Dell XPS 13, Lenovo R720, and Dell XPS 14. Detailed specifications of these laptops (Device No. 31, No. 32, No. 61, No. 49 and No. 62) are summarized in Tab. 3, among which two laptops (T440p-1 and T440p-2) are from the same model and installed with the same operating system and the rest are of different models.

We execute the `while(1)` program to keep the CPU at a 100% utilization and measure MI signals above the CPU module of each laptop. We then perform Fast Fourier Transform (FFT) on the collected MI signals and plot their one-dimensional histograms in Fig. 8, with a logarithmic bin size of $10^{0.1}$. The histogram represents the frequency distribution of the MI signals, from which we can observe distinct "patterns" for the 5 laptops in the frequency range from 20 Hz to 10 kHz. Especially, laptops of different models show more discrepancies compared with those of the same model. Nevertheless, the two T440p laptops remain distinguishable even only with one histogram feature.

The above findings shed light on the existence of CPU fingerprints. However, to make the fingerprint robust and accurate, especially for devices from the same model, more features in both time and frequency domains should be investigated to enhance the fingerprint.

3.3 What Contributes to CPU Fingerprint?

To understand whether the fingerprint is created by the CPU chip, the DC/DC converter, or the combination of both, we exchange the CPUs of the two T440p laptops and obtain two "new" laptops (T440p-1 with CPU from T440p-2 and T440p-2 with CPU from T440p-1). Similar to previous experiments, the CPU utilization is set to 100% during collection and MI signals are measured above the CPU module before and after swapping the CPUs. The results in Fig. 5 show that MI signals for four configurations are all different, which indicates that the fingerprint originates from the combination of the CPU chip and its affiliated DC/DC converter, i.e., the CPU module.

3.4 Temporal and Spatial Consistency

The MI signal from a device should be consistent across time and space to serve as a robust fingerprint. To investigate the temporal consistency, we collect 30 MI signals from the T440p-1 laptop at 5 time instants across two days, i.e., the first three instants are within one day (morning, afternoon and evening) and the other two are in the next day (morning and evening). The T440p-1 laptop is set to 100% utilization and one-second MI signals are collected each time. The results depicted in Fig. 6 indicate that MI signals remain consistent regardless of time.

To investigate the spatial consistency, we collect 30 MI signals from the T440p-1 laptop at 3 locations (one in a lab, two at home; and the two places are about 3 kilometers apart). Note that we do not intentionally avoid or remove metal and magnetic materials around the collecting device during experiments. As a result, due to the impact of the earth's magnetic field and ambient noise (especially in the lab, with numerous electronic devices surrounding), the initial magnetic magnitude of the sensor is geo-spatial dependent. However, the strength of the earth's magnetic field and ambient noise is relatively static at a specific spot and thus mainly contributes to the constant part of the collected MI signals. As a result, the FFT operation shall have eliminated the impact of the earth's magnetic field as well as the ambient noise. The results in Fig. 7 also validate that the frequency-domain MI signals remain consistent regardless of locations.

All these experiments provide strong evidence that CPU modules can produce strong MI signals that maintain good distinguishability and consistency, and the MI signals from CPU modules serve as promising device fingerprints.

4 THREAT MODEL

In this paper, we have the following assumptions.

Impersonation. Although it is feasible for attackers to launch a Denial-of-Service (DoS) attack by emitting EMI or even placing a strong magnet close to the DeMiCPU sensor, the goal of the attackers is to impersonate a legitimate device. Thus, we focus on replay or mimic attacks.

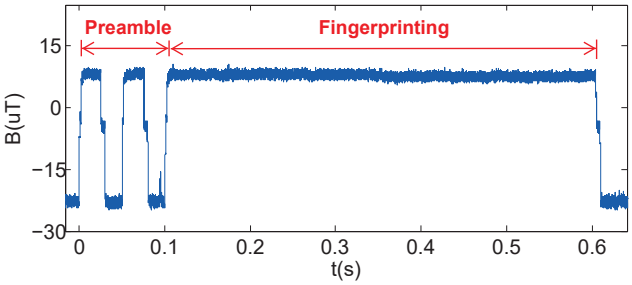


Figure 9: Structure of the MI signal, including a 0.1 *s* preamble and a 0.5 *s* fingerprinting sample.

Acquisition of Similar Device. We assume the adversary can obtain similar devices as the target one, e.g., a device of the same model, to imitate the target device and have full control of them.

Secure Communication. We assume that the communication between the DeMiCPU sensor and the DeMiCPU server and between the server and the software (application) is secure. For instance, DeMiCPU can package the MI measurements or matching results with encryption, by the well-known secure communication protocols [3, 31, 32]. As a result, the attacker cannot create forged measurements or modify the measurements/matching results.

5 DESIGN

In this section, we describe the 3 sub-modules of the overall DeMiCPU system: (1) Fingerprint generation; (2) Fingerprint extraction; (3) Fingerprint matching.

5.1 DeMiCPU Fingerprint Generation

To obtain MI measurements that produce consistent fingerprints, it is important to solve the following two challenges.

- How to stimulate the CPU such that it generates the MI signal that can produce a consistent device fingerprint?
- How to collect and identify the MI signal segment that maps to the one under stimulation even if an attacker may disturb the communication between the stimulation program and the trusted capturing sensor?

To address these two challenges, we design the stimulation program such that it produces the MI signal trace in Fig. 9, which is composed of a preamble and a fingerprinting signal that are both generated by controlling the CPU load in a proactive way. As thus, DeMiCPU only needs to transmit a signal as short as 0.6 *s* for fingerprinting.

5.1.1 Preamble. To identify the MI signal segment that is under stimulation, a preamble is used for the trusted capturing sensor to detect the start of the fingerprinting signal. DeMiCPU stimulates the device such that a unique MI pattern is generated as a preamble, thereby allowing the sensor to identify it with cross-correlation. We realize the preamble by manipulating the CPU load and generate a sequence of [1,0,1,0] ("1" for full-utilization mode and "0" for idle mode) as shown in Fig. 9, which lasts for 100 *ms* in total.

5.1.2 Stimulating CPU. The strength of the MI signals emitted from the CPU module depends on the current, which is related to

the CPU load. In order to obtain stable MI signals to produce CPU fingerprints, we stimulate the CPU by controlling its utilization ratio. Without loss of generality, the total CPU utilization is the sum of CPU utilization from all running processes, including both system and user processes, which can be modeled as follows:

$$CPU\_util = Sys\_processes + User\_processes \quad (3)$$

Utilization Ratio. One intuitive question is what utilization ratio to use, 100%, 50%, or other values? In fact, it is difficult to precisely control the utilization since 1) it is hard to accurately restrict system and user processes to a certain level, and 2) the CPU scheduling policy further worsens the problem. For instance, 50% CPU utilization means that the CPU works in 5 clock cycles and is idle in the remaining 5. Without inspecting and modifying the scheduling algorithm, it is almost impossible to ensure that the CPU behaves the same in all clock cycles.

To address it, we choose to keep the CPU running in the full-utilization (100%) mode to obtain an identical output. Another benefit of such an implementation is that higher CPU utilization generates stronger MI signals, which helps to lighten the impact of ambient noise. We achieve the full-utilization mode by invoking CPU-consuming instructions, such as `while(1)` in our implementation. As thus, system processes, DeMiCPU stimulation process, and other user processes together compose the 100% utilization.

