Graphics Peeping Unit: Exploiting EM Side-Channel Information of GPUs to Eavesdrop on Your Neighbors

Zihao Zhan[¶*§], Zhenkai Zhang[¶†], Sisheng Liang[†], Fan Yao[‡], Xenofon Koutsoukos[§]
[*]University of Florida, [†]Clemson University, [‡]University of Central Florida, [§]Vanderbilt University
[¶](Co-First Authors with Equal Contribution)

2022 IEEE Symposium on Security and Privacy (SP) | 978-1-6654-1316-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/SP46214.2022.9833773

***Abstract*—As the popularity of graphics processing units (GPUs) grows rapidly in recent years, it becomes very critical to study and understand the security implications imposed by them. In this paper, we show that modern GPUs can "broadcast" sensitive information over the air to make a number of attacks practical. Specifically, we present a new electromagnetic (EM) side-channel vulnerability that we have discovered in many GPUs of both NVIDIA and AMD. We show that this vulnerability can be exploited to mount realistic attacks through two case studies, which are website fingerprinting and keystroke timing inference attacks. Our investigation recognizes the commonly used dynamic voltage and frequency scaling (DVFS) feature in GPU as the root cause of this vulnerability. Nevertheless, we also show that simply disabling DVFS may not be an effective countermeasure since it will introduce another highly exploitable EM side-channel vulnerability. To the best of our knowledge, this is the first work that studies realistic physical side-channel attacks on non-shared GPUs at a distance.**

I. Introduction

Over the past few years, graphics processing units (GPUs) have become an integral part of modern computer systems, which are used not only for graphics rendering but also for intensive parallel computing. Given the fact that many tasks running on a GPU operate on sensitive information, concerns about the security of GPUs, especially potential information leakage, have been raised. Several recently developed attacks on GPU have justified such concerns [11], [20], [21], [27], [30], [33], [41], [47], [65].

Although these existing GPU attacks focus on different application scenarios, they all require that GPUs be either logically shared with or physically accessible[1] to adversaries. If an attacker has no physical or logical access to the GPUs used by her targets, is it still possible that the attacker can steal sensitive information from such GPUs? In this paper, we answer this question affirmatively by presenting a new physical side-channel vulnerability of modern GPUs as well as some examples of its exploitation.

Specifically, we have discovered certain electromagnetic (EM) emanations[2] from GPUs which are: ❶ exploitable – we find that these EM signals are computation-dependent and can reveal fine-grained information about the ongoing activity; and ❷ easy to measure – we find that such EM signals are strong and can propagate very far (e.g., more than 6 meters in many cases) as well as even penetrate thick walls. A further investigation reveals that the root cause of such exploitable far-field EM emanations is the dynamic voltage and frequency scaling (DVFS) feature of GPU, which has been playing an important role in either saving energy or improving GPU performance [1], [34], [42].

By exploiting the discovered EM side-channel information, for the first time, we demonstrate that it is not only possible but also practical to mount realistic eavesdropping attacks. Given a victim who uses a modern GPU without sharing, we show that an attacker can spy on the victim and identify the webpages visited by the victim (i.e., website fingerprinting attack) with a high accuracy. In addition, we show that the keystroke timings of the victim can be further inferred, which may be used to recover typed words or passphrases [54], [62]. As the attacker can be several meters away from the victim and the attacker can even hide in a separate cubicle or room, the presented attacks are in effect extremely stealthy.

To the best of our knowledge, our work serves as the *first* physical side-channel attack on non-shared GPUs at a distance. Prior to our work, there are only a few studies on leveraging physical side effects of GPU computation (e.g., power [30] or near-field EM [11]) to breach confidentiality, but all of them require physical access to the GPUs. On the contrary, we discover and exploit a far-field EM side-channel vulnerability, which empowers more practical attacks without too unrealistic proximity requirements.

Even outside of the GPU security area, we find that most of the existing *long-range* EM-based attacks are to build covert communication channels [16], [51], [52], [61]. Surprisingly (or not), there are not many works showing that it is possible to mount EM side-channel attacks on modern computers to steal sensitive information from several meters away. This is because EM emanations that are both far-field and exploitable for eavesdropping attacks appear to be hard to discover. Thus, our work also serves as a good demonstration of long-range EM side-channel attacks.

The main contributions of this paper are as follows:

- We present a new EM side-channel vulnerability that we have discovered in modern GPUs and can be exploited to carry out attacks at a distance and/or through a wall. We identify the ubiquitously used DVFS as the root cause of this side-channel and find that such a vulnerability exists in many GPUs of both NVIDIA and AMD.

[1]In [11], although contactless, the attack needs to remove GPU's heat sink and place a probe near GPU chip surface. In [30], it is the power supply of the computer rather than the GPU that is instrumented by the attacker.

[2]In this paper, we use EM emanations and EM signals interchangeably.

DOI 10.1109/SP46214.2022.00073

- We formulate a signal processing framework to address the challenges introduced by potential EM shielding and strong noise contamination. With the proposed techniques, we can exploit the EM emanations of interest even when they are greatly attenuated or they are overwhelmed by strong legitimate communication signals.
- We conduct two case studies on the exploitation of this newly found EM side-channel vulnerability. The first one is a website fingerprinting attack, and up to 93.2% accuracy can be achieved in a scenario where the attacker and victim are 6 meters apart. The second case study is a keystroke timing inference attack, where we show that keystroke events can be reliably detected to deduce inter-keystroke times.
- We show that even though disabling GPU DVFS can be an effective approach to mitigating the discovered EM side-channel vulnerability, it will unfortunately introduce another new one into many GPUs which can be exploited to mount comparable EM side-channel attacks. We also discuss some potential countermeasures.

As DVFS has been used or may appear in many other hardware components [5], [8], [23], [37], [38], our research also gives a pointer to what may need additional attention during certain security investigations. (Note that Sehatbakhsh *et al.* have also pioneered exploiting EM emanations affected by DVFS to carry out attacks [51], and we discuss the differences between their work and ours in Section X.)

Responsible Disclosure. We have reported our findings to both NVIDIA and AMD. NVIDIA replied to us that "NVIDIA is continually investigating ways to minimize board emissions in future product designs and will take these findings and recommendations under advisement". AMD informed us that their engineering team reviewed the issue and "no effective mitigations or fixes have been identified".

II. Background

In this section, we provide a brief overview of GPU architecture and GPU DVFS feature. Note that, in this paper, we do not consider any integrated GPUs in CPU processors, namely the term GPU is used to indicate the discrete ones designed by NVIDIA or AMD only. Moreover, we briefly present the physical side effects exploited in this paper, namely the EM emanations.

A. GPU Architecture

GPUs have evolved from hardwired graphics accelerators into highly parallel programmable computing devices. Usually, a modern GPU contains a number of single-instruction multiple-thread (SIMT) processors. Each SIMT processor has many simple GPU cores, and each core can perform scalar integer and floating-point arithmetic operations. Such SIMT processors are called *streaming multiprocessors* by NVIDIA and *compute units* by AMD. SIMT processors manage, schedule, and execute groups of parallel threads, which are named *warps* and *wavefronts* in the terminology of NVIDIA and AMD respectively.

To store large amounts of data, a modern GPU normally has several gigabytes of memory. Such large GPU memory is shared by all the SIMT processors on the GPU, and consists of multiple memory modules. These memory modules are of special DRAM type tailored for use in GPUs (e.g., GDDR5 and GDDR6). In general, a GPU needs high memory bandwidth to sustain its high computational throughput, and there are multiple memory controllers used to enable massively parallel access to the memory modules to reach the desired bandwidth. The GPU memory is independent of the main memory on the host side and also managed in its own manner. Data transfers between the main memory and GPU memory are via the PCIe bus.

B. Dynamic Voltage and Frequency Scaling

DVFS is a power management technique that has been widely used with respect to CPUs [60]. It dynamically changes voltage and frequency to adjust performance for power savings.[3] Instead of always staying at the highest level, performance is actively regulated according to current workloads, which can make very efficient use of energy.

