A New Side-Channel Vulnerability on Modern Computers by Exploiting Electromagnetic Emanations from the Power Management Unit

Nader Sehatbakhsh*, Baki Berkay Yilmaz, Alenka Zajic, and Milos Prvulovic
Georgia Institute of Technology
**nader.sb@gatech.edu*

***Abstract*—This paper presents a new *micro-architectural* vulnerability, which is created by power management units of modern computers and can be exploited through electromagnetic, and potentially other, side-channels. The key observations that enable us to discover this side-channel are: 1) in an effort to manage and minimize power consumption, modern microprocessors have a number of possible operating modes (*power states*), in which various sub-systems of the processor are powered down, 2) for some of the transitions between power states, the processor also changes the operating mode of the voltage regulator module (VRM) that supplies power to the affected sub-system, and 3) the electromagnetic (EM) emanations from the VRM are heavily dependent on its operating mode. As a result, these state-dependent EM emanations create a side-channel that can reveal which programs are currently executing, and potentially other sensitive information about the executed programs.**

To demonstrate the feasibility of exploiting this vulnerability, we create a *covert channel* that uses changes in the processor's power states to *exfiltrate* sensitive information from a system that is otherwise secured and completely isolated (*air-gapped*), and then receives that information using a compact, inexpensive receiver placed in proximity to the system. To demonstrate the *severity* of this vulnerability, we also show that information can be successfully exfiltrated even if the receiver is *several meters* away from the system, and even if the system and the receiver are separated by a wall. Compared to the state-of-the-art, the proposed covert channel has >3x higher bit-rate. Finally, to demonstrate that this new vulnerability is not limited to being used as a covert channel, we demonstrate how it can be used for attacks such as *keystroke logging*.

***Keywords*-Side-Channel; Microarchitecture Vulnerability; DVFS, Power Management.**

I. Introduction

Modern computers are architected primarily for *power-performance* and security has often been considered as a secondary concern. However, neglecting security as a primary design factor has led to several recent discoveries of new side-channel vulnerabilities [1]–[7] which, if exploited, could severely compromise the entire system and result in potentially huge economic losses or even greater damages.

A ubiquitous power-performance management strategy in modern computers is the dynamic voltage and frequency scaling (DVFS) [8], which is widely supported in modern computer systems. DVFS, typically implemented and managed by a power management unit (PMU), provides the capability to scale the voltage and frequency settings of the processors dynamically depending on the workload. Much of the work on the design and implementation of PMUs have focused only on optimizing energy consumption and overall efficiency of algorithms that manage the power-performance tradeoff, while security aspects of PMUs in modern systems have, unfortunately, received relatively little attention.

In this paper, we show that the modern PMUs include design and implementation features that can create new, previously unexplored, physical side-channel vulnerabilities which can be exploited even from *outside* of the system by leveraging the underlying electromagnetic emanations of the PMU. Compared to prior attacks on PMUs [7], [9], [10], which could be carried out *remotely*, the proposed side-channel has a different attack model - prior attacks are practical mostly for cloud servers where unrelated (potentially mutually hostile) services/VMs end up being co-located on the same physical server, while the new vulnerability we present is mostly applicable to mobile devices, computers used in offices adjacent to public spaces, etc. Given these differences, our findings are important since the discovered side-channel is applicable even when a computer is highly secured from untrusted users, e.g., when it is physically isolated (air-gapped) from other networks. Moreover, as we will show in this paper, compared to the existing physical side/covert channel attacks [7], [10]–[15] that can successfully attack an isolated system, the discovered side-channel can exfiltrate data with much higher data-rate, and in many cases, it can be received from much further distances.

The *key factors* that create this new side-channel are the following: when the processor is active, it consumes more power, which requires a higher voltage and draws more current from its voltage regulator module (VRM). That, in turn, results in strong EM emanations at the VRM's switching frequency. Conversely, when the processor is idle, it requests a lower voltage and consumes very little current from its VRM, which results in much weaker EM emanations from the VRM. In effect, the processor's idleness state is *amplitude-modulated* onto the EM signal emanated by its VRM, which can be received at some distance. Thus, these activity-dependent analog side-channel signals can *reveal* the power state of the processor and further *leak* sensitive information about the system or its applications.

Experimental evaluation in this paper shows that an at-

2378-203X/20/$31.00 ©2020 IEEE
DOI 10.1109/HPCA47549.2020.00020

tacker can exploit this information in at least two ways: *(i)* using these side-channel EM emanations, a reliable *covert channel* can be established to *exfiltrate* sensitive data from an air-gapped computer where bits of data can be transmitted by switching the processor between active and idle states, and *(ii)* an attacker can systematically infer sensitive information from the system by using this side-channel in *key-logging* (i.e., secretly finding which keys are pressed on a keyboard), website and/or application fingerprinting, etc.

The subject systems in this evaluation are a variety of commodity laptops, and *the discovered side-channel exists on all these laptops regardless of the vendor, OS, and/or processor*. Furthermore, our evaluation shows that this vulnerability can be exploited using a compact and inexpensive software-defined radio, together with a tiny probe, as the receiver, with the entire setup being pocket-sized (easy to conceal) and costing less than $30. Finally, our evaluation also shows that, when using a compact loop antenna such that the entire setup can be hidden in a briefcase, the signal can be received from several meters away, or when the receiver is separated from the target by an office wall.

This paper makes the following contributions:

- Describes a new physical side-channel vulnerability that exploits the signals (e.g., EM emanations) produced by the system's voltage regulator module to infer the processor's *power-states*,
- A proof-of-concept exploitation of this vulnerability by creating a covert channel with low bit-error-rate, and with a high data-rate to exfiltrate data from an air-gapped computer,
- A practical demonstration of data exfiltration at a distance and through a wall in an office environment.
- A proof-of-concept design and implementation of a keystroke logging framework by exploiting the proposed side-channel.

The rest of the paper is organized as follows: § II provides a brief background on the processor's power management, voltage regulator functionality, and side-channel signals, § III presents the side-channel which leaks the processor's activity level through the VRM's EM emanations, § IV describes our proof-of-concept implementation of data exfiltration from an air-gapped system using this side-channel, § V presents our design and implementation of a keylogging framework that leverages this EM emanation to detect which key was pressed by a user, § VI is an overview of related work and, finally, § VII summarizes our conclusions.

II. Background

Side-Channel Signals. These signals are unintentionally generated as an artifact during computation. Side-channels can be classified into two categories: *Digital/Micro-architectural* and *Analog/Physical*. Digital side-channels typically rely on *shared hardware* resources in the computer. Examples of these channels include caches [16]–[22], micro-architectural units [23], [24], DRAM and memory bus [25], [26], processor frequency settings [10], branch predictors [5], [27], [28], GPUs [29], TLBs [6], etc. Physical side-channels, however, rely on physical characteristics of the system such as EM emanations [14], [30]–[36], variation in power consumption [37]–[41], sound/acoustic [42]–[47], temperature [11], [48], [49], chassis potential variation [50], crosstalk [51], peripherals [52], [53], etc.

The main difference between the two categories is that the former (digital), typically, can be measured *within* the system (i.e., by another process), while the latter often requires some physical proximity to the device for measurements or, alternatively, needs to access some sensors on the board to read the desired value. Due to this limitation, physical side-channel attacks often have much lower transmission rate. However, unlike digital side-channels, physical side-channels are much more challenging to mitigate, and are often too expensive and impractical to eliminate. This, makes physical side-channels particularly attractive in scenarios where the system is well-protected from digital side-channels through isolation and/or partitioning [54]. For example, a popular method for strong isolation is creating an *air-gap*, where all the computer's connections to the outside world are either disabled or monitored. However, data can still be exfiltrated using a physical side-channel.

Power Management. To improve energy efficiency, modern computer systems, especially mobile ones where energy efficiency directly affects battery life, employ a number of power-management techniques. One of the most popular such techniques is dynamic voltage-frequency scaling (DVFS) [8], [55], where the processor's clock frequency can be adjusted dynamically depending on the level of performance that is required. The reduced clock frequency reduces power consumption by executing fewer instructions (and thus spending less energy) per unit time. Furthermore, the speed at which the processor's circuitry can operate is dependent on its operating voltage, so the processor's operating voltage can be lowered as its operating frequency is reduced, which (dramatically) reduces the energy consumed per instruction executed. Another very popular technique consists of placing unused units within the processor into a low-power state, typically by no longer clocking the unit (clock gating [56]), but in some cases also by further reducing the unit's voltage level or even completely powering it down. Most modern processors deploy both sets of techniques. For example, the Demand Based Switching (DBS) [57] technology in Intel's processors provides the processor with a number of *performance states* (P-states), each with a different voltage-frequency value, and also a number of *processor states* (C-states) which correspond to different levels of low-power idleness. Recent processors can have more than 10 different P-states, where P0 is the highest-performance

state, and higher state numbers correspond to lower voltage-frequency settings (and thus lower performance). For Intel processors up to Haswell/Broadwell architecture, the desired P-state is specified by the operating system, by writing the corresponding value into a special processor register, and the processor's hardware simply implements the specified voltage-frequency settings [58]. More recently (starting with the Skylake architecture [59]), the operating system can leave the control of the P-states to the processor's hardware (called Speed-Shift technology [57] by Intel), and this is the default behavior in recent operating system releases because it enables more rapid P-state adjustments as the processor's workload changes.