DeMiCPU Priority. During fingerprinting, however, other user processes, i.e., background applications, are not likely to be the same, which may render the stimulation nonidentical. To eliminate the influence of other user processes, we assign a superior priority to the DeMiCPU stimulation program, which is higher than the base one of other user processes yet lower than that of the system processes since they only account for 1-2% CPU utilization on average.

Mainstream operating systems such as Windows, Linux, and Mac OS X, are all able to support such an implementation. For instance, Windows implements a priority-driven, preemptive scheduling system, where the highest priority runnable threads are executed first. Each thread, which is the smallest unit of program execution flow, has a base priority as a function of its process priority class and relative thread priority. Normally, user applications and services start with a base priority level 8, i.e., both process and thread priorities are `normal` [37]. Thus, we shall at least assign the DeMiCPU stimulation program with a priority level higher than that.

In particular, we examine the highest priority of the user threads, which is usually a priority level 8 as mentioned before. Then, we assign a higher priority to the DeMiCPU thread, e.g., a `normal` process priority but an `above normal` thread priority, i.e., a priority level 9, to eliminate the impact of other user processes. In addition, since modern CPU chips support multi-core and multi-thread, we bind a stimulation thread to each available logical processor core, including the virtual ones created by Hyper-Threading [30]. As thus, the CPU utilization under stimulation is as follows:

$$CPU\_util\_stimu = Sys\_processes + \texttt{DeMiCPU} = 100\% \quad (4)$$

Feedback. In general, such a design is able to generate an identical stimulation. However, in a rare case, a thread with a higher priority may be launched during fingerprinting, making the stimulation different than planned. To further guarantee the validity of the DeMiCPU stimulation, we introduce a feedback mechanism, i.e., examining system logs after stimulation to confirm that DeMiCPU exclusively uses the CPU during fingerprinting. If not, DeMiCPU abandons the current measurements and triggers a second collection. Moreover, the CPU frequency may drop due to a high CPU temperature or low battery. Thus, the feedback mechanism examines the CPU working frequency before and during stimulation. If a previous or midway frequency drop is detected, DeMiCPU abandons the current measurements and defers its collection till the CPU recovers from the low frequency mode, as revealed in Algorithm 1.

Algorithm 1: DeMiCPU Stimulation

```
1  CPU_Frequency ← Get_Current_CPU_Frequency()
2  if CPU_Frequency > threshold_1 then
3      C_priority ← Get_Current_Highest_Priority()
4      //get the highest priority level of running user threads
5      DeMiCPU_priority ← Gen_Priority(C_priority)
6      cpunum ← Get_CPU_Core_Num()
7      // get the number of CPU logical processors
8      for i ∈ range(1, cpunum) do
9          hThread(i) ← CreateThread()
10         // create the i^th DeMiCPU stimulating thread
11         SetThreadPriority(hThread(i), DeMiCPU_priority)
             // set the i^th DeMiCPU stimulating thread with the
             generated DeMiCPU priority level
12         C_Thread ← GetCurrentThread ()
13         C_Mask = 0x0001 * 2^(i−1)
14         SetThreadAffinityMask (C_Thread, C_Mask)
15         // bind the i^th DeMiCPU stimulating thread to the i^th
             CPU logical processor
16         preamble_gen()
17         fingerprinting_signal_gen()
18     Stim_Util ← Get_Util_Feedback()
19     Stim_Freq ← Get_Freq_Feedback()
20     if Stim_Util < threshold_2 then
21         DeMiCPU Stimulation
22     if Stim_Freq < threshold_1 then
23         sleep(5)
24         DeMiCPU Stimulation
25 else
26     sleep(5)
27     DeMiCPU Stimulation
```

In this way, we minimize the influence of software environment and output stable fingerprinting signals as shown in Fig. 9.

5.2 DeMiCPU Fingerprint Extraction

5.2.1 Pre-processing. Preliminary analysis confirms the temporal and spatial consistency of the MI signals in the frequency domain. However, the time-domain MI signal is geo-spatial dependent due to the impact of the earth's magnetic field and ambient noise. As the strength of the earth's magnetic field and ambient noise is relatively static at a specific spot, we assume it mainly contributes

to the constant part of the collected MI signals. To eliminate its impact, we normalize the raw MI signal, i.e., the measured signal in Fig. 9, before extracting features.

Denote the measured signal as B, we normalize B to obtain the pre-processed MI signal M for feature extraction as follows:

$$M = \frac{B - min(B)}{max(B) - min(B)} \tag{5}$$

Note that although the above solution is designed for scenarios where ambient MI signals are relatively static, we argue it also works with time-varying magnetic signals such as power frequency interference from nearby electrical equipment, because that the time-varying MI signals from other devices quickly attenuate and thus have little influence.

5.2.2 Feature Selection. For each pre-processed signal M, we extract 30 scalar features from both time and frequency domains. We exploit LibXtract [7], a lightweight feature extraction library for time series, which can output a number of statistical feature candidates. Besides the features offered by LibXtract, we investigate the physical meaning of the MI signal, and manually select features, e.g., Spectrum Kurtosis and Spectrum Smoothness, on which remarkable distinctions can be observed in Fig. 8.

To further determine critical features, we rank them with the help of FEAST toolbox [6], which is a commonly used feature ranking tool in machine learning. From the results, we obtain the top 15 features in time and frequency domains and construct a feature set as: $\mathbb{F}$ ={*Spectrum Roll Off, Spectrum Kurtosis, Average Deviation, Spectrum Spread, Spectrum Smoothness, RMS Amplitude, Spectrum Standard Deviation, Spectrum Irregularity-K, Spectrum Skewness, Spectrum Flatness, Standard Deviation, Spectrum Irregularity-J, Mean, Skewness, Spectrum Mean*}. The orders in the set indicate their ranking orders with *Spectrum Roll Off* giving the largest information gain.

For a fingerprinting signal i from a device, hereafter we define the feature set $\mathbb{F}_i$ as the *fingerprint* of the device.

5.3 DeMiCPU Fingerprint Matching

The DeMiCPU cloud server utilizes supervised learning to classify each trace with the extracted feature set $\mathbb{F}$. To select the appropriate classification algorithm, we compare 10 commonly-used classifiers and the detailed results can be found in Fig. 11(a). For the sake of high classification accuracy and robustness over a single classification algorithm, we employ an ensemble classification approach ExtraTrees [19], which fits a number of randomized decision trees on various sub-samples of the dataset and uses averaging to improve prediction accuracy and avoid over-fitting.

Training. During the training process, for a specific device, k traces from it are utilized as the positive class, and k traces from each of the rest devices serve as the negative class to train a binary classifier. Therefore, for j devices, j binary classifiers are trained in total. In real-world deployment, we may need to extend the classification system when a new device comes and registers. Under that circumstance, the feature sets of the new device are extracted and trained to obtain a new binary classifier without the need of retraining the original j classifiers. The new classifier is finally incorporated with the existing classifiers to constitute a new classification system.

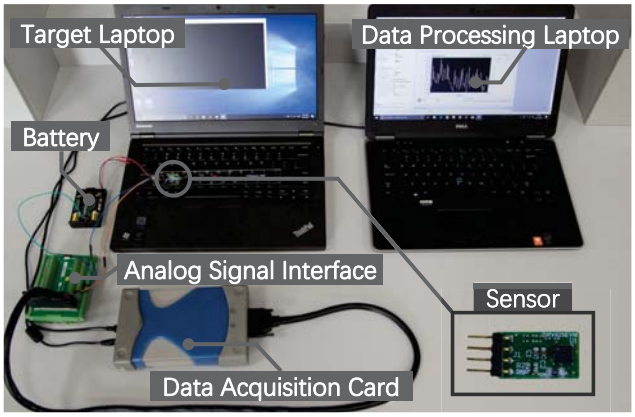


Figure 10: Experimental setup. The magnetic sensor is vertically placed on the surface of the target laptop for MI signal collection.