As GPUs continue to grow powerful, their increasing power consumption has become a radical problem. To address this problem, the DVFS technique has been applied to GPUs. In fact, almost every modern GPU provides hardware support for DVFS [42]. For a GPU, DVFS governs the supply voltage and frequency of both its cores and memory.

Normally, there are multiple performance levels specified by GPU DVFS. (The performance levels are often called performance/power states, namely, P-states.) Each performance level defines a setting of voltage and frequency for the GPU cores and memory. (At a performance level, the frequency and/or voltage of the GPU cores may not be fixed but can vary within a specific range, while the frequency of the GPU memory usually does not change.) GPU DVFS dynamically switches performance levels to meet the current computation needs and minimize power draw, heat generation, and fan noise. In general, the approach to determining performance levels in official GPU drivers is proprietary and not well-documented. The default GPU DVFS approach is employed automatically, although an end user may choose to disable its functionality by manually setting fixed frequencies or to provide a customized approach [31].

C. Electromagnetic Emanations

Given the fact that electric current in the circuitry of a device varies with time, EM emanations inevitably arise. The EM emanations generated by a computer system are distributed widely in the spectrum. As these EM signals generally carry information about the underlying electronic activities, which can be linked with certain high-level activities, some of them have been leveraged in the context of security for attacks [2], [12]–[14], [25], [51] as well as defenses [17], [43], [64].

[3]DVFS principally reduces dynamic power consumption P_D, which is proportional to the voltage V quadratically and frequency f linearly, namely, $P_D \propto V^2 \times f$.

Although the sources of many of the EM emanations are unknown, a few of them are in effect easy to determine, e.g., emanations generated by some components whose activities are periodic, such as voltage regulators and DRAM clocks [3]. The EM signals created by these components having periodic switching behavior are also strong and may propagate to a distance of several meters. Interestingly, some of these signals may be unintentionally modulated by other activities in the form of amplitude-modulation (AM) or frequency-modulation (FM) [3], [48]. For example, the EM signals created by voltage regulators may be AM-modulated by activities in the circuits they power. Therefore, these signals act as carrier signals that convey information about the modulating activities. Moreover, to measure these far-field EM signals, very simple equipment suffices. For instance, a whip antenna and a cheap software-defined radio (SDR) device are adequate [51], [64].

III. Threat Model

We assume that there is an attacker who intends to eavesdrop on a victim to extract some of his/her sensitive information, e.g., the webpages the victim is browsing. The attacker is in the proximity of the victim, but they may still be well spaced apart. For example, the attacker and the victim may be colleagues or neighbors. Furthermore, they may be physically isolated from each other. For instance, the victim may be in a separate cubicle, office, or apartment, to which the attacker has no access.

The victim uses a desktop computer system which is equipped with a discrete GPU. (We do not consider laptops or other mobile computing devices in this threat model.) The GPU may be a product of either NVIDIA or AMD. We assume that the victim uses the official driver and its default settings, which is the most prevalent case in reality. The attacker is assumed to be able to find and employ the same type of GPU as the one used by the victim for profiling. (The attacker may have known what GPU the victim is using, but if this a priori knowledge is not available, we will show that the attacker may still be able to deduce it.) Unlike the prior work on GPU side-channel or memory dump attacks [27], [33], [41], [65], we do not require that the use of the GPU or any other computational resources be shared between the victim and the attacker. We neither require the presence of any software vulnerabilities.

IV. New Exploitable EM Emanations

Given the aforementioned threat model, an attacker may opt for EM side-channel attacks, because (1) they are passive and non-intrusive; and (2) they may be mounted at a distance and work through walls. Nevertheless, one challenging problem is to find certain exploitable EM signals which may hide in any place of the spectrum. In this section, we present our newly discovered, exploitable EM emanations that are generated by the memory clock in a modern GPU.

A. EM Signal of GPU Memory Clock

First of all, we show the characteristics of the EM emanations of interest. As mentioned in Section II-B, GPU DVFS defines multiple performance levels, and it switches between these levels in accordance with the current GPU workloads. In other words, the clock frequencies of the GPU cores and memory often change. Clocks usually create strong EM emanations, and hence when a performance level is on/off, we expect to observe the appearance/disappearance of clear EM signals at the corresponding clock frequencies in the spectrum. To verify this anticipated feature, we test several AMD and NVIDIA GPUs made by different vendors and equipped with different types of GDDR, which are listed in Table I. These GPUs are very commonly used, and almost all the recent architectures of AMD and NVIDIA are covered by them, including the latest NVIDIA Ampere architecture.

We alter the performance level of each GPU, and examine the spectral behavior at the corresponding core and memory frequencies. The inspection results are as follows: (1) the EM emanations generated by the GPU core clock can be hardly found; but, (2) the EM signal of the GPU memory clock is very noticeable and its behavior matches our anticipation; and (3) interestingly, the EM signal consists of many frequency components that are over a wide range in the spectrum. For example, Figure 1[4] illustrates (2) and (3) when the first GPU in Table I (i.e., AMD Radeon RX 580) is used.

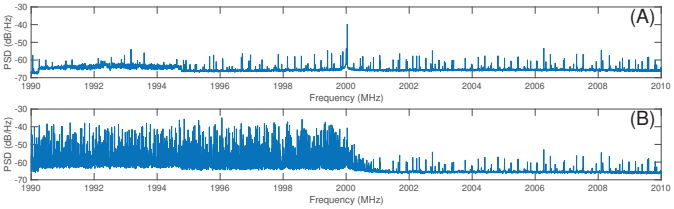

Fig. 1: Spectra around 2000 MHz in the case of using AMD Radeon RX 580: (A) when GPU memory clock frequency is 600 or 4000 MHz, and (B) when frequency is 2000 MHz

Figure 1 (A) shows that we can barely see any signal energy around 2000 MHz if RX 580 memory clock is not set to the corresponding level. On the other hand, Figure 1 (B) shows that we can clearly observe the EM signal of RX 580 memory clock in the frequency-domain when the clock is set to 2000 MHz. Moreover, we can see that the signal has a large number of spectral components in the frequency range below 2000 MHz. The reason for such a phenomenon is due to the use of a hardware feature called spread spectrum clocking (SSC) for meeting electromagnetic compatibility (EMC) regulations. EMC standards impose allowable limits on EM signal energy at any frequency above 30 MHz, and many clock signals (e.g., the GPU memory clock) are strong enough to violate such legal limits. To achieve EMC, SSC uses FM-modulation to vary the clock frequency over a range so that the time spent by the clock signal at a particular frequency is reduced and the energy is spread over that range of frequencies [18].

[4]The power spectral density (PSD) is computed using the Welch's method. The FFT size is 8192, and a Hamming window is used. Ten segments without overlap are averaged.

TABLE I: List of GPUs investigated in this paper

GPU Card (Vendor)	Architecture	Memory	WCK Frequencies	Release Data
AMD Radeon RX 580 (GIGABYTE)	GCN 4.0	8 GB GDDR5	600 MHz, 2000 MHz, 4000 MHz	Apr. 2017
AMD Radeon RX 5600 XT (MSI)	RDNA 1.0	6 GB GDDR6	400 MHz, 2000 MHz, 2500 MHz, 3500 MHz	Jan. 2020
AMD Radeon RX 5700 XT (XFX)	RDNA 1.0	8 GB GDDR6	400 MHz, 2000 MHz, 2500 MHz, 3500 MHz	Jul. 2019
NVIDIA Geforce GTX 1080 (PNY)	Pascal	8 GB GDDR5X	405 MHz, 810 MHz, 4513 MHz, 5005 MHz	May 2016
NVIDIA Geforce GTX 1650 (MSI)	Turing	4 GB GDDR5	405 MHz, 810 MHz, 4001 MHz	Apr. 2019
NVIDIA Geforce RTX 3060 OC (ASUS)	Ampere	12 GB GDDR6	405 MHz, 810 MHz, 5001 MHz, 7301 MHz, 7501 MHz	Jan. 2021

Note that, since GDDR5, GPU memory operates with two types of clocks. One type is referred to as command clock (CK) that is used for sending commands and addresses, and the other one is referred to as write clock (WCK) that is used for data reads and writes. *The EM emanations of interest are specifically generated by the WCK*. In the case of GDDR5, the frequency of WCK is half the data rate [35]. In the cases of GDDR5X and GDDR6, the frequency of WCK is half the data rate if the operating mode is set as double data rate (DDR), or one fourth the data rate if the operating mode is set as quad data rate (QDR) [36].