Whereas P-states allow management of the tradeoff between the processor's performance and energy consumption while active, the *C-states* are a set of *low-power* modes that the processor can switch to when it is idle. The C0 state corresponds to the processor's normal operation (execution of instructions), whereas C1, C2, etc. states correspond to idleness with increasing levels of clock- and power-gating for the units within the processor. Thus higher-numbered C-states save more energy while the processor is idle, but also take more time to "wake up" the processor. Typically, states C1 through C3 only apply clock-gating, C4 through C6 reduce the voltage, and new Enhanced C-states can do both at the same time. The transition between C-states relies on a set of sensors that monitor utilization of the cores, but the actual algorithm for choosing when and which C-state to use is not publicly available (and may change from one generation of processors to the next).

The P- and C-states are enabled by default but, on all recent laptops we examined, the BIOS has settings for disabling them (at significant cost in terms of power-efficiency). Further, P-states can also be controlled through the OS using tools provided by the kernel (e.g., *cpufrequtils* in Ubuntu).

Voltage Regulator Module (VRM). The VRM is the module that actually supplies power to the cores. The processor uses a set of Voltage Identification (VID) hardware signals to inform the VRM which voltage-level to provide [60]. The VRM is typically an integrated circuit that is soldered onto the system's motherboard, but in some recent processors (e.g., Intel's Haswell architecture) the voltage regulator is integrated into the processor's package (an Integrated Voltage Regulator, or IVR), or even into the processor's silicon die (a Fully Integrated Voltage Regulator, or FIVR).

In laptops (and desktops, too), the most commonly used style of a voltage regulator is a *Buck* [61] converter (also called step-down converter). It is a DC-to-DC power converter which is connected at its input to a higher-voltage DC supply (e.g., the laptop's battery pack or AC-power adapter, which typically supply 10-20 V), and outputs a lower voltage, typically between 0.7 V to 1.4 V to its load.

Intuitively, a Buck converter consists of a capacitor at its output that it tries to keep filled to the desired voltage level. Since the load draws current from this capacitor, the charge it holds drains over time, causing its voltage to drop. To compensate for this drop, the converter periodically connects the capacitor to its input, causing a burst of current that replenishes the capacitor's charge, thus bringing the output voltage back to the desired level. The amount of time between these replenishments (***switching period***) in computer-system VRMs is typically a few (1-4) microseconds.

The VRM must be capable of supplying enough current under maximum-load conditions, so its switching period must be short enough that even the maximum output current does not drain the output capacitor below the minimum required level. At low load currents (e.g., when the processor core is idle), however, the voltage regulator becomes less efficient – it switches just as often as under full load, thus wasting a similar amount of power on switching losses, while the power actually provided to the load is very small, so the switching losses become a much larger fraction of the overall power consumption. Since the low load current also means that the VRM's output capacitor is still almost fully charged at the end of each switching period, a typical VRM improves its low-load efficiency using a technique called ***phase shedding*** [62]–[64], where for some of the switching periods the VRM does not switch, skipping the replenishment of the still-almost-full capacitor and saving the energy that would have been wasted to do so.

Electromagnetic Side-channel Emanations. The bursts of current during VRM's switching activity result (according to Faraday's law) in changes in the EM field around the VRM. Since the VRM's switching is periodic (with a switching period T), the changes in the EM field are also periodic at frequency $f = 1/T$, i.e., the switching creates an EM signal that has a very strong component at frequency f and, because the current bursts are square-wave-shaped rather than sinusoidal, at frequencies that are multiples of f. Additionally, the current during VRM's switching is very high compared to most other flows of electrical current in a computer system, so the resulting EM emanations at f and its harmonic frequencies tend to be very strong.

In the frequency-domain, the signal emanated by a VRM forms "spikes" at frequency, $f = 1/T$, and its integer multiples, where T is the switching period of the VRM. For typical VRMs, T is between 1 to 4 microseconds (i.e., spectral spikes at a frequency in the 250 KHz to 1 MHz range and also its harmonics). Thus, these signals can be received by an inexpensive radio receiver by tuning it to f or, if there is significant interference at that frequency, at one of its harmonics. Moreover, these frequency spikes are very prominent when the VRM is under a high-load, and much weaker when it is under low-load. Thus, from the perspective of a radio receiver tuned to frequency f, the received signal

```
1  void microbenchmark(int t1, int t2){
2    int dummy;
3    while (1) {
4      // active state
5      for(int i=0; i<t1; i++)
6        dummy += dummy + i;
7      // idle state
8      usleep(t2);
9    }
10 }
```

Figure 1. The micro-benchmark used in this paper to generate ACTIVE and IDLE states for the processor to create EM side-channel signals.

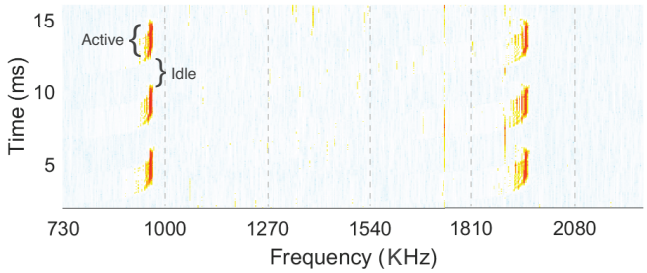

Figure 2. Alternation between *active/idle* states and the spikes (and its first harmonic) created by the emanated EM signals from PMU shown in the frequency domain over time.

is strongly amplitude-modulated by the processor's activity state (active vs. idle) which creates an analog side-channel.

III. Side-Channel Vulnerability on the Power Management States

To demonstrate that switching between *active* and *idle* power states (i.e., P-states and C-states) creates a distinguishable signal, we perform a simple experiment where the system is alternating between an idle and an active state, and the received EM signal is analyzed to find whether this alternation can also be found in the signal. As described in § II, due to the way VRM operates, during the *active* state, we expect to observe strong (in terms of magnitude) spikes, and weak spikes during the *idle* states. We use a program shown in Figure 1 to create such alternations. This program creates an infinite loop that continuously performs some activity (e.g., addition) for a while followed by a period of idleness. The duration of the active period is controlled by the value of t_1, while the duration of idleness periods is controlled by the value of t_2. Note that the actual activity (lines 5-6) can be any processor-intensive activity. Similarly, the idle periods can be implemented in any way that leads the operating system and the processor to believe that the processor will be inactive for a while.

The resulting EM signals emanated from the PMU is depicted in Figure 2. This figure shows how the measured signals change over time in the frequency domain (i.e. a *spectrogram*), where the higher intensity illustrates higher signal amplitude. A repeated pattern of strong/weak spikes is present, as expected for this side-channel. The frequency of the spikes also matches with the expected frequency of the PMU (i.e., around 970KHz) for the tested laptop[1]. To further examine the received signals, we changed the lengths of *active* and *idle* periods by changing t_1 and t_2, and observed that the length of the spikes (i.e., the red lines shown in Figure 2) *do* change as periods change.

To further confirm that these spikes are indeed related to the VRM and caused by the changes in the processor's power states (i.e., P- and C-states in Intel's processors), we repeat experiment with disabling/re-enabling of P-states and C-states in the system's BIOS (i.e., changing DVFS settings) and examine how that affects the received EM signals and their spectra. We found that, even when either C-states or P-states (but not both) are disabled, we observe a signal spectrum similar to that in Figure 2, i.e., the spikes in the spectrum appear and disappear (with no change in the data transmission rate). However, when both C-states and P-states are disabled, the spikes in the spectrum have a much stronger magnitude but are continuously present regardless of the program activity. This is consistent with our expectations - when the processor's management of power states is disabled, the processor is forced to operate at its nominal operating voltage and frequency regardless of its workload, even when the system is "idle" [2]. This forces the VRM to continuously remain in its high-power mode. The results of experiments where only C-states or only P-states were disabled are also consistent with our expectations - in those cases, the processor can still switch between idle and active states (e.g., C_0 and C_N for C-states, or P_N and P_M where $M > N$ for P-states). This observation indicates that, fundamentally, to observe this side-channel, the processor needs to be able to switch between at least one high-power and at least one low-power state (which can be different C-states, different P-states, or a combination of both).

Attack Model. Based on the observations above, an attacker, $\mathcal{A}$, can exploit these signals in two meaningful ways: ***(i)*** The attacker can create a covert channel by intentionally forcing the processor to alternate between periods of high activity and periods of idleness, according to the values of (secret) data bits she desires to exfiltrate. § IV describes, in detail, how such a covert channel can be created. ***(ii)*** The attacker can monitor the emanated signals to infer *(a)* whether the processor has become active or not. Such information can be particularly helpful to find, for example, whether a key is pressed. Further, the attacker can monitor these signals to infer *(b)* how long the processor was active to process a certain task. Such information, for example, can be used for website fingerprinting (i.e., by measuring how long it takes

[1]These emanations are around the clock frequency of *PMU* and not processor's clock frequency since it is created by PMU and not the CPU.

[2]When the system is idle while the C-states are disabled, it actually runs the operating system's "idle" process, usually an infinite loop, so the system's processor is not actually idle.

```
1  void transmitter(){
2   char bit; int dummy1;
3   FILE *file = fopen("secret", "r");
4   while ((bit = getc(file)) != EOF) {
5     if(bit == '1') {
6        for(int i=0; i<LOOP_PERIOD; i++)
7          dummy1 += dummy1 + i;
8          // keeping the processor active
9          usleep(SLEEP_PERIOD);}
10         // return-to-zero coding
11    else usleep(SLEEP_PERIOD * 2);
12  }
13  fclose(file);
14 }
```

Figure 3. The "transmitter" code for the covert channel communication.

to load a webpage, the attacker can infer which website was loaded). § V describes how this side-channel can be leveraged for keylogging in details.