Matching. When matching, the server analyzes the fingerprint signal from the device to be identified and extracts its feature set $\mathbb{F}$. Then, the server feeds it to the classifier of which class the device claims to be, to verify its identity.

6 EVALUATION

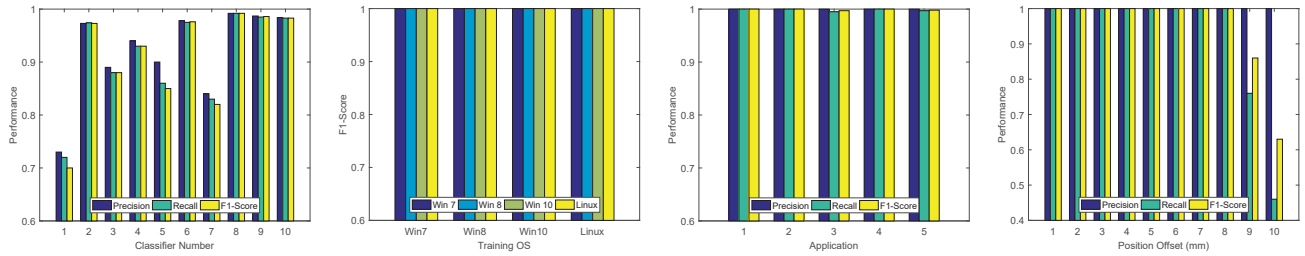
To evaluate the performance of DeMiCPU, we have conducted experiments with 70 laptops and 20 phones across 30 days, among which 30 laptops are of the same model. The detailed information of each device is shown in Tab. 3 (in Appendix A.1). In summary, the performance of DeMiCPU is:

- DeMiCPU achieves 99.1% precision and recall for both laptops and phones, and more than 98.6% precision and recall for 30 identical devices with one-round fingerprinting, and the performance can be further improved to 99.9% with multi-round fingerprinting.
- DeMiCPU can operate with little influence from operating systems, background applications, fan on/off states or CPU temperature.
- DeMiCPU supports low sampling rate which makes it a universal approach running on ubiquitous smart devices.

6.1 Experiment Setup

With the experiment setup described in Tab. 3 and Fig. 10, we collect 100 MI traces for each of the 90 devices and each trace lasts for 0.5 s (excluding the preamble). The settings for the laptops and smartphones are as follows.

Stimulation Program Setup. We implement the stimulation program in Algorithm 1 on five operating systems, i.e., Windows (in C++), Linux (in C++), Mac OS (in Java), Android (in Java), and iOS (in C++), to stimulate the CPU and generate a fingerprinting



(a) Performance of 10 common classifiers. (b) Impact of OSs. (c) Impact of background applications. (d) Impact of test point offset.

Figure 11: Micro-benchmark evaluation results of DeMiCPU on 5 randomly-chosen devices.

signal. The lightweight program is pre-installed on the experimental laptops/ smartphones.

Data Collection Setup. We collect MI signals from the 90 devices using a magnetic-field sensor DRV425 [22] from TI. As shown in Fig. 10, the sensor is vertically placed on the surface of laptops or phones (test points are shown in Tab. 3 in detail) since MI signals emitted from the CPU modules are in the vertical direction. A data acquisition (DAQ) card U2541A [25] from Keysight is utilized for AD conversion with different sampling rates, e.g., 100 Hz, 200 Hz, 1 kHz, and etc. A data processing laptop connects with the DAQ card through a USB, which locally stores and processes the collected data.

6.2 Performance Metrics

Given an MI fingerprint from a device, DeMiCPU verifies whether it belongs to the device (classifier) that it claims to be. For each classifier i, we define TP_i as the true positives for classifier i, i.e., the number of fingerprints that are correctly accepted as i. Similarly, FN_i and FP_i refer to the number of fingerprints that are wrongly rejected, and wrongly accepted as i, respectively. We define the standard classification metrics for each classifier i as:

$$Precision(i) = \frac{TP_i}{(TP_i + FP_i)} \tag{6}$$

$$Recall(i) = \frac{TP_i}{(TP_i + FN_i)} \tag{7}$$

$$F1 - Score(i) = \frac{2 \times Pr_i \times Re_i}{(Pr_i + Re_i)} \tag{8}$$

The final precision, recall and F1-Score for DeMiCPU are the average of the 90 classes.

6.3 Micro-benchmark Evaluation

In this subsection, we evaluate the impact of classifier choices, operating systems, background applications, on/off states of fans, temperatures and displacements of test points. Five devices from Tab. 3 are randomly chosen for the micro-benchmark evaluation.

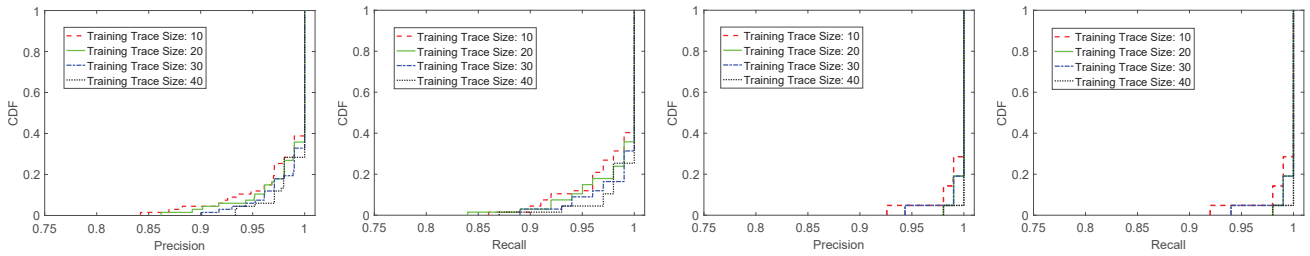
6.3.1 Classifier Choice. To select the appropriate classifier for DeMiCPU, we compare 10 commonly-used supervised learning algorithms. They are 1) Logistic Regression, 2) Gaussian Naive Bayes, 3) K-Nearest Neighbors, 4) Linear Discriminant Analysis, 5) Quadratic Discriminant Analysis, 6) Decision Tree, 7) Support Vector Machine, 8) ExtraTrees, 9) Random Forest, and 10) Gradient Boosting. We employ the 10-fold cross validation to evaluate the classifier performance, which can combine measures of fit and thus derive a more accurate estimation for model prediction performance.

We randomly choose 30 traces from each device, feed them into the classifiers and record the corresponding accuracy. The results in Fig. 11(a) show that 6 out of 10 classifiers show an F1-score above 0.9, with the classifier 8) ExtraTrees, 9) Random Forest, and 10) Gradient Boosting being the best 3 classifiers. Thus, we can assume that the data possesses good property in terms of discrepancies, i.e., CPU fingerprints are able to discriminate devices. In the following experiments, we employ ExtraTrees since 1) it shows the best accuracy, and 2) it's an ensemble classification approach which achieves better robustness over a single classification algorithm.