Since many GPUs have different sets of WCK frequencies, an attacker may exploit this fact to find out what GPU is being used by a victim if this knowledge is unknown to the attacker beforehand. Essentially, the attacker monitors the spectrum at all of the possible WCK frequencies and uses the appearance of the EM signals similar to the one shown in Figure 1 (B) to determine which WCK frequencies the target GPU has. Such reconnaissance information can be used to pinpoint potential GPUs.

B. Activity Identification

To be exploitable, the EM emanations should be computation-dependent so that high-level activities can be inferred to reveal certain sensitive information. We notice that the performance level of a modern GPU is usually changed rapidly to seek a balance between performance and power consumption. Given a computational activity that creates some GPU workloads, there can be multiple GPU performance level switches during the activity. As different activities potentially impose different loads on GPU at different times, distinct performance level switching behaviors should be induced, which can thus serve as activity signatures. In the above discussion, we have learnt the correlation between the performance level switches and the appearance/disappearance of targeted EM signals, which implies that the EM emanations of interest can be leveraged to identify different activities.

To fully capture the behavior of performance level switching, we may try to monitor all of the frequency ranges where the GPU memory clock signals can arise in a synchronized manner. However, our experiments show that such a heavy-weight approach to gaining a complete picture of switching behavior is not necessary, but a partial picture concentrating on when a specific performance level is switched on/off suffices to distinguish different activities. For instance, Figure 2 demonstrates the spectrograms when launching three different programs, that are Chrome, Firefox, and LibreOffice Writer, on a system equipped with an AMD Radeon RX 580 GPU. (The OS is Ubuntu 18.04 and the GPU driver is AMDGPU 20.20.) The frequency range on which we focus corresponds to the second lowest WCK level (c.f. Table I and Figure 1). As we can see from the figure, the patterns of stripe appearance on the spectrograms are distinguishable from each other and we have also verified that they are fully repeatable. Thus, such patterns can be treated as fingerprints to help infer which program is being launched.

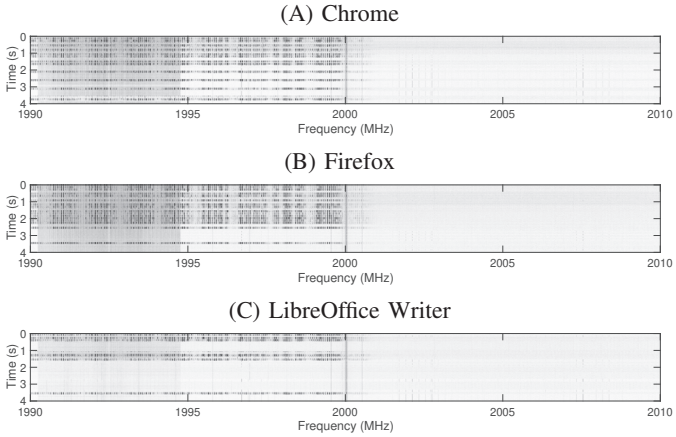

Fig. 2: Spectrograms having frequency range of 20 MHz centered at 2000 MHz when launching three applications on a system equipped with an AMD Radeon RX 580 GPU

Therefore, we need only to consider one specific frequency of WCK and keep track of when the corresponding EM signal energy appears and disappears in the focused frequency range to gain knowledge (e.g., in the form of spectrogram) that can be exploited for activity identification. Theoretically, we may choose any of the possible WCK frequencies, but empirically, we find that the second lowest one generally yields the optimal exploitation results. The reasons for the second lowest WCK frequency being a better choice than others include: (1) this frequency is reached much more often than the other higher ones, especially when an activity does not use the GPU intensively; and (2) we notice that it normally has a higher signal-to-noise ratio (SNR) than the lowest one. Thus, in the following, if not otherwise specified, we focus on the EM emanations generated by the WCK when switched to its second lowest level.

Note that, to further substantiate the tight correlation between the patterns of stripe appearance on the spectrograms in Figure 2 and the WCK frequency switching behavior, we also obtain the traces of GPU memory clock frequency changes via

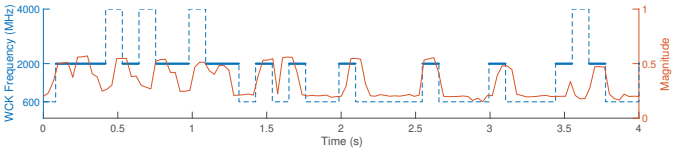

Fig. 3: Traces of WCK frequency alterations and averaged magnitude with respect to Figure 2 (A)

an interface exposed by the AMD GPU driver[5] and compare the traces with the spectrograms. The experiments verify that the patterns well match the frequency change traces, and Figure 3 shows such an example corresponding to Figure 2 (A), i.e., launching Chrome. In Figure 3, we can see that the WCK frequency is switched among its three possible levels (namely the blue dashed line), and we make the line segments solid and bold when the WCK frequency is switched to its second lowest one, namely, 2000 MHz. For each of the spectra constituting the spectrogram in Figure 2 (A), we plot the average of the magnitudes in the frequency range from 1990 MHz to 2000 MHz (namely the red solid line). As we can see from the figure, the local peaks of the averaged magnitude match closely with the line segments indicating that the WCK is switched to 2000 MHz.

C. Propagation Distance and Wall Penetration

To exploit a physical side-channel in an air-gapped setting, we also need to consider how far it can propagate and whether it can go through obstacles like a wall. To this end, we have performed several experiments on the GPUs listed in Table I and found that the EM emanations of interest have a desirable wall-penetrating property and can be measured by an attacker from a distance long enough to carry out attacks practically. The experiments are performed using an SDR device, USRP B210, and an ultra-wideband directional antenna, RFSPACE UWB-3. (We also tried a much cheaper device, LimeSDR, and derived very similar results.) More details about the corresponding setup can be found in Appendix A.

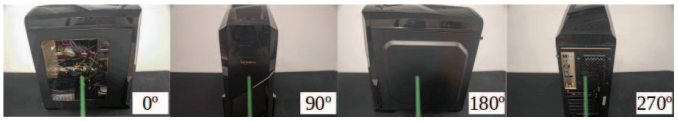

Fig. 4: The definition of directions used in this paper

Figure 5 shows the distances with respect to the directions from which the EM emanations of interest can still be "picked up" when there is no obstacle in between the GPU machine and the antenna. Here we define "pick up" as that the SNR after applying the technique described in Section V is at least 7 dB. In general, we can find that the longest measurement distance changes with the direction. Note that, due to our office space limitation, the maximum distance we can reach is 6 meters. The definition of directions is illustrated in Figure 4. We can see that 0° is defined as when the antenna is orthogonal to the case side from which the motherboard and GPU are installed, and 90° is defined as when the antenna is perpendicular to the front side of the case.

[5]The interface is `/sys/class/drm/card0/device/pp_dpm_mclk` that does not give WCK frequency directly. In terms of AMD Radeon RX 580, we need to double the reading to get the current WCK frequency.