IV. Covert Channel Communication

A. Transmitter Design

To create a covert channel, the *transmitter* (also called source or data sender) application, which has access to the secret data, should create the side-channel signal depending on this sensitive information. The transmitter code is shown in Figure 3. For each bit of data, depending on the value of the bit, the code either performs some activity for a while followed by a period of idleness (i.e., *return-to-zero* encoding [17]) or only a (longer) period of idleness. In this code, the duration of the active and idleness periods are controlled by the value of `LOOP_PERIOD` and `SLEEP_PERIOD` respectively.

Note that none of this code requires elevated (e.g., root-level) privileges, i.e., ***in our threat model***, the attacker's program runs as a user-level process that (i) has access to the data it desires to exfiltrate, but (ii) is denied access to any I/O that would allow it to send that data out of the system. Given the simplicity and brevity of the code, any number of programming languages can be used for this purpose, including most scripting languages or even shell scripts. Also, note that methods for creating such a malicious code inside an air-gap computer are abundantly represented in the research literature (e.g., advanced persistent threat [65]), although we do not discuss them in this paper.

The `LOOP_PERIOD` and `SLEEP_PERIOD` parameters in this code determine the bit-rate of the channel - in general, smaller values result in higher data rates. However, in practice, the bit-rate is limited by several practical constraints. First, the active period also includes the execution of the library and system code that implements the actual call to `usleep` and its house-keeping activity, so even when `LOOP_PERIOD` is zero the actual active period includes execution of tens or hundreds of instructions. The usable values of the `SLEEP_PERIOD` are also limited, mostly due to the granularity and precision of time measurement for `usleep` and the variability in the time needed to exit the idle state. Finally, the duration of the active and the idle phase should be roughly similar. Based on our experiments, we found that around 10μs is the limit below which the actual idleness period of $usleep()$ becomes highly variable[3].

B. Receiver Design

1) Signal Acquisition: In conventional communication systems, the transmitter, the receiver, and the signal transmitted between them, are all carefully engineered to achieve a low bit-error-rate (BER) while sustaining high throughput. This includes maintaining good synchronization between the transmitter and receiver to minimize insertion and deletion of bits, a sophisticated encoding of data bits into the amplitude and phase of the transmitted signal, etc. Furthermore, the carrier frequency at the transmitter is chosen carefully to support the desired range of distances between the transmitter and receiver. However, covert channels rely on signals that are produced unintentionally, thus, there is no control over the transmitter design, i.e., the carrier frequency and its stability, the range in which the signal's amplitude and phase can be changed depending on the activity, how stable the duration of these changes is, etc. Therefore, after the signal is received, the detection algorithm must deal with the problems of discovering when each transmitted bit begins and ends, changes in the signal's amplitude, etc.

As discussed in § III, the observed signal patterns when the transmitter code is executing is similar to Figure 2. The key observation from this figure is that the received signal in frequency domain behaves like on-off keying (OOK) for the considered frequency components. Therefore, signal power level for each bit will be enough to identify the received bit. Leveraging the knowledge that there exist many frequency components related to the transmitter activity, we acquire the signal as

$$\mathbf{Y}[n] = \sum_{k \in \mathcal{S}} \mathtt{abs}\left(\mathcal{F}_n[k]\right), \tag{1}$$

where $\mathbf{Y}[n]$ is the signal of interest, $\mathtt{abs}(\bullet)$ takes the absolute value of its argument, $\mathcal{S}$ is the set of considered frequency components,

$$\mathcal{F}_n[k] = \sum_{m=0}^{M-1} r[m - M + 1 + n] e^{-2i\pi km/M},$$

M is the size of the Fast Fourier Transform (FFT) function and $r[m]$ is m^{th} sample of the received signal. The goal here is to increase the difference in magnitude between bit 0 and bit 1 to minimize the error-rate of the covert communication.

An example for $\mathbf{Y}[n]$ is given in Figure 4 where we only used the fundamental frequency and its first harmonic from the actual signal. We also plot the time-interval for each

[3]Even the manual page for $usleep()$ states that the sleep time may be lengthened slightly due to other system activities and hence cause some randomness to the overall timing.

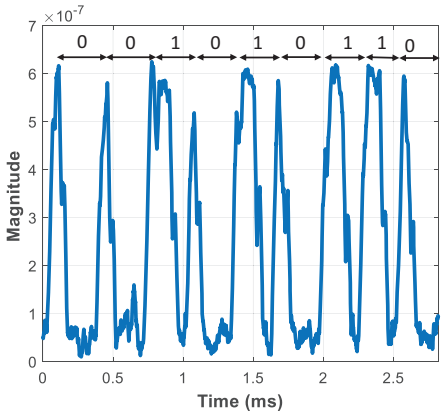

Figure 4. Average magnitude of the considered frequency components and the corresponding bit sequence.

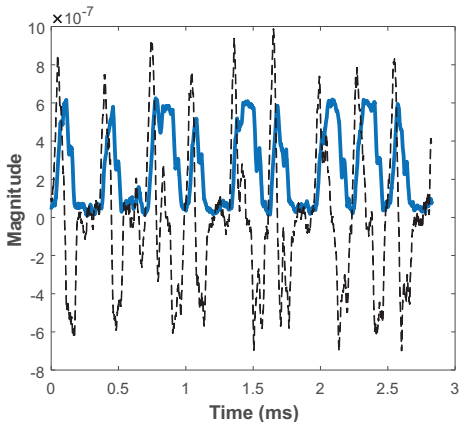

Figure 5. Obtaining the start points of each bit by exploiting that sharp increase whenever a bit is transmitted.

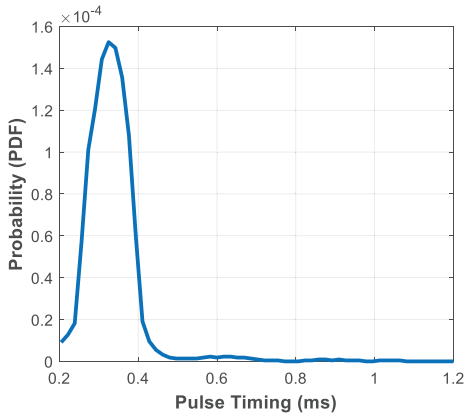

Figure 6. Pulse width variation of the covert communication system.

signal (called *signal timing*) and the transmitted bit. The main observations from this figure are the following:

- The signal exhibits a sharp increase whenever a new bit is transmitted, even when the bit is a zero, because processor activity is needed to execute the program code that cleans up at the end of a previous `usleep`, reads a new bit of data, and begins a new interval.
- The magnitudes are affected by not only the additive noise but also by the variations in the execution of the transmitter.
- Due to these variations (especially the sleep time), the duration of a signal that corresponds to one "transmitted" bit varies among instances of these bits, even when the bits have the same value.

It is a common practice for the conventional communication systems to use a *matched filter* and sample the filtered signal at each symbol (bit), but that approach assumes that the symbols have practically no variation in their duration, i.e., the transitions from symbol to symbol are synchronous with a highly stable clock, and can thus be re-created at the receiver accurately. We found that, when applying the matched filter approach to our received signal, the BER was high, and upon further investigation we found that the main reason for this is the asynchronous nature of the signal - the actual bit positions in the signal quickly become misaligned with the clock created by the receiver in an attempt to match the transmitter's symbol-rate. Therefore, we had to devise a more robust (but also more computationally intensive) method for determining the timing of each bit.

2) Signal Timing for the Covert Communication: We determine signal timing (i.e., the time interval for each bit) using *batch processing*, i.e., we determine the timing of the bit by examining not only the signal that corresponds to that bit, but, also by considering a number of bit periods that precede and follow it which, in turn, reduce error-rate significantly while adding negligible detection latency.

The first step for batch processing is to find the starting locations of each bit by knowing that the derivative on these edges is almost infinity. To mimic the derivative operation, we convolve the batch signal with a vector of length, l_d (which depends on the sampling-rate). Half of the vector is set to one and the rest is set to minus one. This convolution is followed by finding the local maximum points of the convolution. An example of the process is shown in Figure 5 where the black dotted line shows the result of the convolution operation. As seen from the figure, the resulting points of the convolution peak at the edges of $\mathbf{Y}[n]$ which indicate the starting points of a bit transmission.

Obtaining the starting points of the transmitted bits helps to find the expected signaling time of each bit. Next, we calculate the distances between the starting points of subsequent bits. The probability density function (PDF) of the distances between subsequent bits are given in Figure 6. This figure illustrates that the signal time has a Rayleigh distribution. The tails of the distribution indicate that some of the bit locations could not be captured because the distances between points have a positive-skewed distribution (which results in detection errors that will be discussed later).

Having the signaling time of the transmitted bits helps to fill the gaps that the detection algorithm could not find at its first attempt. We choose the signaling time of the covert

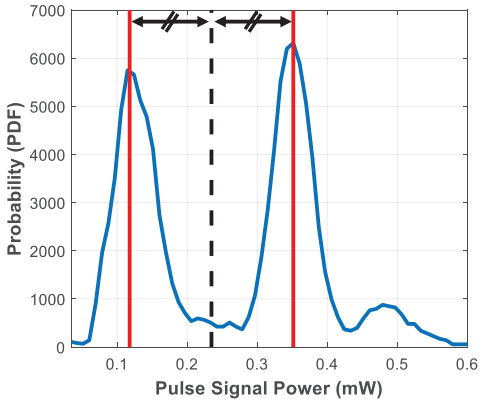

Figure 7. The power distribution of the pulses generated by the power management unit for IDLE (left) and ACTIVE (right) states.

communication as the point whose cumulative probability distribution equals to 0.5 since the distance distributions accumulate around this point and using the median value can minimize the false insertion and deletion rate of bits.