6.3.2 Operating Systems. A device may install different OSs during its lifetime. To investigate whether OSs affect DeMiCPU, we install 4 OSs which are 1) Window 7 Home Basic 7601, 2) Kali Linux 2.0, 3) Windows 8 Professional 9200, and 4) Windows 10 Enterprise 10240 on the experimental laptops, and conduct experiments under each OS to investigate the impact of OSs. We train the classifier with traces from one OS and test it under all the four OSs. The results in Fig. 11(b) indicate that with the DeMiCPU stimulation program, the same device can be successfully identified across different OSs with precision, recall and F1-Score of 1. It confirms that with elaborately designed stimulation, OS-associated processes only account for a tiny portion of the CPU utilization during fingerprinting, which is within the tolerance of DeMiCPU. Thus, we believe DeMiCPU fingerprint is independent on OSs.

6.3.3 Background Applications. DeMiCPU stimulation is designed to be undisturbed by other user processes. To evaluate its performance against background applications in practice, we conduct experiments on each device with several daily-used applications. They are 1) WeChat, 2) Microsoft Word, 3) Google Chrome, 4) YouTube, and 5) MATLAB, with statistically increasing CPU utilization when normally used. We train the classifier using traces with no background application, and test it using traces with one of the aforementioned background applications, respectively. The results shown in Fig. 11(c) confirm that, background applications barely have impact on the performance of DeMiCPU since it can preempt the CPU even if user applications run.

6.3.4 Displacement of Test Point. Due to that all electronic components inside a device emit MI signals, the measuring sensor may capture MI signals from other components when moved away from



(a) Overall precision for laptops. (b) Overall recall for laptops. (c) Overall precision for smartphones. (d) Overall recall for smartphones.

Figure 12: Overall performance of DeMiCPU with different training data sizes (10, 20, 30 and 40).

the CPU module. To investigate the impact of the test point displacement, we vary the position of the sensor as follows: Starting from the center of the original test point (depicted in terms of key positions in Tab. 3), we gradually move the sensor with a step of 1 *mm* in four directions: upwards, downwards, left and right. The classifier is trained at the original test point, and tested at each changed position. For each displacement, we average the precisions, recalls and F1-scores in four directions, and show the final results in Fig. 11(d). From the results, we can see that within an offset of 8 *mm*, DeMiCPU achieves a high accuracy (> 99%). That is, a user can conduct DeMiCPU fingerprinting with a displacement tolerance of around a key size, which is approximately $10 - 15$ *mm* wide.

6.3.5 Fans. When fingerprinting a laptop, an electric fan aside the CPU module emits MI signals as well. To investigate the impact of fans, we collect 200 traces from each device with fan on and off, i.e., 100 traces each. We train the classifier with the fan-on traces and test it with the fan-off traces. The resulting F1-Score is 1, indicating that MI signals from the fan have little influence. We assume it is because that fans have much lower power (several watts) compared with CPUs (tens of watts), and the large distance (around 10 *cm*) between fan and CPU makes the MI signal from a fan quickly attenuate.

6.3.6 Temperature. CPU temperature changes over time and load, and might be an influence factor for DeMiCPU. To investigate, we test DeMiCPU under different CPU temperatures. Note that in DeMiCPU stimulation, we introduce a CPU frequency check before stimulation since the CPU protection mechanism will decrease the CPU frequency when its temperature becomes too high, e.g., above 90 °C. Thus, DeMiCPU normally works when the CPU temperature is not too high to cause a frequency drop and we first test DeMiCPU under this range. We train the classifier using traces collected when CPU temperature is 65 °C, and test it under the cases of 43 °C, 52 °C, 60 °C, 68 °C, and 78 °C, respectively. The F1-Scores for the five cases are all 1. To further explore the performance of DeMiCPU under a high temperature, we manually turn off the CPU protection mechanism and test the system under 90 °C. The resulting F1-score is also 1, indicating that DeMiCPU works as well. Thus, we believe DeMiCPU is robust to CPU temperature changes.

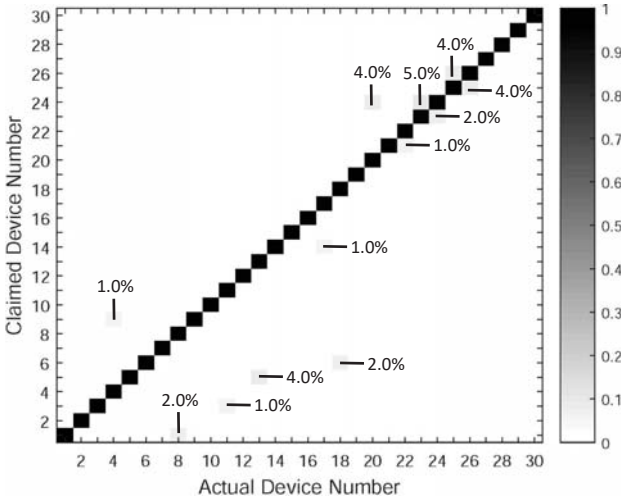


Figure 13: Confusion matrix of 30 identical laptops.

6.4 Overall Performance

In the overall performance evaluation, 100 traces are collected from each device in Tab. 3, and the employed classifier is ExtraTrees with a tree number of 100.

6.4.1 Impact of Training Size. In the first set of experiments, we train the system with x traces and test it with the rest $100 - x$ traces (they are never used for training). x is set to 10, 20, 30, and 40 (correspond to 5, 10 ,15, and 20 seconds) respectively, to evaluate the appropriate size of training data. We calculate the $Precision(i)$ and $Recall(i)$ for each device (class) i, and plot their CDFs in Fig. 12(a) - 12(d) with different training data size x. Even with 10 training traces (correspond to 5 seconds), 90% of the precisions and recalls are above 93.0% for all the laptops and smartphones. The average precision and recall are 98.3% and 98.2% for the 70 laptops, and 99.4% and 99.3% for the 20 smartphones. With the increasing of training data size, both precision and recall are improved. Given the training size 20, DeMiCPU is able to achieve an average precision and recall of 99.0% and 99.0% for the laptops, and 99.8% and 99.8% for the smartphones. Besides, when the training data size further increases, the performance of DeMiCPU approaches 100%. To strike the balance between usability and accuracy, we choose 20 traces for training, which only amount to 10 *s*. Training data size is then set to 20 in the rest of the evaluation.

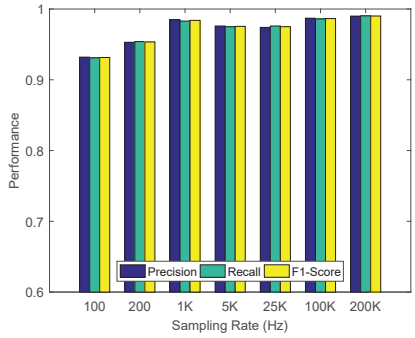


Figure 14: Impact of different sampling rates.

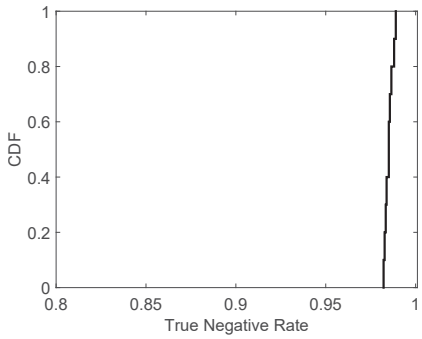


Figure 15: Performance of DeMiCPU against 5 random aliens.