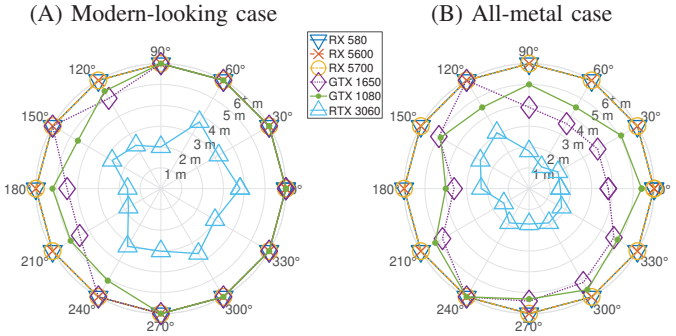

Fig. 5: Propagation distance (no obstacles in between)

Figure 5 shows two scenarios in which two types of computer cases are used. In the first scenario, a modern-looking computer case with a translucent side panel is used (as shown in Figure 4), and in the second scenario, a computer case whose every side is made of metal is used. Comparing Figure 5 (A) and (B), we can observe that the translucent side panel in the first scenario benefits the propagation of the exploitable EM signals. As illustrated by Figure 5 (A), when a modern-looking case is used, we can capture the EM emanations of our interest from more than 3 meters away in almost every direction no matter which GPU is used. On the other hand, in the second scenario, the all-metal computer case can attenuate the strength of the EM signals in the directions from 270° to 90° counter-clockwise, but the signals of our interest can still be picked up at a distance of 3 meters or more in many other directions for each GPU. Notice that, nowadays, modern-looking computer cases with a translucent side panel dominate the market [44] and are in effect extremely popular among users of mid-range to high-end GPUs. Therefore, in reality, it is very likely that an attacker can easily capture the EM emanations of interest at a very far distance. Even if a computer case with all metal sides is used, the exploitable EM signals can still be measured from several meters away.

TABLE II: The signal strength reduction due to the walls

	RX 580	RX 5600	RX 5700	GTX 1080	GTX 1650	RTX 3060
Plaster Wall	-1.06 dB	-1.03 dB	-1.18 dB	-1.32 dB	-0.21 dB	-0.51 dB
Concrete Wall	-4.41 dB	-4.28 dB	-6.92 dB	-8.64 dB	-5.24 dB	-4.50 dB

We also isolate the GPU machine having the modern-looking computer case in two rooms and test if the EM emanations of our interest can be captured from the outside of the rooms. The first one is an office room whose wall is as thick as 15 cm and made of plaster, and the second one is a lab room which has very thick concrete walls (∼15 cm). We measure and compare the strengths of the exploitable EM signals when the antenna and the target machine are separated by 1 m with and without the walls in between. Table II shows the EM signal strength reduction due to the walls. From the

results, we can observe that the plaster wall can only reduce the EM signal strength by at most 1.32 dB while the concrete wall can reduce the strength by up to 8.64 dB. These results indicate negligible effects of normal plaster walls and manageable effects of thick concrete walls on potential attacks exploiting this newly discovered EM side-channel vulnerability.

V. Signal Transformation and Enhancement

Although we may directly exploit the derived spectrograms like the ones shown in Figure 2 to identify activities, it can become very difficult to do so under circumstances where the SNR is too low to induce visible stripes on the spectrograms. To address this problem, we introduce two signal processing techniques that can preserve the target signal's appearance and disappearance patterns even when the SNR becomes very low.[6]

A. Time Series Derivation

As mentioned in Section IV-A, SSC is used to spread the energy of a clock signal over a frequency range for meeting the EMC regulations. Given a clock whose frequency is f_c, SSC in effect scatters its energy to a series of N sub-clocks at $f_c - nf_m$, where $0 \leq n < N$ and f_m is the modulating frequency [52]. Typically, f_m is 30 to 33 kHz. Due to factors like path loss or better EM shielding, the power of such sub-clock signals may become too weak compared to the background noise. Inspired by the work in [52], we leverage the folding technique to amplify the manifestation of the targeted memory clock signal being present.

Assume we perform an M-point discrete Fourier transform (DFT) to derive a spectrum X. Since the frequencies of the sub-clocks are separated by f_m, they should be separated by Δ DFT bins in X, where

$$\Delta = f_m \times \frac{M}{f_s} , \tag{1}$$

and f_s is the sampling rate. Let us define $S[i]$ as the sum of the magnitudes of the i^{th}, $(i-\Delta)^{\text{th}}$, $\cdots$, $[i-(N-1)\times\Delta]^{\text{th}}$ DFT bins of X, namely we have

$$S[i] = \sum_{j=0}^{N-1} X[i - j \times \Delta] . \tag{2}$$

If the clock frequency f_c is located in the k^{th} bin of X, $S[k]$ can be treated as the accumulated energy of all the sub-clocks, and it will reach a much higher value compared to any $S[i]$ where $i \neq k$, given the fact that sub-clocks coherently increase the power at the corresponding frequency locations.

Therefore, given a sequence of sampled values, it is divided into subsequences without overlap and each subsequence has L samples. (We should have $L \leq M$, and if $L < M$, it is expanded to M using zero-padding.) For the i^{th} subsequence, we calculate $S[k]_i$, and again the k^{th} bin in X contains the highest sub-clock's spectral content. The sequence $\{S[k]_0, S[k]_1, \cdots\}$ will be the one-dimensional *time series data* derived from the measured EM emanations. We show an example of this technique in Appendix B.

Note that, although f_m is unknown to us, it can be exhaustively searched given the fact that its search space is small (30 to 33 kHz). An incorrect f_m will not produce noticeably high $S[k]$'s. Another unknown parameter is N, i.e., the number of sub-clocks. Nevertheless, we do not need to know the exact number. The GPU memory sub-clocks generated by SSC span at least 4 MHz, which means that there are at least 121 sub-clocks even when f_m is 33 kHz. Thus, the number 120, which is large enough to make $S[k]$ stand out, may be chosen as N if no other information is available.

Another issue is that although f_c is known and theoretically fixed, it may still vary in a small range due to clock jitters; hence, if the DFT frequency resolution is fine-grained (e.g., in this paper we use 100 Hz), the k^{th} bin computed directly from f_c may not be the one where the highest sub-clock truly locates. To address this problem, we compute multiple $S[i]$'s around k and update k to the one whose result is significantly larger than others.

It is worth highlighting that, aside from coping with the low SNR problem, using this time series derivation technique can also help separate signals generated by multiple similar GPUs. Given two GPUs having the same second lowest WCK, their generated signals in the spectrum can be mingled together; yet, we observe that the k^{th} bin of f_c for one GPU may be different from the k^{th} bin for the other one (e.g., due to imperfection of clocks), and thus two independent traces can be derived. For more detailed discussion, please refer to Appendix C.

B. Strong Noise Contamination Effect Reduction

The second lowest WCK frequency of all NVIDIA GPUs is 810 MHz, and their SSC sub-clocks are distributed in the 800 MHz – 810 MHz frequency band. In certain areas, this band may be too noisy for us. For instance, in North America, the Federal Communications Commission allocates the 614 MHz – 806 MHz frequency band for TV communication use. The contamination induced by such strong background noise makes it very difficult to find the correct sub-clock positions using the aforementioned technique. As an example, Figure 6 (A) shows a spectrum where communication signals exist, and we cannot rely on comparing different $S[i]$'s around the initial k to find where the highest sub-clock truly locates, because one significantly large noise peak can easily dwarf the sum of all the sub-clocks.

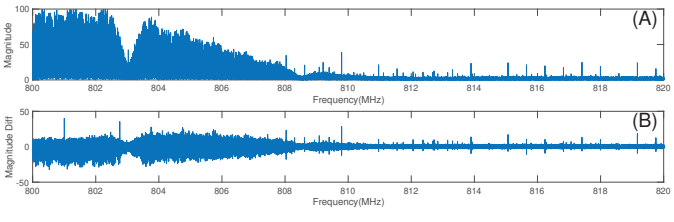

Fig. 6: Spectrum comparison: (A) the original spectrum X, and (B) the derived spectrum X' after the proposed operation is performed

[6]The implementation is available at https://github.com/0x5ec1ab/gpu-mem-em-sig-processing.

To address this issue, we propose to process the spectrum X using a convolution kernel $[-0.5,\ 1,\ -0.5]$. In other words, we derive X' from X as

$$X'[i] = -\frac{1}{2}X[i-1] + X[i] - \frac{1}{2}X[i+1]\ , \tag{3}$$

and replace X with X' in Equation 2. Note that, after this operation, the local peaks originally in X should have positive values in X'; otherwise, negative values. Thus, this operation will pinpoint all the local peaks which include (most of) the sub-clocks. Figure 6 (B) shows the spectrum after the operation is performed on the one in Figure 6 (A).