3) Signal Labeling Based on Average Signal Power: The variation in signal timing can also cause incorrect labeling of the received bits. The total power of the received signal could be high only because the usually very short active part of the signaling period has lasted much longer than usual. Therefore, while decoding the received signal, the algorithm also needs to take into account the variation of the signaling period's duration.

Specifically, the receiver detection algorithm utilizes the average power of the received signal samples for each bit. Let's assume we have the samples $s[n] \in \{0,\ 1,\ \cdots,\ N-1\}$ for the m^{th} received bit. The detection algorithm label the received bit as one if

$$\frac{1}{N}\sum_{n=0}^{N-1}|s[n]|^2 > \texttt{thr}, \tag{2}$$

where `thr` is the threshold value. However, the threshold value must be chosen carefully to minimize the error-rate. Figure 7 illustrates the distribution for the average signal magnitude for each bit. We observe that there exist two peaks which indicate the power of bit zero and bit one. Therefore, the algorithm selects the threshold as the mean of the points corresponding to these two peaks. This threshold selection process is also illustrated in Figure 7. Red lines in the figure correspond to the local maximum of average power distribution, and the dotted black line is the selected threshold value for the batch. The equal signs on the arrows mean that the distances indicated by these arrows are equal.

4) Bit Deletion/Insertion: This covert channel presents many challenges not only because its signaling periods are not explicitly synchronized, but also because of occurrences of other system activity, such as interrupts and micro-architectural events (e.g., page faults, cache misses). These events can cause errors in the signaling periods they occur

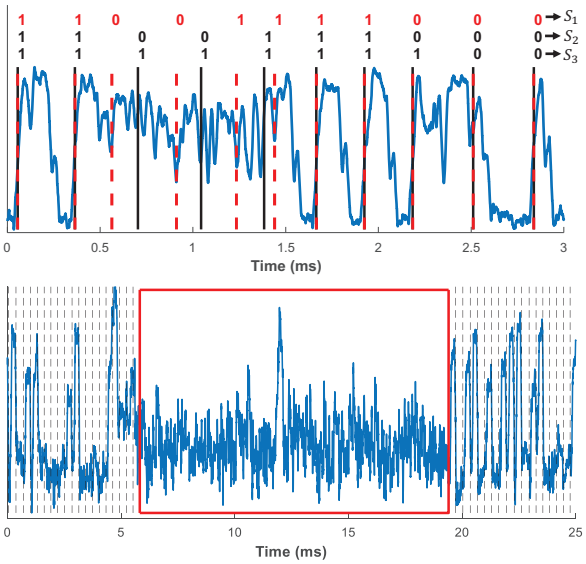

Figure 8. Bit deletion (top) and insertion (bottom) in the covert communication channel due to variations in signal timing.

in, and also, in a sort of "domino effect", make errors much more likely for other signaling periods in the same batch.

An example of the bit deletion and insertion is shown in Figure 8. In the figure (top), the blue line represents the received signal, black lines correspond to the estimation of the detection algorithm for the possible bit starting location. The dashed black lines (bottom) represent the starting location of a bit signal and the red box contains the region where the insertion occurs due to an interrupt. The actual bit starting points are given with red dotted lines (top). We observe that one of the bits is deleted. We also provide three sequences labeled with S_1, S_2 and S_3, which correspond to an actual transmitted bit sequence, actual transmitted bit sequence after deletion, and the estimated bit sequence, respectively. Here, the receiver's detection algorithm fails because the edges at the beginning of each transmitted bit completely disappear. The reason behind this disappearing is that other system activities get activated which suppresses the effect of the designed micro-benchmark. However, we observe that the deletion probability of the system is pretty low ($<$0.2%). Therefore, this problem can be addressed by employing even relatively simple error correcting codes in the "transmitter" application. In our experiments, we use a very simple (parity) code, which keeps our "transmitter" application simple enough to manually implement on a target machine in a few minutes.

C. Experimental Evaluation

1) Measurement Setup: To show the feasibility of exploiting this covert channel, we present our experimental results in two practically relevant scenarios: when a compact and stealthy receiver apparatus is placed in close proximity to the target system, and when a larger (briefcase-sized) antenna is

placed up to 2.5 meters away, and also when the antenna is in an adjacent room, 1.5 m away but with a structural 35 cm thick wall included in that distance.

We used a software-defined-radio (RTL-SDRv3 [66], commercially available for $25) for signal acquisition which is as large as a small flash drive. For close proximity measurements, we used a coin-shaped handmade 33-turn coil magnetic field probe with a radius of 5 mm which costs <$5 (no amplifier is used for these measurements). For the distance and non-line-of-sight (NLoS) measurements, we used the same SDR with a magnetic loop antenna (AOR-LA390 [67]) with a radius of 30 cm which costs $200, including a built-in 20dB amplifier.

For the *transmitter*, we used $usleep()$ for UNIX-based machines, and $Sleep()$ for Windows-based machines. The sampling-rate for the SDR was set to 2.4 million samples per second, which is the maximum this SDR is capable of. We used 1024 point FFT with maximum overlapping. The receiver's detection algorithm was implemented in MATLAB 2017-b. For synchronization between the transmitter and receiver at the start of the communication, the transmitter sends a pre-defined bit-stream of interleaved ones and zeros followed by a known short bit-stream of zeros only. The transmitter then sends a *preamble* to indicate the start of the transmission, and then sends the actual data. Depending on the requirement, the data can be sent in packets or continuously. Unless otherwise indicated, we used `SLEEP_PERIOD`$= 100\mu s$ (for UNIX-based machines) or $= 1ms$ (minimum possible for Windows-based machines) and set `LOOP_PERIOD` such that the active and idle periods have (almost) equal lengths.

We used 6 laptops from 5 different vendors (see Table I), various processor architecture generations, and three popular OS families (Linux, MacOS, and Windows). To receive the EM signals, we placed the probe 10 cm away from the computer. To find the position where the signal power is the strongest, we manually localized the source of the signal. We found that the position which the signal is strongest is slightly different from one laptop to another but they are mostly concentrated in the middle and/or the bottom right quarter of the laptop (on top of the keyboard). Note that all these measurements were done without making any changes to the laptop's package. Also, all the measurements were done in the presence of other system's normal activities (i.e., handling interrupts, context-switch, etc.).

2) Near-Field Measurements: To measure the BER and bit-rate, we created a randomly-generated sequence of bits. Also, to decrease the BER, the *transmitter* application inserts parity bits such that the minimum Hamming distance between different codewords was at least three (to correct one error). In Table II, we provide the experimental results for six laptops. The columns of the table correspond to the average number (for 5 runs) of BER, transmission rate (TR), insertion probability (IP), and deletion probability (DP). To calculate the IP and DP, we compared the actual transmitted sequence with the received sequence based on the algorithm described in § IV.

Table I
LIST OF LAPTOPS AND THEIR OS AND (INTEL) PROCESSOR ARCHITECTURE USED IN OUR EXPERIMENTS.

Model	OS	Architecture
Dell Precision 7290	Windows 10	Kaby Lake
MacBookPro-2015	macOS (Mojave)	Broadwell
Dell Inspiron 15-3537	Linux (Debian)	Haswell
MacBookPro-2018	macOS (Mojave)	Coffee Lake
Lenovo Thinkpad	Linux (Ubuntu)	SkyLake
Sony Ultrabook	Windows 8	Ivy Bridge

Table II
EXPERIMENTAL RESULTS FOR CLOSE PROXIMITY. THE RESULTS INCLUDE THE BIT-ERROR-RATE (BER), TRANSMISSION-RATE (TR), INSERTION PROBABILITY (IP), AND DELETION PROBABILITY (DP) FOR THE PROPOSED COVERT CHANNEL ON DIFFERENT LAPTOPS.

	BER	TR (bps)	IP	DP
DELL Precision	2×10^{-3}	982	0	0
MacBookPro (2015)	3×10^{-2}	3700	0	3×10^{-3}
DELL Inspiron	8×10^{-3}	3162	4.5×10^{-3}	6.3×10^{-3}
MacBookPro (2018)	2.8×10^{-2}	3640	0	2.9×10^{-3}
Lenovo Thinkpad	5×10^{-3}	3020	0	1×10^{-3}
Sony Ultrabook	4×10^{-3}	974	0	5×10^{-3}

As shown in the table, the proposed covert channel can achieve up to 3.7 kbps (kilo-bits per second) while having less than 0.1% BER with a low-cost and compact setup, and that the main determinant of the bit-rate is the operating system, i.e., the precision with which the *transmitter* application can control idleness time - the `usleep` in Linux and MacOS is significantly more precise than `sleep` used in Windows, so the TR in Linux and MacOS systems is 3-4 kbps while the TR for Windows systems is slightly below 1 kbps.

Figure 9 compares the maximum TR of the proposed covert channel to the state-of-the-art. Specifically we compared our method to 7 different attacks that leveraged *physical* side-channels to establish a covert channel. As can be seen from the figure, the proposed covert channel can achieve more than 3x faster TR compared to the fastest existing covert channel attack, GSMem [12]. Note that, to provide a *fair comparison*, for each work we only report the TR with a similar measurement setup (if available), i.e., similar distance and/or measurement equipment.