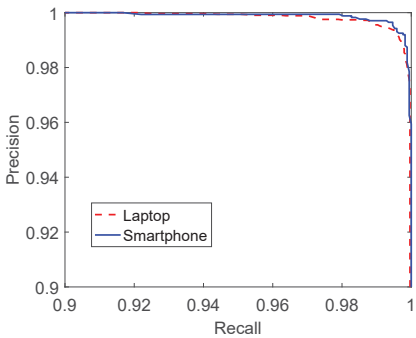


Figure 16: Precision-recall curves for laptops and smartphones.

6.4.2 Performance of Devices of Same Model. The reason why precision and recall behave better on smartphones is that there are more devices of the same model for laptops. As a result, false positives and false negatives may occur. To take a close look, we plot a confusion matrix for devices No. 1-30 (i.e., the 30 ThinkPad T430 laptops) in Fig. 13. These 30 laptops are of the same model and installed with the same operating system, and thus are more likely to be confused with each other. From the confusion matrix, we can observe that device No.23 and No.24, as well as device No.25 and No.26 contribute a relatively lower accuracy, with the worst precision of 91.6%. Nevertheless, DeMiCPU can still achieve an average precision of 98.7%, and an average recall of 98.6% for the 30 identical devices.

6.4.3 Impact of Sampling Rate. To investigate the sampling rate requirement of DeMiCPU, we test the system by setting the sampling rate to 100, 200, 1 k, 5 k, 25 k, 100 k, and 200 kHz respectively. 100 traces from each of the 90 devices are collected at each sampling rate for training and testing. The resulting precisions and recalls are shown in Fig. 14, from which we can observe that precisions and recalls of DeMiCPU do not change significantly with lower sampling rates. Especially, with a 1 kHz sampling rate, DeMiCPU achieves a precision of 98.5% and a recall of 98.3%, which are nearly equivalent to the results under higher sampling rates. Even with a 100 Hz sampling rate, the precision and recall can be as high as 93.2%. This finding is encouraging since it indicates that DeMiCPU can even use ubiquitous smart devices with limited sampling rate capability for fingerprint collection. For instance, most smartphones nowadays are equipped with a built-in magnetometer that supports 100 Hz sampling rate. Low requirement of sampling rate makes DeMiCPU a more universal device fingerprinting mechanism.

6.4.4 Scalability of DeMiCPU. Although it's difficult to evaluate the capability of DeMiCPU with a very large set of devices, we conduct several experiments in which we increase the number of tested devices gradually to get a sense of how DeMiCPU scales. With the same settings in the 90-device experiments, we change the total number of tested devices and repeat the experiments. First, we randomly choose and use 20 devices to obtain the precision and recall of DeMiCPU. Then, we increase the quantity of devices to 30, 50, 70 and 90, and recalculate the precisions and recalls. Tab. 2 shows how accuracy changes with the increasing number of devices,

Table 2: Average precision, recall and F1-Score of DeMiCPU with different numbers of tested devices.

Number of devices	Precision	Recall	F1-Score
20	1.000	1.000	1.000
30	0.997	0.996	0.996
50	0.997	0.997	0.997
70	0.995	0.994	0.994
90	0.991	0.991	0.991

from which we can find that the performance of DeMiCPU does not change significantly as the number of devices increases. It provides encouraging signs that DeMiCPU is likely scalable to a large number of devices.

6.4.5 Impact of Alien Devices. In real-world deployment, it is likely that DeMiCPU needs to identify alien devices, i.e., devices that are not trained beforehand. To understand how DeMiCPU performs with alien devices, we conduct the following experiments. From the 90 devices, we randomly choose 85 devices for training and get the corresponding 85 binary classifiers. The rest 5 devices, which serve as aliens to the trained system (they are never used for training), are utilized to test the performance. The 5 devices take turns to input their traces to each of the 85 classifiers to see if they can be accepted. We repeat the experiment for 10 times to eliminate the random errors and plot the CDF of true negative rates in Fig. 15. The results reveal that DeMiCPU can successfully reject alien devices with a minimum probability of 98.2% and an average probability of 98.7%, which indicates its high reliability.

6.4.6 Multi-round Fingerprinting. In aforementioned evaluation, the threshold for each binary classifier is 0.5 by default. However, in practice, precision is likely to be prior to recall for the sake of high reliability and security, and recall can be further improved through multi-round fingerprinting.

To investigate the appropriate threshold to achieve high precision and the minimum fingerprinting round to achieve high recall, we plot the precision-recall curve by varying the threshold for each classifier. As DeMiCPU is a system consisting of multiple binary classifiers, we employ the same threshold in each classifier and average their precisions and recalls as the final performance. The results shown in Fig. 16 reveal that, for both laptops and smartphones, the

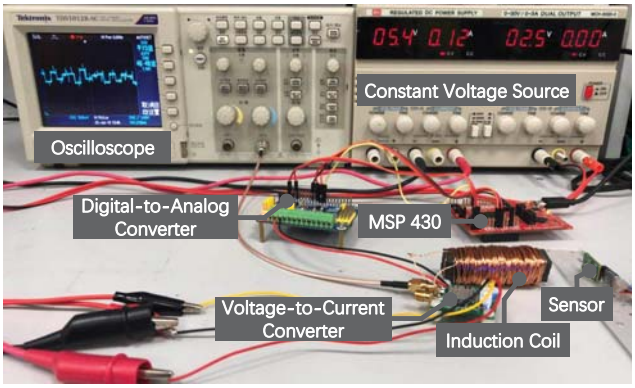


Figure 17: DIY replay attack equipment with a handcrafted induction coil. The recorded MI sample is emitted by the MSP430 in the form of discrete voltages, which are first converted into analog signals by a Digital-to-Analog Converter (DAC) and then converted into corresponding current signals by a Voltage-to-Current Converter (VCC).

precision approaches 100% when the threshold increases. Specifically, for laptops, the recall is 97.0% when the precision is 99.9% with a threshold of 0.54, which can be further improved to 99.9% with two-round fingerprinting and 99.99% with three-round fingerprinting. Similarly, for smartphones, the recall is 98.3% when the precision is 99.9% with a threshold of 0.64, and the recall can approach 99.999% with only three-round fingerprinting. Therefore, with three-round fingerprinting, DeMiCPU can achieve a 99.9% precision and an over 99.99% recall on both laptops and smartphones.

To summarize, our evaluation with 90 laptops and smartphones shows that smart devices can be identified leveraging the fingerprints of their CPU modules. While even a larger study is needed to confirm the scalability of our findings, to the best of our knowledge, this is the first work to attempt device fingerprinting based on fingerprints of CPU modules.

7 DISCUSSION

In this section, we conduct the security analysis and discuss the limitations of DeMiCPU.

7.1 Security Analysis

Since the goal of the attackers is to impersonate a legitimate device, we discuss two attacks: replay attacks and mimicry attacks. To launch a replay attack, an adversary may have a brief physical access to the target device. She may record the MI signal of the target device and replay the recorded sample to fool the DeMiCPU sensor. For mimicry attacks, she may find a similar device to imitate the legitimate one.

7.1.1 Replay Attack. A replay attack consists of two steps: recording and reproducing. We study the feasibility of such attacks based on two sets of equipment: commercial off-the-shelf (COTS) devices and DIY sets with handcrafted coils.