The reason why using X' can help reduce the negative effect of strong background noise on finding the correct sub-clock positions is that: (1) if the highest sub-clock is in the j^{th} bin, $S[j]$ will sum up the bins which are certainly dependent (as they correspond to sub-clocks), and thus it should be a positive value; but (2) if the highest sub-clock is not in the j^{th} bin, $S[j]$ will be the sum of bins which are independent, and given the fact that the kernel makes the expectation of randomly summing up X' bins be 0, $S[j]$ is very likely to be close to 0 in this case. Therefore, we can still find the correct sub-clock positions even in the presence of strong background noise.

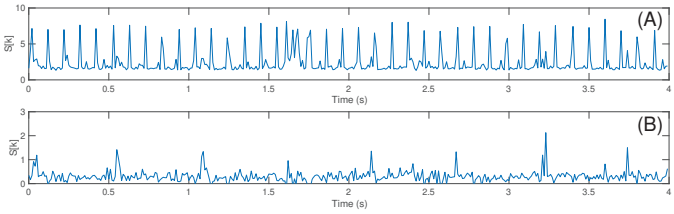

Fig. 7: Time series comparison: (A) time series derived from X's, and (B) time series derived from X''s

Figure 7 demonstrates the effectiveness of our approach in the context of NVIDIA RTX 3060 being used with the existence of strong background noise as in Figure 6 (A). The EM signal of the second lowest WCK should appear around every 0.5 s, namely, there should be a peak in the derived time series about every 0.5 s. However, due to the strong noise contaminating the frequency band of our interest, the peak appearance pattern is completely incorrect in Figure 7 (A). After applying the proposed approach, we can observe the correct peak appearance pattern in Figure 7 (B).

VI. Case Study 1: Website Fingerprinting

In Section IV, we have illustrated that GPU performance level switching patterns derived from the EM emanations of interest can be exploited to identify which application is being launched by a user. To further exemplify the exploitability of this DVFS-induced EM side-channel vulnerability, we show a case study in this section where an attacker can leverage this vulnerability to infer which webpages have been visited by a victim user, namely, to mount a website fingerprinting attack.

A. GPU-Accelerated Webpage Rendering

When browsing websites, the GPU is actually involved in a much more complicated fashion than simply showing pages on the screen. Modern web browsers such as Chrome and Firefox use GPU not only for displaying but also for helping webpage rendering.

Webpage rendering is a procedure that translates an HTML file along with its associated cascading style sheets (CSS) and JavaScript code into a rasterized image. The whole procedure consists of multiple stages: it builds a document object model (DOM) tree, calculates the style for each DOM node, creates the layout of the page, separates the DOM-represented page into layers, rasterizes each layer, and combines the rasterized results into a final screen image [24]. In such a complicated procedure, GPU is often leveraged to accelerate operations that involve large numbers of pixels. For example, a layer is normally divided into a grid of tiles, and these tiles need to be rasterized into bitmaps which are then uploaded to the GPU as textures. In the presence of GPU-accelerated rasterization, the GPU may be directly used to rasterize many tiles into textures, based on certain heuristics (e.g., if the tiles can be affected by animations or transition effects, it is better to employ the GPU). In addition, the GPU can be used to accelerate compositing textures into screen images.

B. EM-Based Website Fingerprinting

As stated above, GPU is extensively used during webpage rendering in a modern web browser. Since different webpages have different designs and contents, rendering them are likely to have different GPU workloads generated. In the light of the investigation presented in Section IV-B, such differences in workloads should be able to induce different patterns of GPU performance level switches, and these patterns can be captured approximately through monitoring the EM emanations of the GPU memory clock at a specific frequency. Exploiting such derived patterns, we should be able to distinguish the rendered webpages from each other (i.e., fingerprinting).

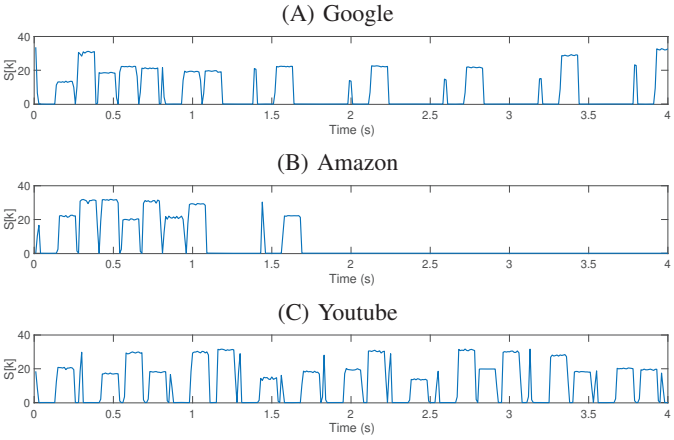

Fig. 8: Time series derived from the EM emanations that are measured when opening three websites using Chrome on a system equipped with an AMD Radeon RX 580 GPU

Specifically, we monitor the EM emanations generated by the WCK at its second lowest frequency and leverage the techniques described in Section V to derive time series to fingerprint webpage rendering activities. To illustrate this, we use three popular websites, Google, Amazon, and Youtube, as an example, and compare the time series derived from the signals captured when opening these three websites in Chrome on a system equipped with an AMD Radeon RX 580 GPU. Chrome uses GPU-accelerated webpage rendering by default. Accordingly, rendering the homepages of these three websites should create different GPU workloads, as their contents differ significantly (e.g., Google homepage is more concise, Amazon has more animations, and Youtube is populated with videos). Figure 8 shows the corresponding time series, and as expected, we can notice clear differences between them.

From Figure 8, we can observe that the peaks in the time series data appear very frequently corresponding to Youtube and Google. In terms of Youtube, when opened, it has some video being played, which will continuously employ the GPU for displaying (or even decoding); yet, we can still see the adjustment of the performance level due to GPU DVFS. In terms of Google, it has a blinking text cursor in its input box, whose blinking rate is about 1.67 Hz; and every time it blinks, the affected tiles need to be re-rasterized and screen image needs to be re-composited by the GPU. We can see that after the initial 1 s, the interval between each wide peak in Figure 8 (A) is about 600 ms, which matches the periodic blinking behavior of the cursor. Although not shown in Figure 8 (B), Amazon also induces periodic peak appearance in the time series data at about 0.2 Hz due to its animated advertisement pictures that are switched about every 5 s.

Notice that users normally tend to have multiple tabs opened in a browser. We find that using multiple tabs does **not** affect our website fingerprinting at all. The reason is that popular browsers like Chrome and Firefox only send the workloads of the currently focused tab to the GPU for optimizing resource utilization. To verify this, we use Chrome or Firefox to open a website in a tab while having several other tabs with Youtube playing videos, and we confirm that the EM signal pattern of interest is not disturbed by other unfocused tabs. An example is given in Figure 9, where we are watching a Youtube video in a Chrome tab (0 – 2.8) until we open a new tab (2.8 – 3.2) with the video still being played in the prior tab, and then we open Twitter in this new tab (4.2 – 5.4). The figure illustrates the above statement clearly. Moreover, the figure further shows that we can identify possible start points of webpage rendering in a long trace.

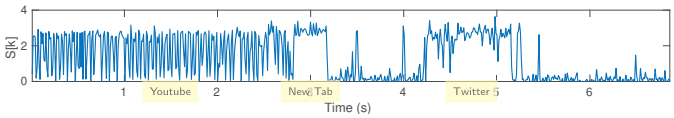

Fig. 9: Time series corresponding to watching a Youtube video in a Chrome tab and later opening another website in a new tab (performed *w.r.t.* NVIDIA RTX 3060)

While using multiple tabs has no impact, changing browser window size may affect the webpage rendering workloads and thus has a negative impact on our website fingerprinting. Even so, it is observed that most people (if not all) just use the full window size when using a web browser. Therefore, this impact may not be critical in reality.

C. Evaluation

Because our main intention is to showcase the exploitation of the newly discovered EM side-channel, the evaluation is just performed in a closed-world scenario, where a victim is assumed to visit a list of popular websites and an attacker tries to pinpoint the webpages browsed by the victim from the set of possibilities. The open-world scenarios need novelty detection, which is left as our future work, where we may use some recently proposed techniques [46], [49].