To study the effect of **background activity** on the TR,

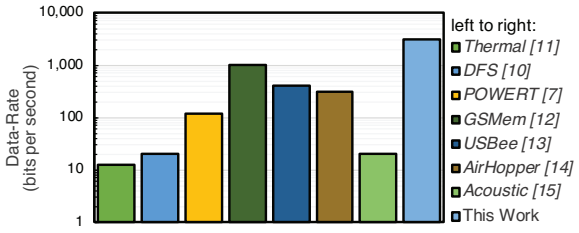

Figure 9. Transmission-rate comparison (higher is faster) between the proposed covert channel and the state-of-the-art (shown in log-scale). Each bar represents an existing attack. The proposed work achieve more than 3x higher TR compared to the fastest attack.

Table III
EXPERIMENTAL RESULTS WITH DISTANCE. THE RESULTS INCLUDE THE BIT-ERROR-RATE (BER), TRANSMISSION-RATE (TR), INSERTION PROBABILITY (IP), AND DELETION PROBABILITY (DP).

Distance	BER	TR (bps)	IP	DP
1 m	9×10^{-3}	1872	5×10^{-3}	0
	9×10^{-4}	1645	1×10^{-3}	0
1.5 m	5×10^{-3}	1454	0	0
2.5 m	8×10^{-3}	1110	0	0

we repeated our measurements, this time with adding a resource-intensive background activity (in addition to normal OS background activities which were present in all the results showed in Table II and Figure 9). We observed that to handle background activities, the OS tends to produce short bursts of activity which do not affect our covert-channel detection much since they are smaller than one sleep/active period. Longer bursts of activity do create some errors in our detection, but these are fixed/corrected using parity-bits (c.f. §IV-B4). However, if there are longer periods of activity, e.g., intense activities by other processes, the covert-channel transmission can either lower the transmission-rate or even pause temporarily. Using this new measurement, we found that to achieve similar BER to that of Table II in the presence of resource-intensive background activity, the TR has to be decreased (only for UNIX and macOS laptops), on average, by 15% (with worst-case of 21%).

3) Distance Measurements: **LoS Measurements**. To study the effect of distance on TR, we computed the BER and TR while the loop antenna was put 1, 1.5, and then 2.5 meters away from the target laptop. We manually set the antenna's orientation to maximize the signal SNR. As mentioned in §IV-C1, the loop antenna has about a 30 cm diameter and it can be easily hidden in a briefcase.

Table III shows the results for these 3 distances for the DELL Inspiron laptop. As can be seen from this table, the TR can be approximately 2 kbps when the distance for the communication link is around 1m. Additionally, we decrease TR so that BER of the system at different distances is almost the same for a fair evaluation of the performance of the communication link. It can be observed that having smaller TR for larger distances makes the communication more reliable. Although the rate decreases as the distance increases, we can receive 1110 bps when the distance is about 2.5m.

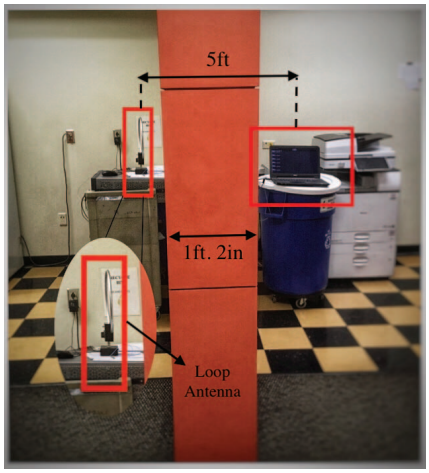

Figure 10. Experimental setup when an attacker and a victim are separated by a wall.

NLoS Measurements (through the wall). The setup for NLoS measurements is shown in Figure 10 where the transmitter and receiver are separated by an office wall which has about 35 cm thickness. Moreover, as can be seen, there are other electronic devices such as a printer in the transmitter's room and a refrigerator in the receiver's room which also generates unintentional EM emanations which can interfere with the laptop's emanations and hence makes the signal noisier. Note that we intentionally chose this setup to show that the proposed covert communication can work reliably even in the presence of other sources of EM emanations.

To maintain a low BER compared to the near-field measurements, the bit-rate had to be decreased to 821 bps while having 6×10^{-3} BER. However, as the signaling period is now significantly longer than the typical duration of an interrupt, the detection algorithm is better able to tolerate these system events, and so deletion and/or insertion of bits occurs less often or not at all. Overall, the NLoS measurements show that the covert communications are still possible (although at the lower rate) even when the transmitter and receiver are in two separate rooms which makes the attack more stealthy.

V. KEYLOGGING

A. Overview

The goal in keystroke logging, or *keylogging*, is to find when and which key has been pressed on a computer keyboard. This, in turn, can lead to stealing sensitive information, passwords, etc. In general, every keystroke can

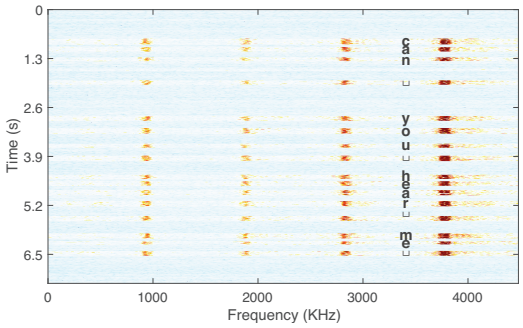

Figure 11. PMU's EM emanations over time when the user is typing: "`can you hear me `".

be shown as a 3-tuple [68], (t_p, t_r, k), where t_p is when the key is pressed, t_r is when the key is released, and k is the physical key identifier. Using this definition, keylogging becomes a two-phase problem: *keystroke detection* (i.e., correctly finding t_p and t_r) and *key identification* (i.e., finding k). It is important to mention that while, ideally, the goal for keylogging is to find the exact character for each keystroke, realistic attacks [68] typically provide *a significantly reduced possible states* for a keystroke/word. Using this reduced state space, an attacker can then leverage a *brute-force* attack to eventually find the actual characters. In this section, we show how EM emanations from PMU can be leveraged to provide highly accurate keystroke detection.

Compared to the existing works that leveraged digital (e.g., cache usage) [69]–[72] and/or physical [45], [52], [73]–[77] for keylogging, the main advantage of using the proposed side-channel for keylogging is that it enables the attackers to perform an attack from a distance behind a wall even on an isolated air-gapped computer.

B. Keylogging Side-Channel Attack

The main idea behind this attack is that pressing a key creates a *burst* of activity on the processor which, in turn, causes the (otherwise idle) processor to briefly switch to an *active* state, hence creates (stronger) EM emanations from the PMU.

Figure 11 shows the spectrogram for the EM emanations received from the PMU (using the setup used in §IV-C2) when the user is typing a sentence/password: `can you hear me`. As can be seen in the figure, each character (including the white-spaces shown as '␣') has a distinguishable pattern which means the total number of characters can be found with high accuracy using a simple-yet-effective detection algorithm. Moreover, as can be observed in Figure 11, the number of words and their length can also be inferred by grouping relatively close spikes (lines in the spectrogram) together.

Apart from the number of characters, the number of words, and the length of each word, existing work [78], [79] has shown that the duration of each keystroke and the time difference between two consecutive keys can also be leveraged to further reduce the search space for key identification. e.g., Salthouse [78] has empirically shown that *(i)* keys that are far apart are pressed in quicker succession than keys that are close together (e.g., see `you` vs. `can` in Figure 11). *(ii)* letter pairs that occur frequently in language are typed in quicker succession than infrequent letter pairs (e.g., see `hear` in Figure 11). *(iii)* practicing a specific keystroke sequence can significantly reduce inter-key timings (e.g., compare the timing for the white-space character in the first word, `can`, vs. the rest of the sentence). Using these findings, after properly detecting each keystroke's timing, supervised or unsupervised classifiers can be used to identify the keystrokes/words and/or to reduce the search space.

C. Experimental Evaluation

Setup. We used the same setup discussed in § IV-C for near-field and distance measurements and performed our measurements at three distances: 10cm, 2m, and 1.5m through the wall while the DELL Precision laptop was used. For each distance, a randomly-generated text with 1000 words[4] is typed (by the same person) in the Chrome browser while the signal was recorded. It is important to mention that all the measurements were done in the presence of other system's normal activities, including activities related to the browser.

Keystroke/Character Detection. To find the total number of characters (i.e., total number of keystrokes), we first normalized and then transformed the signal into a sequence of (non-overlapping) spectral samples (SS) by using short-time Fourier transform (STFT), which divides the signal into equal-sized segments (*windows*) each 5ms long. STFT then applied the Fast Fourier Transform (FFT) to each window to obtain its spectrum. We then chose a frequency band which contains the spikes related to PMU (the band is typically known for each device, but can also be easily found using standard peak detection techniques). We then used a simple thresholding technique (cf. §IV-B3) to decide whether the particular window contains a keystroke or not. To reduce false-positives, our detection algorithm then filtered out keystrokes that are shorter than a threshold (i.e., 30 ms in our experiments) knowing that a valid keystroke should take longer than this threshold.

Table IV shows the accuracy (in terms of true-positive and false-positive rates, denoted as TPR and FPR) of detecting characters for the three distances. False positives are mainly caused by other system activity, such as handling of the browser requests, which also appeared as a (typically much shorter) bursts of activity. As the distance increased, such activity became less prominent (i.e., lower signal amplitude) which, in turn, caused a reduction in FPR. However, this reduction in the amplitude also affected the

[4]obtained from https://www.livechatinc.com/typing-speed-test/#/

Table IV
EXPERIMENTAL RESULTS FOR KEYLOGGING IN DIFFERENT DISTANCES. RESULTS SHOW THE ACCURACY OF CORRECTLY DETECTING CHARACTERS AND WORD LENGTHS.