COTS device. The effectiveness of recording and emitting radiation signals is determined by the sensitivity of the sampling devices and the gain of the antennas. Much work has demonstrated the feasibility of replaying radio frequency (RF) signals at reasonable cost, e.g., utilizing a Universal Software Radio Peripheral (USRP) with a matching antenna to replay signals at 2.4 or 5 GHz for Wi-Fi, 900 MHz for GSM (Global System for Mobile Communications), 13.56 MHz for NFC (Near-field Communication), and etc. These RF bands are at least at the order of MHz and a variety of off-the-shelf matching antennas are available. In comparison, the effective frequency range of DeMiCPU is below 10 kHz, whose matching antennas, i.e., VLF (very low frequency) antennas, are usually used for military communication with submarines and few commodity antennas are available. Moreover, VLF antennas are typically large, e.g., a dipole antenna for 10 kHz can be longer than 7.5 km.

Without matching antennas, we may refer to dedicated equipment to record MI samples. For instance, we found a N9038A MXE EMI receiver from Keysight that can analyze signals from 3 Hz to 44 GHz at the cost of \$90, 000 USD. However, we were unable to find equipment that can reproduce the recorded samples with abundant signals ranging from DC to 10 kHz since most RF generators on the market only support frequency higher than 9 kHz.

DIY set with handcrafted coils. Unable to replay MI signals with COTS devices, we design our own replay equipment: We record the MI sample with the DRV425 magnetic sensor and replay the signal with a handcrafted induction coil driven by a MSP430F5529 LaunchPad [23], as shown in Fig. 17. We program the LaunchPad to output the recorded MI sample in a form of discrete voltages, which are then converted into analog signals by a Digital-to-Analog Converter (DAC). The analog voltage signals are further converted into corresponding current signals to drive the induction coil. A ferrite core is inserted into the coil to augment its permeability. A Constant Voltage Source (CVS) is utilized to power the VCC, and an oscilloscope is used to monitor the output voltage of the DAC.

To quantify the MI signals measured by sensors, we refer to the Ampere's circuital law [45], which models the magnetic flux generated by a charged coil as follows:

$$\Phi_B = \mu NIScos\theta \tag{9}$$

where μ is the magnetic permeability of the coil, N is the number of turns, I is the current flowing through the coil, S is the area of the magnetic sensor's sensing surface, and θ is the angle between the magnetic field lines and the normal line (perpendicular) to S. Therefore, although we elaborately reproduce the MI signal, the distance and angle between the coil and sensor affect the measurement. Given the dynamic nature of the produced magnetic field and the noise introduced during DA conversion, it is extremely difficult for the sensor to record MI signals that equal the recorded one.

To validate, we randomly choose five samples from five devices, and obtain 10 replayed samples for each. Although we try our best to obtain a similar replayed signal, none of them matches with the enrolled fingerprints. We believe that is because the fingerprint discrepancy caused by the CPU hardware is subtle and the differences as well as noises introduced during the replay attack is likely to ruin such subtle characteristics. Thus, replay attacks targeted at DeMiCPU are challenging to perform even at a single point and the difficulty will increase dramatically with the increasing of testing sensors.

7.1.2 Mimicry Attack. The mimicry attack utilizes a similar device to imitate the target device by manipulating the software or configurations of the attack device. To impersonate the target device, the attack device has to precisely learn and mimic the fingerprint of the victim. However, the essential discrepancies of DeMiCPU fingerprints originate from the hardware of CPU modules. Manipulating software or configurations may alter the CPU fingerprint but the mapping between the configurations and the fingerprint is difficult to profile. As a result, the mimicry is likely to be unsupervised. In addition, according to our observations, the fingerprint discrepancy caused by the hardware of the CPU module is subtler compared with that caused by configurations. Thus, mimicry attack is not likely to make the attack device's fingerprint exactly the same as the one of the target device.

In summary, given the low frequency nature and the high precision of DeMiCPU, we believe it is difficult for adversaries to launch either a replay attack or a mimicry attack against DeMiCPU.

7.2 Limitation

Authentication Point. DeMiCPU that relies on one sensor requires the test point within a 16 *mm* range, which may affect the usability. A significant displacement of the DeMiCPU sensor from the CPU module may lead to failure in identification. However, we envision it can be addressed by exploiting a sensor array, which shall effectively reduce the requirement of test points and enlarge the fingerprinting area.

Long-term Consistency. We conducted our experiments over 30 days. However, a smart device usually can be used for years and it may experience changes due to aging, which in turn may change the features gradually. For example, the number of available CMOS transistors in the CPU may decrease due to the hardware aging. Nevertheless, we assume that we can compensate the aging by postulating a fingerprint slow updating technique: We update the fingerprints in the database occasionally if the current fingerprint is still classified to the legitimate user yet a small constant offset is detected, such that slow changes can be compensated.

User Process Suppress. DeMiCPU employs a higher priority for stimulation compared with other user processes. As a result, other user applications will be suppressed during fingerprinting. However, as DeMiCPU stimulation only lasts for 0.6 s, we argue it is relatively short and might be acceptable for most applications without affecting user experience.

Firmware-update Resistance. The firmware and CPU microcode of a smart device can be updated in accordance with requirements. During our experiments, the devices were kept natural and haven't been updated intentionally. As the firmware and CPU microcode may affect the execution of CPU instructions, they may have impact on DeMiCPU fingerprinting. We remain it as the future work.

8 RELATED WORK

Device Fingerprinting. Fingerprint is one of the most common biometrics in user identification [24, 35]. The same concept was extended to device identification by the US government in 1960s to identify and track unique mobile transmitters [27]. Since then, much effort has been devoted to identifying network devices by building a fingerprint out of their software or hardware. In terms of software-based fingerprint, the combination of chipsets, firmware and device drivers [15], timing interval of probe request frames [12], patterns of wireless traffic [33], and browser properties [46], can be used to identify devices. The downside of these methods is that fingerprints will change once device configuration or user behavior changes. Hardware-based approaches fingerprint a device through their physical components or properties. Clock skews [26, 34], radio frequency (RF) discrepancy at the waveform [20, 36, 41] or modulation [5] levels are well explored to identify wireless devices such as Wi-Fi routers. Mobile device fingerprinting utilizes the difference in hardware compositions [34, 40] or components such as accelerometers [13, 42], gyroscopes [2], microphones [11, 48], speakers [47], cameras [14, 29], Bluetooth implementation [1], or some of them in combination [4, 21]. The advantage of hardware-based device fingerprinting is that fingerprints are generated essentially from manufacture discrepancies, which can remain stable during the lifecycle of the device and are difficult to mimic.

EMI Leakage Based Side-channels. The use of EMI leakage as a side-channel has been widely investigated. This work [17] extracts the key of RSA software implementation on a Lenovo laptop using a near-field magnetic probe with a frequency around 100 kHz. Vaucelle et al. [43] detect the existence of ambient electromagnetic fields using a magnetometer bracelet with a frequency of up to 50 kHz. DOSE [9] detects the usage of electrical appliances by monitoring device EMI radiations with an expensive EMI measurement equipment. Magnifisense [44] recognizes the electrical appliance usage using a wrist-worn magnetic sensor and a set of data acquisition device, with a sampling rate of 16-bit resolution at 44.1 kHz. ZOP [8] utilizes electromagnetic emanations generated by computing systems during program execution to track a program's execution path and generate profiling information.

DeMiCPU is inspired by the aforementioned work and utilizes the natural discrepancies existing in CPU modules. Given the fact that a CPU module is indispensable for almost all mobile or smart devices, DeMiCPU makes a more universal method compared with aforementioned built-in sensor based approaches.