We evaluate the EM-based website fingerprinting technique on all the GPUs listed in Table I. The evaluation is focused on Chrome web browser, since it dominates the market share[7]. In the evaluation, no Chrome extensions are installed. Note that, as studied in [55], some browser extensions can affect what is rendered on a webpage and hence affect the generated GPU workloads such that the GPU performance level switching patterns become affected. In practice, an attacker may need to take into account the popular extensions (e.g., AdBlock and Ghostery) during profiling.

We try to use different operating systems, but we surprisingly observe that all the AMD GPUs under Windows seldom change the performance level when opening a website. (This phenomenon should be caused by its driver, and we leave the further inspection to our future work.) Therefore, we evaluate the attack on AMD GPUs only under Linux, where the official driver AMDGPU 20.20 is used. In contrast, the attack can be mounted against all the NVIDIA GPUs under either Windows or Linux. We pair NVIDIA GTX 1080 with Linux where the official Linux driver 450.51.06 is installed, and pair NVIDIA GTX 1650 with Windows where the official Windows driver 461.40 is installed. For NVIDIA RTX 3060, we evaluate the attack under both Windows and Linux. Table III summarizes these circumstances.

TABLE III: Feasibility of website fingerprinting attack under Windows and Linux

	RX 580	RX 5600	RX 5700	GTX 1080	GTX 1650	RTX 3060
Windows	✗	✗	✗	✓	✓*	✓*
Linux	✓*	✓*	✓*	✓*	✓	✓*

The symbol ✓ indicates that website fingerprinting can be performed.
The symbol ✗ indicates that website fingerprinting cannot be performed.
The symbol * indicates that the evaluation is performed in this section.

We use an USRP B210 SDR and a RFSPACE UWB-3 antenna to capture the EM signals of interest, and we use the GNU Radio to manage the entire measurement process and process the captured raw data. The SDR is tuned to the second lowest WCK frequency of the corresponding GPU, and is set

[7]According to NetMarketShare, Chrome has around 70% browser market.

to use a 25 MHz sampling frequency[8]. More details about this setup can be found in Appendix A.

TABLE IV: Different spots where EM signals are measured

Nearby Spots	Distance	Direction	Faraway Spots	Distance	Direction
N1	0.5 m	315°	F1	3 m	315°
N2	1 m	315°	F2	6 m	315°
N3	0.5 m	0°	F3	3 m	0°
N4	1 m	0°	F4	6 m	0°
N5	0.5 m	30°	F5	3 m	30°
N6	1 m	30°	F6	6 m	30°
N7	0.5 m	60°			
N8	1 m	60°			

We select 50 websites according to Alexa Top Sites, which are listed in Appendix E. For each website, we measure the EM signals from different directions (which are 315°, 0°, 30°, and 60°) at different distances (which are 0.5 m, 1 m, 3 m, and 6 m), as listed in Table IV. At each spot, we measure the EM emanations for 8 seconds when its webpage is opened, and we repeat this process for 50 times. (To facilitate data collection, we use a script that repeatedly lets the current tab return to the blank page, waits for 5 seconds, and opens the target webpage as well as notes the time for latter trace alignment.) For each measured signal, we use the techniques described in Section V to generate a time series. Given the 25 MHz sampling rate, we use $L = M = 250,000$, namely, each subsequence has 250,000 samples and a 250,000-point DFT is used, which means that the DFT bin resolution is 100 Hz and each derived point $S[k]$ represents 10 ms.

1) Nearby Scenario: We start with a nearby scenario, where only the EM signals measured at N1, N2, N3, and N4 are used to train a classification model, and the EM signals measured at N5, N6, N7, and N8 are used for testing. Given the fact that the EM signals have been transformed into time series, we adopt the ResNet model from [59], whose architecture is duplicated in Appendix D. (As for each website there are 50 EM signals measured at each spot, there are 200 time series for training and 200 time series for testing with respect to each website.) The evaluation results in terms of accuracy are shown in Table V, and the confusion matrices are shown in Appendix F.

TABLE V: Fingerprinting accuracy in the nearby scenario

	RX 580	RX 5600	RX 5700	GTX 1080	GTX 1650	RTX 3060_1	RTX 3060_2
N5	85.9%	83.3%	74.3%	81.9%	84.0%	80.7%	56.7%
N6	84.0%	86.0%	79.5%	88.2%	85.4%	61.3%	62.7%
N7	85.3%	82.6%	74.0%	73.3%	83.6%	72.4%	67.6%
N8	86.0%	83.2%	70.4%	72.4%	78.6%	70.0%	68.5%
Avg.	85.3%	83.8%	74.6%	79.0%	82.9%	71.1%	63.9%
Std.	0.8%	1.3%	3.2%	6.5%	2.6%	6.9%	4.7%

Note that, in reality, an attacker can profile the EM signals from any direction at any distance. Therefore, this scenario is *biased against* attackers, and the evaluation *underestimates* the achievable accuracy. Even so, from the results, we can see that the averaged classification accuracy is above 63% in all cases, and it reaches 85.3% in the case of RX 580. Given a test example, if we randomly guess which of the 50 websites it corresponds to, the accuracy will be only 2% (i.e., 1/50). Thus, the results signify that an abundant amount of information can be leaked through this new EM side channel. (The better and also more pragmatic results after relaxing the limitation on the capability of attackers can be found in Appendix G.)

An interesting case is to compare the results corresponding to using the same GPU but under different OSes. The last two columns of Table V show such a case, where NVIDIA RTX 3060 is evaluated under Windows (the last column) and Linux (the second last column). We can observe that, except for the anomalous spot N5, the performance for any other spot appears to be very similar, although it is slightly better under Linux. We also use the Linux-related model to classify the data captured under Windows and vice versa, but interestingly the accuracy is just slightly better than random guessing (4.3% and 5.6% respectively). This means that factors like drivers and Chrome engines for different OSes can strongly affect the GPU DVFS behavior when rendering webpages. Notice that, an attacker can simply perform profiling against both OSes and combine the training data, and thus we train a single model in such a manner to test the performance. The resultant accuracy becomes 64.1%, which is very similar to the one in the last column (i.e., 63.9%). Therefore, this fingerprinting can work no matter which OS is used on the target.

AMD RX 5600 and RX 5700 do differ but both of them are based on AMD RDNA 1.0 architecture. We attempt to use the models trained for them to cross fingerprint each other. The accuracy of using the model trained for RX 5600 to fingerprint the signals of RX 5700 is 63.3%, and the accuracy of using the model trained for RX 5700 to fingerprint the signals of RX 5600 is 50.6%. Even though they are made by different vendors and use different GPU chipsets, we can still obtain reasonable website fingerprinting results. This implies that an attacker can use the model trained with respect to his/her own GPU to fingerprint the signals of another similar GPU (of course, more accurately if two GPU chipsets also match).

2) Faraway Scenario: Next, we perform evaluations in a faraway scenario, where the EM signals measured at F1, F2, F3, and F4 are used to train a classification model, and the EM signals measured at F5 and F6 are used for testing. In addition, we test signals measured at N6 using this model. We evaluate two GPUs that are AMD RX 580 and NVIDIA GTX 1650. We place GTX 1650 in the modern-looking computer case, while we place RX 580 in the all-metal computer case (see Appendix A for details about these two computer cases). The evaluation results are shown under "Faraway Scenario 1" in Table VI, and the confusion matrices are shown in Appendix F.