Distance	Char. Acc.		Word Acc.	
	TPR	FPR	Precision	Recall
10 cm	100%	3%	71%	100%
2m	99%	1.8%	70%	100%
1.5m (with wall)	97%	0.7%	70%	98%

TPR as the emanations caused by keystrokes also became weaker.

Word Detection. Once characters are correctly detected, the next step is to group the characters into the words. There are numerous techniques to reconstruct words based on individual characters. Particularly, we used the method proposed by Berger *et al.* [75]. Table IV shows the accuracy. Since word-length is a multi-class classification problem, instead of TPR and FPR, we report *precision* percentage as the fraction of correctly labeled words (i.e., words with correctly predicted length) among the retrieved words, and also report *recall* as the fraction of words that have been retrieved over the total amount of existing words in the text. As the table shows, we were able to detect almost all the words while keeping the precision accuracy above 70%. The distance did not have a significant impact on either of these two metrics.

VI. RELATED WORK AND COUNTERMEASURES

Electromagnetic Side-Channels. Prior work on EM side-channel mainly focused on the extraction of small amounts of highly sensitive data (such as cryptographic keys) from the system [30], [36], [36], [80]–[82]. Beyond extracting sensitive data values, EM emanations have also been used to learn more about program behavior, e.g., for identifying web pages during browsing [83], program execution profiling [84], [85], finding anomalies in software activity [86]–[90], establishing trust [91], etc.

While existing works have shown the existence of side-channel EM emanations from different electronic components within a computer, to the best of our knowledge, this is the first work that identifies the EM-based vulnerability created by the power management unit and exploits this vulnerability to establish a covert channel and/or a keylogging framework. To our knowledge, the work closest to ours are the methods that systematically find the amplitude and frequency-modulated EM emanations from modern systems [92], [93]. Those works discovered that the strongest FM and AM-modulated signals are created by voltage regulators, memory refreshes, and DRAM clocks in modern computers. While the existence of the side-channel signal from the voltage regulator has been presented in these works, the fundamental relationship between CPU's power states and the resulting EM leakages was not identified, and these signals were not exploited to establish a covert channel/keylogging. In fact, this prior work identifies that different *activity* on the processor weakly modulates the "carrier" signal emitted from the voltage regulator, not that the difference between active and idle states practically amounts to on-off keying, the strongest possible form of amplitude modulation, which is a much more powerful source of information leakage.

Also related to this paper are the methods based on the variations on the voltage and their corresponding EM emanations [32], [39], [50]. The main difference between the proposed side-channel and these works is that voltage-variation is *instruction-dependent* while the VRM side-channel is a *micro-architecture vulnerability*. Note that while the EM emanations for the voltage-variation are the strongest close to the VRM (because VRM spends most of the power) its existence is unrelated to how the VRM operates (i.e., the created side-channel is application-dependent, not VRM-dependent). Due to this difference, the proposed side-channel creates much stronger signals which can be received several meters away. Moreover, unlike other methods, it is resistant against techniques like randomization/blinding.

Physical Side-Channels. Exploiting other physical side-channels such as acoustic, temperature, etc. [11], [37]–[53] to create a covert channel/keylogging, is another groups of related work. Overall, depending on the setup, different physical side-channels can be used. The main differences between them are *a)* measurement requirements (i.e., cost and size of the receiver, signal-to-noise ratio and maximum achievable distance, etc.), and *b)* the maximum achievable bit-rate which indicates how severe and stealthy the attack could be. As shown in §IV-C, compared to the state-of-the-art, the proposed side-channel can achieve more than 3x faster bit-rate mainly because it relies on the fast switches between *idle* and *active* states on PMU.

Yet another related work to this paper are the methods to exploit the power management unit by leveraging *digital* side-channels [7], [9], [10], [94]. Most recently, Khatamifard *et al.* [7] proposed POWERT, a new method to leverage the power budget for creating a covert channel. To transmit data, POWERT either creates power-intensive activities or remains idle. To infer the transmitted data, the Sink process (also located on the same physical device), measures its own performance by running a known application. The key idea is that due to the limitation in the power budget, the Sink's performance is directly impacted (modulated) by the source activity. In this paper, we showed that compared to POWERT [7], our proposed covert channel can achieve significantly higher data-rate (>20x) since it does not rely on the indirect measurement of the performance while it

is less susceptible to the noises generated by the system. Moreover, as mentioned in § II, the threat model for these attacks are different from the one used in this paper since they are leveraging digital side-channels.

Circuit-Level Methods. Another related body of work to this paper are circuit-level techniques to improve the EM/Power side-channel resistance, especially for integrated cryptographic accelerators on the chip such as AES engines [95]–[98]. These techniques are mainly focused on using on-chip integrated voltage regulators [99] (hence reducing the amount of pin accessibility to the attacker) and adding randomness to the voltage regulator which, in turn, creates random patterns in the side-channel signals [100].

Countermeasures. There are multiple possible countermeasures to mitigate the discovered vulnerability. At the system-level, the system software and/or user can disable P/C-states to perform highly-sensitive computation (briefly, because the energy overheads are significant and/or because of the temperature-control requirements). Another solution in this vein would be to use methods like architecture-blinking [101] (processor is briefly disconnected from the PMU during sensitive computation to avoid PMU-related leakage of information). A more fundamental solution, however, would involve adding pre-determinism, randomness, and/or noise to the operation of the PMU which, mainly, should be performed at the circuit-level. Finally, traditional EMI-shielding methods can be used to reduce the SNR and increase the BER (with their own limitations/overheads).

VII. Conclusions

In this paper, we presented a new, previously unexplored, physical side-channel vulnerability caused by the design, implementation, and typical use of the power management unit (and its main component, the voltage regulator module) in modern computers. Specifically, we showed that different power states on the system (which are primarily utilized to optimize the energy efficiency) and the way existing power management units create these states can lead to a side-channel which can leak sensitive information about the current state of the system.

To exploit this side-channel, we demonstrated two real exploits: *a)* a covert channel which leveraged an inexpensive and compact setup to establish the covert communication and *b)* a keylogging framework which can accurately detect the keystrokes even when the receiver was behind a wall on a separate room.

In our experimental evaluation, we used a number of laptop systems, from various vendors, as the target system. We also performed our measurements under realistic conditions that include environmental EM noise and having other applications run on the target system concurrently with our data-exfiltration applications.

Our results showed that this newly-discovered side-channel exists on all systems we evaluated, regardless of hardware platform, operating system, and vendor. Further, we showed that, compared to the state-of-the-art, exploiting this side-channel can lead to a much faster covert channel (when a physical side-channel is used).

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable feedbacks. This work has been supported, in part, by NSF grant 1563991 and DARPA LADS contract FA8650-16-C-7620. The views and findings in this paper are those of the authors and do not necessarily reflect the views of NSF and DARPA.

References

[1] P. Kocher, J. Horn, A. Fogh, , D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom, "Spectre attacks: Exploiting speculative execution," in *40th IEEE Symposium on Security and Privacy (S&P)*, 2019.

[2] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, A. Fogh, J. Horn, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, and M. Hamburg, "Meltdown: Reading kernel memory from user space," in *27th USENIX Security Symposium*, 2018.

[3] J. Van Bulck, M. Minkin, O. Weisse, D. Genkin, B. Kasikci, F. Piessens, M. Silberstein, T. F. Wenisch, Y. Yarom, and R. Strackx, "Foreshadow: Extracting the keys to the Intel SGX kingdom with transient out-of-order execution," in *Proceedings of the 27th USENIX Security Symposium*, 2018.

[4] E. M. Koruyeh, K. N. Khasawneh, C. Song, and N. Abu-Ghazaleh, "Spectre Returns! Speculation Attacks Using the Return Stack Buffer," in *12th USENIX Workshop on Offensive Technologies (WOOT)*, 2018.

[5] D. Evtyushkin, R. Riley, N. Abu-Ghazaleh, and D. Ponomarev, "Branchscope: A new side-channel attack on directional branch predictor," in *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pp. 693–707, 2018.

[6] B. Gras, K. Razavi, H. Bos, and C. Giuffrida, "Translation leak-aside buffer: Defeating cache side-channel protections with TLB attacks," in *27th USENIX Security Symposium*, pp. 955–972, 2018.

[7] S. K. Khatamifard, L. Wang, A. Das, S. Kose, and U. R. Karpuzcu, "POWERT channels: A novel class of covert communication exploiting power management vulnerabilities," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 291–303, 2019.

[8] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for reduced CPU energy," in *Proceedings of the 1st USENIX Conference on Operating Systems Design and Implementation (OSDI)*, 1994.

[9] A. Tang, S. Sethumadhavan, and S. Stolfo, "CLKSCREW: Exposing the perils of security-oblivious energy management," in *Proceedings of the 26th USENIX Conference on Security Symposium*, pp. 1057–1074, 2017.

[10] M. Alagappan, J. Rajendran, M. Doroslovački, and G. Venkataramani, "DFS covert channels on multi-core platforms," in *2017 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, pp. 1–6, 2017.

[11] R. J. Masti, D. Rai, A. Ranganathan, C. Müller, L. Thiele, and S. Capkun, "Thermal covert channels on multi-core platforms," in *24th USENIX Security Symposium*, pp. 865–880, 2015.