9 CONCLUSION AND FUTURE WORK

In this paper, we propose DeMiCPU, an effective device fingerprinting approach utilizing the unique features of magnetic induction (MI) signals generated from CPU modules, as a result of hardware discrepancy. We evaluate DeMiCPU with 90 mobile devices, including 70 laptops and 20 smartphones. The results show that DeMiCPU can achieve 99.1% precision and recall on average and 98.6% precision and recall for 30 identical devices, with a fingerprinting time of 0.6 s. Both precision and recall can be further improved to 99.9% with multi-round fingerprinting.

Future directions include exploring a larger study to confirm the scalability of DeMiCPU.

ACKNOWLEDGMENTS

We thank all anonymous reviewers for their insightful comments on this paper. This work is supported by China NSFC Grant 61702451, ZJNSF Grant LGG19F020020, and the Fundamental Research Funds for the Central Universities 2019QNA4027.

REFERENCES

[1] Aksu, H., Uluagac, A. S., and Bentley, E. Identification of wearable devices with bluetooth. *IEEE Transactions on Sustainable Computing* (2018).

[2] Baldini, G., Steri, G., Dimc, F., Giuliani, R., and Kamnik, R. Experimental identification of smartphones using fingerprints of built-in micro-electro mechanical systems (mems). *Sensors 16*, 6 (2016), 818.

[3] Bellovin, S. M., and Merritt, M. Cryptographic protocol for secure communications, Aug. 31 1993. US Patent 5,241,599.

[4] Bojinov, H., Michalevsky, Y., Nakibly, G., and Boneh, D. Mobile device identification via sensor fingerprinting. *arXiv preprint arXiv:1408.1416* (2014).

[5] Brik, V., Banerjee, S., Gruteser, M., and Oh, S. Wireless device identification with radiometric signatures. In *MobiCom* (2008), ACM, pp. 116–127.

[6] Brown, G. *FEAST*, January 2017. https://github.com/Craigacp/FEAST.

[7] Bullock, J. *LibXtract*, July 2014. http://jamiebullock.github.io/LibXtract/documentation/.

[8] Callan, R., Behrang, F., Zajic, A., Prvulovic, M., and Orso, A. Zero-overhead profiling via em emanations. In *ISSTA* (2016), ACM, pp. 401–412.

[9] Chen, K.-Y., Gupta, S., Larson, E. C., and Patel, S. Dose: Detecting user-driven operating states of electronic devices from a single sensing point. In *PerCom* (2015), IEEE, pp. 46–54.

[10] Cleveland, T. L. Bi-directional power system for laptop computers. In *APEC* (2005), vol. 1, IEEE, pp. 199–203.

[11] Das, A., Borisov, N., and Caesar, M. Do you hear what i hear?: Fingerprinting smart devices through embedded acoustic components. In *CCS* (2014), ACM, pp. 441–452.

[12] Desmond, L. C. C., Yuan, C. C., Pheng, T. C., and Lee, R. S. Identifying unique devices through wireless fingerprinting. In *WiSec* (2008), ACM, pp. 46–55.

[13] Dey, S., Roy, N., Xu, W., Choudhury, R. R., and Nelakuditi, S. Accelprint: Imperfections of accelerometers make smartphones trackable. In *NDSS* (2014).

[14] Dirik, A. E., Sencar, H. T., and Memon, N. Digital single lens reflex camera identification from traces of sensor dust. *IEEE Transactions on Information Forensics and Security 3*, 3 (2008), 539–552.

[15] Franklin, J., McCoy, D., Tabriz, P., Neagoe, V., Randwyk, J. V., and Sicker, D. Passive data link layer 802.11 wireless device driver fingerprinting. In *USENIX Security* (2006), vol. 3, pp. 16–89.

[16] Gartner. *Gartner Forecasts Flat Worldwide Device Shipments Until 2018*, January 2017. http://www.gartner.com/newsroom/id/3560517.

[17] Genkin, D., Pachmanov, L., Pipman, I., and Tromer, E. Stealing keys from pcs using a radio: Cheap electromagnetic attacks on windowed exponentiation. In *CHES* (2015), Springer, pp. 207–228.

[18] Getz, R., and Moeckel, B. Understanding and eliminating emi in microcontroller applications. *National Semiconductor* (1996).

[19] Geurts, P., Ernst, D., and Wehenkel, L. Extremely randomized trees. *Machine learning 63*, 1 (2006), 3–42.

[20] Hall, J., Barbeau, M., and Kranakis, E. Radio frequency fingerprinting for intrusion detection in wireless networks. *IEEE Transactions on Defendable and Secure Computing 12* (2005), 1–35.

[21] Hupperich, T., Hosseini, H., and Holz, T. Leveraging sensor fingerprinting for mobile device authentication. In *Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2016, pp. 377–396.

[22] Instrument, T. *Integrated Fluxgate Magnetic Sensor IC for Open-Loop Applications*, March 2016. https://www.ti.com/product/DRV425.

[23] Instruments, T. *MSP430F5529 LaunchPad Development Kit*, April 2017. http://www.ti.com/lit/ug/slau533d/slau533d.pdf.

[24] Jain, A. K., Hong, L., Pankanti, S., and Bolle, R. An identity-authentication system using fingerprints. *IEEE 85*, 9 (1997), 1365–1388.

[25] Keysight. *U2541A 250kSa/s USB Modular Simultaneous Data Acquisition*, June 2017. https://tinyurl.com/yb5r768y.

[26] Kohno, T., Broido, A., and Claffy, K. C. Remote physical device fingerprinting. *IEEE Transactions on Dependable and Secure Computing 2*, 2 (2005), 93–108.

[27] Langley, L. E. Specific emitter identification (sei) and classical parameter fusion technology. In *WESCON* (1993), IEEE, pp. 377–381.

[28] Le Sueur, E., and Heiser, G. Dynamic voltage and frequency scaling: The laws of diminishing returns.

[29] Lukas, J., Fridrich, J., and Goljan, M. Digital camera identification from sensor pattern noise. *IEEE Transactions on Information Forensics and Security 1*, 2 (2006), 205–214.

[30] Marr, D., Binns, F., Hill, D., Hinton, G., Koufaty, D., et al. Hyper-threading technology in the netburst® microarchitecture. *Hot Chips* (2002).

[31] Mondri, R., and Bitan, S. Inspected secure communication protocol, Sept. 1 2009. US Patent 7,584,505.

[32] Nguyen, K. T., Laurent, M., and Oualha, N. Survey on secure communication protocols for the internet of things. *Ad Hoc Networks 32* (2015), 17–31.

[33] Pang, J., Greenstein, B., Gummadi, R., Seshan, S., and Wetherall, D. 802.11 user fingerprinting. In *MobiCom* (2007), ACM, pp. 99–110.

[34] Radhakrishnan, S. V., Uluagac, A. S., and Beyah, R. Gtid: A technique for physical device and device type fingerprinting. *IEEE Transactions on Dependable and Secure Computing 12*, 5 (2015), 519–532.

[35] Ratha, N. K., Bolle, R. M., Pandit, V. D., and Vaish, V. Robust fingerprint authentication using local structural similarity. In *WACV* (2000), IEEE, pp. 29–34.

[36] Remley, K., Grosvenor, C. A., Johnk, R. T., Novotny, D. R., Hale, P. D., McKinley, M., Karygiannis, A., and Antonakakis, E. Electromagnetic signatures of wlan cards and network security. In *ISSPIT* (2005), IEEE, pp. 484–488.

[37] Solomon, D. A., Russinovich, M. E., and Ionescu, A. *Windows internals*. Microsoft Press, 2009.