TABLE VI: Fingerprinting accuracy in the faraway scenarios

	Faraway Scenario 1		Faraway Scenario 2	
	RX 580	GTX 1650	RX 580	GTX 1650
F5	80.1%	95.4%	83.6%	95.6%
F6	78.3%	93.2%	80.4%	94.1%
N6	70.9%	83.9%	87.0%	94.8%

Similar to the previous nearby scenario, the evaluation also *underestimate* the achievable accuracy. Nevertheless, we can observe that the accuracy is very high, and in terms of NVIDIA GTX 1650, the resulting performance at far distances is even much better than that at near distances (e.g., it reaches 95.4% at 3 m and 93.2% at 6 m in comparison to 85.4% at 1 m). We

[8]Since quadrature sampling is used, it provides 25 MHz bandwidth.

discuss a possible reason for this phenomenon in Appendix H. Moreover, in this scenario, training examples are neither from N6's direction nor around its distance, but the result *w.r.t.* N6 is comparable to that in Table V for GTX 1650 and fairly decent for RX 580. The results indicate that, as long as the EM signals of interest can be picked up, the differences in direction and distance are generally tolerable.

In the second faraway scenario, we include the EM signals measured at N2 and N4 for training as well, and still test the EM signals measured at F5, F6, and N6. The evaluation results are shown under "Faraway Scenario 2" in Table VI. We can observe that after expanding training examples with the ones measured at nearby spots N2 and N4, the accuracy *w.r.t.* faraway spots F5 or F6 does not change much, but it becomes much higher *w.r.t.* N6. The results indicate that, between the two profiling factors distance and direction, distance affects performance more. Thus, if resources are limited, an attacker should choose distance over direction during profiling.

VII. Case Study 2: Keystroke Timing Inference

In this section, we present the second case study we have conducted on the exploitation of the DVFS-induced EM side-channel vulnerability, which is to detect the keystroke events and learn the time between successive keystrokes, namely, a keystroke timing inference attack. Even though such an attack cannot directly recover the specific keys pressed by a user, it is still treated as a type of keylogging [39], because the knowledge about the keystroke timing can be exploited to infer the typed passphrases or other words [54], [62]. Thus, this attack poses a greater hazard to security and privacy.

When combined with the website fingerprinting attack studied in the last section, it can even cause more serious violation of user privacy. For instance, when it is detected that a user has opened the login page of some website, the attacker can try to recognize the length of typed username and password through the number of identified keystrokes and the attacker can further try to infer the details of such items via the timing information using some well-studied statistical techniques [39], [54], [62].

A. Keystroke Detection

If we can detect the keystroke events and mark them precisely on the time axis, it will be a straightforward task to learn the time between successive keystrokes. Hence, we investigate if keystrokes are detectable from the EM emanations of the GPU memory clock, especially during the time when a user is typing on a webpage.

In essence, typing in a text box on a webpage makes the affected tiles of the corresponding layer re-rasterized and the final screen image re-composited. As previously mentioned, a browser often delegates the computation generated by these operations to GPU for acceleration. According to our earlier observations, the GPU performance level will be consequently changed by DVFS, and such level switches can be captured by monitoring the EM emanations of the GPU memory clock. Therefore, we expect that the keystroke events can be detected by exploiting the DVFS-induced EM side-channel vulnerability in modern GPUs.

To verify this hypothesis, we carry out several experiments. Firstly, we use keyboard activity generation tools to create a sequence of fake keystrokes regarding certain patterns and check if the appearance of the EM signals of the GPU memory clock match these patterns. Figure 10 shows an example of this experiment performed on NVIDIA GTX 1080, where we use a script to repeatedly generate a sequence of 'a', 'b', and 'c' in the search box of Google. After each 'a', there is a 200 ms pause; after each 'b', there is a 350 ms pause; and, after each 'c', there is a 500 ms pause. The time series in Figure 10 is derived using the techniques described in Section V. From this figure, we can easily see that the appearance of peaks match the 'a', 'b', 'c' keystroke timing pattern.

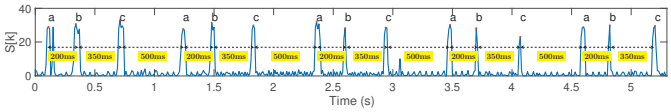

Fig. 10: Keystrokes generated using xdotool with a predefined pattern on Google (performed *w.r.t.* NVIDIA GTX 1080)

Next, we ask three people to quickly type "`username`" and "`password`" in the corresponding boxes on the Facebook login page. The keyboard is Dell L30U. Figure 11 illustrates the processed EM signals of interest in terms of NVIDIA RTX 3060 when the fastest typist among the participants is typing. From the results, we confirm that the keystroke events can be correctly detected and the inter-keystroke timing information can be precisely derived. Notice that, even though we are able to determine the time between the keystrokes by exploiting the EM emanations of interest, we have not found any correlation between the signal and the typed characters.

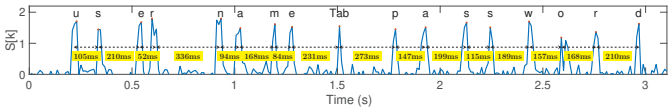

Fig. 11: Keystrokes typed by a user on Facebook (performed *w.r.t.* NVIDIA RTX 3060)

We also need to mention that an attacker could have a hard time to correctly detect keystroke events if a user were typing on a webpage that is crowded with animations. Nevertheless, this limitation does not restrict an attacker too much in reality from learning the inter-keystroke timings related to typing very valuable information like usernames and passwords, because it is auspicious for the attacker that the login or sign-up pages of many popular or important websites are deliberately designed to be plain and simple (e.g., Google, Amazon, Twitter, PayPal, Bank of America, and so forth).

B. Evaluation

In this evaluation, we mainly explore how close in time two successive keystrokes can be such that they are still detectable

via exploiting the discovered DVFS-induced EM side-channel vulnerability. To facilitate the evaluation, we use the keyboard activity generation tools to create fake keystrokes, since they are certainly much more precise in time than manual inputs. As mentioned above, we focus on deriving the time between successive keystrokes when typing on webpages in a browser. Thus, the evaluation is primarily conducted in such a scenario. Like in Section VI, we exclusively use Chrome as the browser. We choose Google homepage and PayPal login page to be the representative venues for the evaluation.

In terms of the setup of signal measurement (e.g., SDR and antenna), it is the same as that described in Appendix A. The captured EM signals are still processed using GNU Radio. Currently, we do not use any automatic approach to identifying the keystroke events in the processed data but only perform a manual analysis.

TABLE VII: How close in time two keystrokes could be such that they are still distinguishable from each other

	RX 580	RX 5600	RX 5700	GTX 1080	GTX 1650	RTX 3060
Interval	150 ms	N/A	N/A	50 ms	70 ms	30 ms

We create sequences of keystrokes with the inter-keystroke time interval being 10 ms, 20 ms, 30 ms, $\cdots$, respectively. We test each sequence on each GPU target machine listed in Table I to check if all the keystrokes in the sequence can be detected via the peak appearance and disappearance patterns in the derived time series. Table VII shows the evaluation results.

From the results, we can see that NVIDIA RTX 3060 has the highest time resolution, where keystrokes at 30 ms intervals can be clearly recognized. (When it is lower than 30 ms, e.g., 20 ms, more than 90% keystrokes can also be recognized.) In terms of other NVIDIA GPUs, GTX 1080 and GTX 1650, a high resolution can also be achieved.[9]

Compared to NVIDIA GPUs, the timing resolution in terms of AMD RX 580 is much coarser, that is almost 150 ms. The reason for this discrepancy is that when RX 580 is at the second lowest performance level, it appears to stay there longer than those tested NVIDIA GPUs. Hence, if two or more keystrokes occur very closely in time, they can be treated as one keystroke event. (Nevertheless, a recent study on human typing behavior and performance has revealed that 250 ms inter-keystroke time interval is already very fast for many normal people [9].) The interesting cases are AMD RX 5600 and RX 5700, on which we cannot mount the discussed attack. Even though their performance level changes correspondingly when a webpage is being rendered (as shown in Section VI), we find that this switching behavior seldom happens when typing on webpages (namely, the GPU workloads due to normal keystrokes do not always cause the DVFS of high-end AMD GPUs to bump up their performance level). Therefore, we may not exploit the DVFS-induced EM side-channel vulnerability for this attack when these AMD GPUs are used.

[9]Additionally, we temporarily borrow two other NVIDIA GPUs, RTX 2060 Super and RTX 2080, from another group, and find that they behave the same as RTX 3060 and can reach 30 ms.