[12] M. Guri, A. Kachlon, O. Hasson, G. Kedma, Y. Mirsky, and Y. Elovici, "GSMem: Data exfiltration from air-gapped computers over GSM frequencies," in *24th USENIX Security Symposium*, pp. 849–864, 2015.

[13] M. Guri, M. Monitz, and Y. Elovici, "USBee: Air-gap covert-channel via electromagnetic emission from USB," *CoRR*, vol. abs/1608.08397, 2016.

[14] M. Guri, G. Kedma, A. Kachlon, and Y. Elovici, "Airhopper: Bridging the air-gap between isolated networks and mobile phones using radio frequencies," in *9th International Conference on Malicious and Unwanted Software: The Americas (MALWARE)*, pp. 58–67, 2014.

[15] M. Hanspach and M. Goetz, "On covert acoustical mesh networks in air," *Journal of Communications*, vol. 8, no. 11, 2013.

[16] Y. Yarom and K. Falkner, "FLUSH+RELOAD: A high resolution, low noise, L3 cache side-channel attack," in *23rd USENIX Security Symposium*, pp. 719–732, 2014.

[17] F. Liu, Y. Yarom, Q. Ge, G. Heiser, and R. B. Lee, "Last-level cache side-channel attacks are practical," in *Proceedings of the 2015 IEEE Symposium on Security and Privacy (S&P)*, pp. 605–622, 2015.

[18] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds," in *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS)*, pp. 199–212, 2009.

[19] F. Yao, M. Doroslovacki, and G. Venkataramani, "Are coherence protocol states vulnerable to information leakage?," in *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 168–179, 2018.

[20] G. Irazoqui, T. Eisenbarth, and B. Sunar, "SA: A shared cache attack that works across cores and defies VM sandboxing and its application to AES," in *2015 IEEE Symposium on Security and Privacy (S&P)*, pp. 591–604, 2015.

[21] Y. Oren, V. P. Kemerlis, S. Sethumadhavan, and A. D. Keromytis, "The spy in the sandbox: Practical cache attacks in javascript and their implications," in *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 1406–1418, 2015.

[22] M. Lipp, D. Gruss, R. Spreitzer, C. Maurice, and S. Mangard, "ARMageddon: Cache attacks on mobile devices," in *25th USENIX Security Symposium*, pp. 549–564, 2016.

[23] D. Evtyushkin and D. Ponomarev, "Covert channels through random number generator: Mechanisms, capacity estimation and mitigations," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 843–857, 2016.

[24] S. K. Khatamifard, L. Wang, S. Köse, and U. R. Karpuzcu, "A new class of covert channels exploiting power management vulnerabilities," *IEEE Computer Architecture Letters*, vol. 17, no. 2, pp. 201–204, 2018.

[25] P. Pessl, D. Gruss, C. Maurice, M. Schwarz, and S. Mangard, "DRAMA: Exploiting DRAM addressing for cross-cpu attacks," in *25th USENIX Security Symposium*, pp. 565–581, 2016.

[26] Z. Wu, Z. Xu, and H. Wang, "Whispers in the hyper-space: High-speed covert channel attacks in the cloud," in *21st USENIX Security Symposium*, pp. 159–173, 2012.

[27] O. Aciiçmez, C. K. Koç, and J.-P. Seifert, "On the power of simple branch prediction analysis," in *Proceedings of the 2Nd ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, pp. 312–320, 2007.

[28] D. Evtyushkin, D. Ponomarev, and N. Abu-Ghazaleh, "Understanding and mitigating covert channels through branch predictors," *ACM Trans. Archit. Code Optim.*, vol. 13, no. 1, pp. 10:1–10:23, 2016.

[29] H. Naghibijouybari, A. Neupane, Z. Qian, and N. Abu-Ghazaleh, "Rendered insecure: GPU side channel attacks are practical," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 2139–2153, 2018.

[30] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, "The EM side-channel(s)," in *Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, pp. 29–45, 2003.

[31] R. Callan, A. Zajić, and M. Prvulovic, "A practical methodology for measuring the side-channel signal available to the attacker for instruction-level events," in *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 242–254, 2014.

[32] D. Genkin, L. Pachmanov, I. Pipman, and E. Tromer, "Stealing keys from PCs using a radio: Cheap electromagnetic attacks on windowed exponentiation," in *Cryptographic Hardware and Embedded Systems (CHES)*, pp. 207–228, 2015.

[33] A. Zajić and M. Prvulovic, "Experimental demonstration of electromagnetic information leakage from modern processor-memory systems," *IEEE Transactions on Electromagnetic Compatibility*, vol. 56, no. 4, pp. 885–893, 2014.

[34] Y. Han, S. Etigowni, H. Liu, S. Zonouz, and A. Petropulu, "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 (CCS)*, pp. 1095–1108, 2017.

[35] Z. Hadjilambrou, S. Das, M. A. Antoniades, and Y. Sazeides, "Sensing CPU voltage noise through electromagnetic emanations," *IEEE Comput. Archit. Lett.*, vol. 17, no. 1, pp. 68–71, 2018.

[36] M. Alam, H. A. Khan, M. Dey, N. Sinha, R. Callan, A. Zajic, and M. Prvulovic, "One&done: A single-decryption EM-based attack on OpenSSL's constant-time blinded RSA," in *27th USENIX Security Symposium*, pp. 585–602, 2018.

[37] A. G. Bayrak, F. Regazzoni, P. Brisk, F. Standaert, and P. Ienne, "A first step towards automatic application of power analysis countermeasures," in *2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 230–235, 2011.

[38] D. Brumley and D. Boneh, "Remote timing attacks are practical," in *Proceedings of the 12th Conference on USENIX Security Symposium - Volume 12*, pp. 1–1, 2003.

[39] D. Genkin, L. Pachmanov, I. Pipman, E. Tromer, and Y. Yarom, "ECDSA key extraction from mobile devices via nonintrusive physical side channels," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 1626–1638, 2016.

[40] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology (CRYPTO)*, pp. 388–397, 1999.

[41] Y. Liu, L. Wei, Z. Zhou, K. Zhang, W. Xu, and Q. Xu, "On code execution tracking via power side-channel," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 1019–1031, 2016.

[42] B. Farshteindiker, N. Hasidim, A. Grosz, and Y. Oren, "How to phone home with someone else's phone: Information exfiltration using intentional sound noise on gyroscopic sensors," in *10th USENIX Workshop on Offensive Technologies (WOOT)*, 2016.

[43] M. Backes, M. Dürmuth, S. Gerling, M. Pinkal, and C. Sporleder, "Acoustic side-channel attacks on printers," in *Proceedings of the 19th USENIX Conference on Security*, pp. 20–20, 2010.

[44] D. Genkin, A. Shamir, and E. Tromer, "RSA key extraction via low-bandwidth acoustic cryptanalysis," in *Advances in Cryptology (CRYPTO)*, pp. 444–461, 2014.

[45] T. Zhu, Q. Ma, S. Zhang, and Y. Liu, "Context-free attacks using keyboard acoustic emanations," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 453–464, 2014.

[46] C. Song, F. Lin, Z. Ba, K. Ren, C. Zhou, and W. Xu, "My smartphone knows what you print: Exploring smartphone-based side-channel attacks against 3D printers," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 895–907, 2016.

[47] T. Halevi and N. Saxena, "Acoustic eavesdropping attacks on constrained wireless device pairing," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 3, pp. 563–577, 2013.

[48] J. Brouchier, T. Kean, C. Marsh, and D. Naccache, "Temperature attacks," *IEEE Security and Privacy*, vol. 7, pp. 79–82, Mar. 2009.

[49] Z. Long, X. Wang, Y. Jiang, G. Cui, L. Zhang, and T. Mak, "Improving the efficiency of thermal covert channels in multi-/many-core systems," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1459–1464, 2018.

[50] D. Genkin, I. Pipman, and E. Tromer, "Get your hands off my laptop: Physical side-channel key-extraction attacks on pcs," in *Proceedings of the 16th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, pp. 242–260, 2014.

[51] Y. Su, D. Genkin, D. Ranasinghe, and Y. Yarom, "USB snooping made easy: Crosstalk leakage attacks on USB hubs," in *Proceedings of the 26th USENIX Conference on Security Symposium*, pp. 1145–1161, 2017.

[52] P. Marquardt, A. Verma, H. Carter, and P. Traynor, "(Sp)iPhone: Decoding vibrations from nearby keyboards using mobile phone accelerometers," in *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS)*, pp. 551–562, 2011.

[53] R. Ogen, K. Zvi, O. Shwartz, and Y. Oren, "Sensorless, permissionless information exfiltration with Wi-Fi micro-jamming," in *12th USENIX Workshop on Offensive Technologies (WOOT)*, 2018.

[54] V. Kiriansky, I. Lebedev, S. Amarasinghe, S. Devadas, and J. Emer, "DAWG: A defense against cache timing attacks in speculative execution processors," in *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 974–987, 2018.

[55] Intel-Corp., "Enhanced Intel® SpeedStep® Technology for the Intel® Pentium® M Processor ," tech. rep., 2004.

[56] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits (2nd Edition)*. Prentice Hall, 2013.

[57] Intel-Corp., "Power Management in Intel® Architecture Servers ," tech. rep., 2009.

[58] Intel-Corp., "Intel® Xeon® Processor E3-1200 v3 Family," tech. rep., 2014.

[59] H. Mujtaba, "Intel's 6th Gen Skylake Unwrapped – CPU Microarchitecture, Gen9 Graphics Core and Speed Shift Hardware P-State." https://wccftech.com/idf15-intel-skylake-analysis-cpu-gpu-microarchitecture-ddr4-memory-impact/4/, 2015 (accessed Feb., 2019).