[38] Suleiman, D., Ibrahim, M., and Hamarash, I. Dynamic voltage frequency scaling (dvfs) for microprocessors power and energy reduction. In *ICEEE* (2005).

[39] Travers, M. Cpu power consumption experiments and results analysis of intel i7-4820k.

[40] Uluagac, A. S., Radhakrishnan, S. V., Corbett, C., Baca, A., and Beyah, R. A passive technique for fingerprinting wireless devices with wired-side observations. In *CNS* (2013), IEEE, pp. 305–313.

[41] Ureten, O., and Serinken, N. Wireless security through rf fingerprinting. *Canadian Journal of Electrical and Computer Engineering 32*, 1 (2007), 27–33.

[42] Van Goethem, T., Scheepers, W., Preuveneers, D., and Joosen, W. Accelerometer-based device fingerprinting for multi-factor mobile authentication. In *ESSoS* (2016), Springer, pp. 106–121.

[43] Vaucelle, C., Ishii, H., and Paradiso, J. A. Cost-effective wearable sensor to detect emf. In *CHI* (2009), ACM, pp. 4309–4314.

[44] Wang, E. J., Lee, T.-J., Mariakakis, A., Goel, M., Gupta, S., and Patel, S. N. Magnifisense: Inferring device interaction using wrist-worn passive magneto-inductive sensors. In *UbiComp* (2015), ACM, pp. 15–26.

[45] Wikipedia. *Ampère's circuital law*, May 2018. https://en.wikipedia.org/wiki/Amp%C3%A8re%27s_circuital_law.

[46] Yen, T.-F., Xie, Y., Yu, F., Yu, R. P., and Abadi, M. Host fingerprinting and tracking on the web: Privacy and security implications. In *NDSS* (2012).

[47] Zhou, Z., Diao, W., Liu, X., and Zhang, K. Acoustic fingerprinting revisited: Generate stable device id stealthily with inaudible sound. In *CCS* (2014), ACM, pp. 429–440.

[48] Zou, L., He, Q., and Wu, J. Source cell phone verification from speech recordings using sparse representation. *Digital Signal Processing 62* (2017), 125–136.

A APPENDIX

A.1 Experimental Device

Table 3: Experimental devices and their detailed specifications. A total of 90 devices are used, including 70 laptops and 20 smartphones. Among them, 1-30, 31-33, 50-51, 84-85 and 88-89 are of the same model and OS respectively.

No.	Manuf.	Model	OS	CPU Parameters			
				Model	Core Number	Thread Number	Test Point
1-30	Lenovo	ThinkPad T430	Win 7	i5-3320M	2	4	S
31-33	Lenovo	ThinkPad T440p	Win 7	i5-4210M	2	4	S
34	Lenovo	G480	Win 7	i5-3210M	2	4	R
35	Lenovo	G480	Win 10	i5-3210M	2	4	R
36	Lenovo	ThinkPad X201	Win 10	i5-540M	2	4	F6
37	Lenovo	ThinkPad T440	Debian	i7-4500U	2	4	N
38	Lenovo	ThinkPad W520	GNOME	i7-2760QM	4	8	E
39	Lenovo	ThinkPad Edge E431	Win 10	i7-3632QM	4	8	S
40	Lenovo	ThinkPad Edge E530	Win 10	i5-3210M	2	4	S
41	Lenovo	IdeaPad Y470	Win 7	i5-2450M	2	4	E
42	Lenovo	IdeaPad Y485	Win 7	A8-4500M	4	4	F5
43	Lenovo	Yoga2 13	Win 10	i5-4210U	2	4	F4
44	Lenovo	Yoga 710	Win 10	i5-6200U	2	4	O
45	Lenovo	U430P	Win 10	i5-4200U	2	4	F1
46	Lenovo	Erazer Z410	Win 10	i7-4702MQ	4	8	6
47	Lenovo	E47a	Win 7	i5-2520M	2	4	S
48	Lenovo	X200 7455	GNOME	Intel P8600	2	2	F
49	Lenovo	R720	Win 10	i5-7300HQ	4	4	7
50-51	Apple	MacBook Air A1466	OS x	i5-4260U	2	4	W&E
52	Apple	MacBook Pro A1707	OS x	i7-6920HQ	4	8	W&E
53	Apple	MacBook Pro A1502	OS x	i5-4278U	2	4	C
54	Dell	Inspiron N4050	Win 7	i3-2350M	2	4	F
55	Dell	Inspiron N5110	Win 7	i5-2450M	2	4	F
56	Dell	Inspiron 14 7460	Win 10	i5-7200U	2	4	6
57	Dell	Inspiron 15R 5520	Win 10	i5-3210M	2	4	Fn
58	Dell	Inspiron 15 7559	Win 10	i5-6300HQ	4	4	F6
59	Dell	Latitude E4300	Win XP	Intel SP9400	2	2	F
60	Dell	Latitude E7440	Win 10	i5-4200U	2	4	E&R
61	Dell	XPS13	Win 10	i5-3317U	2	4	6
62	Dell	XPS14 L421X	Win 10	i7-3537U	2	4	4
63	Asus	Eee PC 1201HA	Win 7	Intel Z520	1	2	A
64	Asus	N46V	Win 8.1	i5-3210M	2	4	B&N
65	Asus	X450EI323VC-SL	Win 10	i5-3230M	2	4	F
66	Acer	V5-471G	Win 7	i5-3337U	2	4	D
67	HP	TPN-Q173	Win 10	i5-6300HQ	4	4	Backspace
68	MSI	MS16-H8	Win 10	i7-6700HQ	4	8	Scroll Lock
69	Sony	SVT131A11T	Win 7	i5-3317U	2	4	X
70	Sony	SVT131A11T	Win 10	i5-3317U	2	4	X
71	Mi	5	Android 6.0	Snapdragon 820	4	4	BVK*
72	Mi	5S	Android 6.0	Snapdragon 820	4	4	BVK*
73	Huawei	Honor 5X	Android 5.1	Snapdragon 616	8	8	BVK*
74	Huawei	Honor 8	Android 6.0	Kirin 950	8	8	BVK*
75	Huawei	Honor V8	Android 6.0	Kirin 950	8	8	BVK*
76	Huawei	P9	Android 6.0	Kirin 955	8	8	BVK*
77	LG	Nexus 5	Android 4.4	Snapdragon 800	4	4	BVK*
78	LG	Nexus 5X	Android 6.0	Snapdragon 808	4	4	BVK*
79	Vivo	X7	Android 5.1	Snapdragon 652	4	4	BVK*
80	Samsung	Galaxy S6	Android 5.0	Exynos 7420	8	8	BVK*
81	Apple	iPhone 6	iOS 10.2.1	Apple A8	2	2	BPK•
82	Apple	iPhone 6	iOS 11.0.3	Apple A8	2	2	BPK•
83	Apple	iPhone 6 Plus	iOS 11.1.1	Apple A8	2	2	BPK•
84-85	Apple	iPhone 6s	iOS 10.3.3	Apple A9	2	2	BPK•
86	Apple	iPhone 6s	iOS 10.2.1	Apple A9	2	2	BPK•
87	Apple	iPhone 6s	iOS 11.2.1	Apple A9	2	2	BPK•
88-89	Apple	iPhone SE	iOS 11.2.1	Apple A9	2	2	BPK•
90	Apple	iPhone 7 Plus	iOS 10.3.3	Apple A10	4	2	BPK•

*BVK = Beside Volume Key

•BPK = Beside Power Key