[60] Intel-Corp., "Voltage Regulator-Down 11.1: Processor Power Delivery Design Guide," tech. rep., 2009.

[61] Intel-Corp., "ER6230QI: 3A DC-DC Step-Down Switching Regulator," tech. rep., 2018.

[62] J. Sun, M. Xu, Y. Ren, and F. C. Lee, "Light-load efficiency improvement for buck voltage regulators," *IEEE Transactions on Power Electronics*, vol. 24, pp. 742–751, March 2009.

[63] J. Su and C. Liu, "A novel phase-shedding control scheme for improved light load efficiency of multiphase interleaved DC–DC converters," *IEEE Transactions on Power Electronics*, vol. 28, pp. 4742–4752, Oct 2013.

[64] Y. Ahn, I. Jeon, and J. Roh, "A multiphase buck converter with a rotating phase-shedding scheme for efficient light-load control," *IEEE Journal of Solid-State Circuits*, vol. 49, pp. 2673–2683, Nov 2014.

[65] E. Cole, *Advanced persistent threat: understanding the danger and how to protect your organization*. Newnes, 2012.

[66] RTL-SDR.COM, "RTL-SDR Blog V3 Users." https://www.rtl-sdr.com/rtl-sdr-quick-start-guide/, 2019 (accessed Feb., 2019).

[67] Universal-Radio-Inc., "AOR LA390 Wideband Loop Antenna." https://www.universal-radio.com/catalog/sw_ant/2320.html, 2014 (accessed Feb., 2019).

[68] J. V. Monaco, "SoK: Keylogging side channels," in *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 211–228, 2018.

[69] M. Schwarz, M. Lipp, D. Gruss, S. Weiser, C. Maurice, R. Spreitzer, and S. Mangard, "Keydrown: Eliminating software-based keystroke timing side-channel attacks," in *Proceedings of Network and Distributed System Security Symposium (NDSS)*, 2018.

[70] D. Wang, A. Neupane, Z. Qian, N. B. Abu-Ghazaleh, S. V. Krishnamurthy, E. J. Colbert, and P. Yu, "Unveiling your keystrokes: A cache-based side-channel attack on graphics libraries.," in *Proceedings of Network and Distributed System Security Symposium (NDSS)*, 2019.

[71] D. Gruss, R. Spreitzer, and S. Mangard, "Cache template attacks: Automating attacks on inclusive last-level caches," in *Proceedings of the 24th USENIX Conference on Security Symposium*, pp. 897–912, 2015.

[72] P. Vila and B. Kopf, "Loophole: Timing attacks on shared event loops in chrome," in *26th USENIX Security Symposium*, pp. 849–864, Aug. 2017.

[73] K. Ali, A. X. Liu, W. Wang, and M. Shahzad, "Keystroke recognition using WiFi signals," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking (MobiCom)*, pp. 90–102, 2015.

[74] M. Vuagnoux and S. Pasini, "Compromising electromagnetic emanations of wired and wireless keyboards," in *Proceedings of the 18th Conference on USENIX Security Symposium*, pp. 1–16, 2009.

[75] Y. Berger, A. Wool, and A. Yeredor, "Dictionary attacks using keyboard acoustic emanations," in *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS)*, pp. 245–254, 2006.

[76] X. Liu, Z. Zhou, W. Diao, Z. Li, and K. Zhang, "When good becomes evil: Keystroke inference with smartwatch," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 1273–1285, 2015.

[77] D. Asonov and R. Agrawal, "Keyboard acoustic emanations," in *IEEE Symposium on Security and Privacy (SP).*, pp. 3–11, May 2004.

[78] T. A. Salthouse, "Perceptual, cognitive, and motoric aspects of transcription typing.," *Psychological bulletin*, vol. 99, no. 3, p. 303, 1986.

[79] A. M. Feit, D. Weir, and A. Oulasvirta, "How we type: Movement strategies and performance in everyday typing," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pp. 4262–4273, 2016.

[80] C. H. Gebotys, S. Ho, and C. C. Tiu, "EM analysis of Rijndael and ECC on a wireless java-based PDA," in *Proceedings of the 7th International Conference on Cryptographic Hardware and Embedded Systems (CHES)*, pp. 250–264, 2005.

[81] G. Camurati, S. Poeplau, M. Muench, T. Hayes, and A. Francillon, "Screaming channels: When electromagnetic side channels meet radio transceivers," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 163–177, 2018.

[82] H. J. Highland, "Random bits & bytes: Electromagnetic radiation revisited," *Comput. Secur.*, vol. 5, pp. 85–93, June 1986.

[83] S. S. Clark, H. Mustafa, B. Ransford, J. Sorber, K. Fu, and W. Xu, "Current events: Identifying webpages by tapping the electrical outlet," in *Computer Security (ESORICS)*, pp. 700–717, 2013.

[84] N. Sehatbakhsh, A. Nazari, A. Zajic, and M. Prvulovic, "Spectral profiling: Observer-effect-free profiling by monitoring EM emanations," in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 1–11, 2016.

[85] R. Callan, F. Behrang, A. Zajic, M. Prvulovic, and A. Orso, "Zero-overhead profiling via EM emanations," in *Proceedings of the 25th International Symposium on Software Testing and Analysis (ISSTA)*, p. 401–412, 2016.

[86] S. S. Clark, B. Ransford, A. Rahmati, S. Guineau, J. Sorber, W. Xu, and K. Fu, "Wattsupdoc: Power side channels to nonintrusively discover untargeted malware on embedded medical devices," in *Presented as part of the 2013 USENIX Workshop on Health Information Technologies*, 2013.

[87] A. Nazari, N. Sehatbakhsh, M. Alam, A. Zajic, and M. Prvulovic, "EDDIE: EM-based detection of deviations in program execution," in *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pp. 333–346, 2017.

[88] N. Sehatbakhsh, M. Alam, A. Nazari, A. Zajic, and M. Prvulovic, "Syndrome: Spectral analysis for anomaly detection on medical iot and embedded devices," in *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 1–8, 2018.

[89] N. Sehatbakhsh, A. Nazari, M. Alam, F. Werner, Y. Zhu, A. Zajic, and M. Prvulovic, "REMOTE: Robust external malware detection framework by using electromagnetic signals," *IEEE Transactions on Computers*, pp. 1–1, 2019.

[90] H. A. Khan, N. Sehatbakhsh, L. N. Nguyen, R. L. Callan, A. Yeredor, M. Prvulovic, and A. Zajic, "IDEA: Intrusion detection through electromagnetic-signal analysis for critical embedded and cyber-physical systems," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2019.

[91] N. Sehatbakhsh, A. Nazari, H. Khan, A. Zajic, and M. Prvulovic, "EMMA: Hardware/software attestation framework for embedded systems using electromagnetic signals," in *Proceedings of the 52Nd Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 983–995, 2019.

[92] R. Callan, A. Zajić, and M. Prvulovic, "FASE: Finding amplitude-modulated side-channel emanations," in *Proceedings of the 42nd Annual International Symposium on Computer Architecture (ISCA)*, pp. 592–603, 2015.

[93] M. Prvulovic, A. Zajić, R. L. Callan, and C. J. Wang, "A method for finding frequency-modulated and amplitude-modulated electromagnetic emanations in computer systems," *IEEE Transactions on Electromagnetic Compatibility*, vol. 59, pp. 34–42, Feb 2017.

[94] R. JayashankaraShridevi, C. Rajamanikkam, K. Chakraborty, and S. Roy, "Catching the flu: Emerging threats from a third party power management unit," in *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2016.

[95] C. Tokunaga and D. Blaauw, "Secure aes engine with a local switched-capacitor current equalizer," in *2009 IEEE International Solid-State Circuits Conference - Digest of Technical Papers*, pp. 64–65,65a, Feb 2009.

[96] D. Das, S. Maity, S. B. Nasir, S. Ghosh, A. Raychowdhury, and S. Sen, "ASNI: Attenuated signature noise injection for low-overhead power side-channel attack immunity," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, pp. 3300–3311, Oct 2018.

[97] O. A. Uzun and S. Köse, "Converter-gating: A power efficient and secure on-chip power delivery system," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 4, pp. 169–179, June 2014.

[98] M. Kar, A. Singh, S. Mathew, S. Ghosh, A. Rajan, V. De, R. A. Beyah, and S. Mukhopadhyay, "Blindsight: Blinding EM side-channel leakage using built-in fully integrated inductive voltage regulator," *CoRR*, vol. abs/1802.09096, 2018.

[99] M. Kar, A. Singh, S. K. Mathew, A. Rajan, V. De, and S. Mukhopadhyay, "Reducing power side-channel information leakage of aes engines using fully integrated inductive voltage regulator," *IEEE Journal of Solid-State Circuits*, vol. 53, pp. 2399–2414, Aug 2018.

[100] A. Singh, M. Kar, S. Mathew, A. Rajan, V. De, and S. Mukhopadhyay, "Improved power side channel attack resistance of a 128-bit AES engine with random fast voltage dithering," in *ESSCIRC 2017 - 43rd IEEE European Solid State Circuits Conference*, pp. 51–54, Sep. 2017.

[101] A. Althoff, J. McMahan, L. Vega, S. Davidson, T. Sherwood, M. B. Taylor, and R. Kastner, "Hiding intermittent information leakage with architectural support for blinking," in *Proceedings of the 45th Annual International Symposium on Computer Architecture (ISCA)*, pp. 638–649, 2018.