VIII. Ineffectiveness of Disabling GPU DVFS

To mitigate the aforementioned exploitation possibilities, a straightforward approach is to disable GPU DVFS by setting the GPU to run at a specific performance level. However, such a countermeasure has two major problems. The first one is that this countermeasure hurts either performance or energy efficiency. If a relatively low performance level is selected, it will contradict with the purpose of using the GPU for acceleration; yet, if a high performance level is selected, it will be highly energy-inefficient. The second and much severer problem is that, when NVIDIA GPUs are used, this countermeasure will unfortunately introduce another highly exploitable EM side-channel vulnerability.

A. AM-Modulated EM Emanations

The reason for such ineffectiveness is that we have discovered a new type of exploitable EM emanations appearing when the performance level of an NVIDIA GPU is fixed. Given an NVIDIA GPU, a user may use tools included in the official driver to set its performance level to be maximum. (Unlike AMD GPUs whose driver allows us to fix the performance level at any defined one, the performance level of NVIDIA GPUs can only be fixed at the highest.) We find that when the performance level is fixed as such, there appears strong EM emanations that are inadvertently AM-modulated by the GPU memory accesses. In other words, the strength of these EM emanations varies when the amount of data reads and writes changes.

Interestingly, the EM emanations are around the frequency that is one eighth the data rate in the cases of all NVIDIA GPUs we have tested (i.e., GTX 1080, GTX 1650, and RTX 3060). Although we do not know the exact cause of such EM signals at the moment, an educated guess is that they are created by some clock driving certain components in the GPU memory system. We leave the search for this clock to our future work.

Since the emerging EM emanations will be AM-modulated by the GPU memory accesses, even though the DVFS-induced EM side-channel vulnerability were removed by using a fixed GPU memory clock frequency, information about the patterns of GPU memory traffic would be encoded into these new EM emanations, which can be exploited to effectively identify the high-level activities. Essentially, such EM emanations act as a modulated carrier signal that bears the modulating activity information and propagates to large distances.

As an example, Figure 12 shows the above-mentioned carrier signal of interest that emerges when we set the performance level of NVIDIA GTX 1650 to its maximum. Given the 8 Gbps data rate of GTX 1650 at its maximum level, the EM carrier signal on which we focus will be at 1000 MHz. Yet, from Figure 12 (A), which illustrates the carrier signal of interest in the frequency-domain, we can observe that it has a number of spectral components in the frequency range from 996 MHz to 1000 MHz. Recall the discussion in Section IV-A that SSC is used to vary the frequency of a clock over a range for meeting EMC regulations. These components spread over

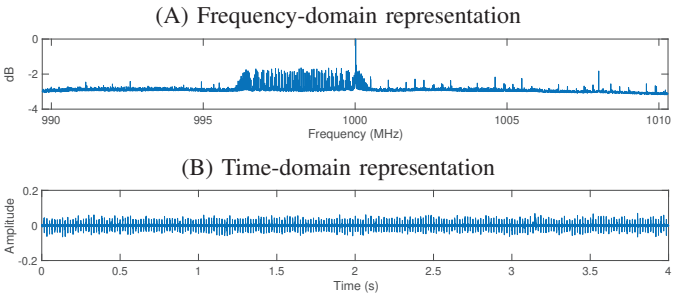

Fig. 12: Carrier signal emitted by NVIDIA GTX 1650 that can be AM-modulated by GPU memory accesses

the 4 MHz range in Figure 12 (A) are caused by SSC, which indicates that the carrier signal is due to a clock. Figure 12 (B) shows the signal in the time-domain and its amplitude will change over time when AM-modulated.

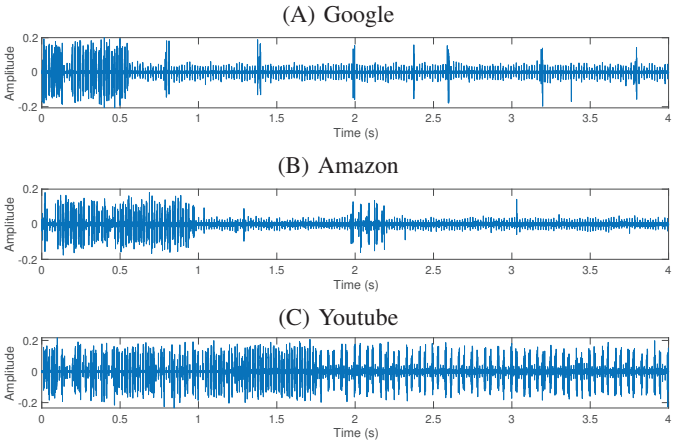

Fig. 13: The amplitude of carrier signal is modulated when opening three websites in Chrome

To exemplify that the EM signal shown in Figure 12 can be AM-modulated and exploitable, we examine the signal when browsing different websites using Chrome. Like the example demonstrated in Section VI-B, we use the three most popular websites, Google, Amazon, and Youtube. As the performance level is fixed this time, we cannot exploit its switching patterns to discern these websites. However, Figure 13 shows that the EM signal arising after the DVFS-induced EM side-channel vulnerability is mitigated can be exploited to infer the website identities, as its amplitude is modulated by the GPU memory accesses during webpage rendering, which should be different in general if different pages are being rendered. Comparing Figure 8 and Figure 13 with respect to the time axis, we can note resemblances. For instance, the 1.67 Hz blinking cursor behavior on Google search page is obvious in both cases. The AM-modulated signal of interest certainly carries information fine-grained enough for being exploited.

Notice that, when GPU DVFS is disabled, we have found this EM side-channel vulnerability only in NVIDIA GPUs but not in AMD GPUs. Nevertheless, it does not mean that disabling DVFS is a completely effective countermeasure in terms of AMD GPUs. There may be some other undiscovered EM vulnerabilities emerging when DVFS is disabled on AMD GPUs. We will discuss some possible countermeasures in the next section.

B. Evaluation and Discussion

To show the exploitability of such AM-modulated EM emanations, we evaluate the performance of website fingerprinting attack on NVIDIA RTX 3060 with Windows being used as the OS. The evaluation setting is exactly the same as the one used in Table V, namely, the signals measured at N1, N2, N3, and N4 are used to train a classification model and the signals measured at N5, N6, N7, and N8 are used for testing. At each spot, it is still 50 signals being captured for each website.

The signal measurement setup is the same as that in Section VI-C with one exception – we tune the SDR to center at 1875 MHz, as it is the frequency that is one eighth the corresponding data rate when we set the performance level of RTX 3060 to be the highest. We directly use the measured signal in the time-domain, as shown in Figure 13. Since it is also time series data, we still use the same ResNet classification model. The evaluation results are shown in Figure 14.

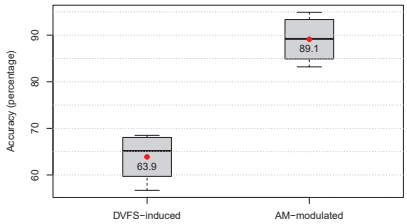

Fig. 14: Website fingerprinting accuracy comparison of exploiting DVFS-induced and AM-modulated EM emanations from NVIDIA RTX 3060 with Windows being the OS

From the results, we can find that the accuracy is as high as 89.1%, which is much better than that exploiting the DVFS-induced EM emanations (i.e., 63.9%). Such good performance indicates that the AM-modulated EM emanations can be exploited to effectively mount website fingerprinting attacks. Even compared with the last column in Table IX in Appendix G where data collected at the other seven spots is used for training, we can observe that the accuracy here is much higher. If considering only the fact that disabling GPU DVFS is actually used as a countermeasure, it would be unexpected to find that such a mitigation method helps rather than thwarts attacks.

We have also verified that we are able to perform keystroke timing inference by exploiting such AM-modulated EM signals. In this case, two keystrokes separated by 20 ms are still well distinguishable no matter which NVIDIA GPU is used.

Our conjecture about the considerably increased attack performance when leveraging the AM-modulated EM emanations is that the granularity of information carried in such EM signals is much finer than the DVFS-induced ones. Although GPU DVFS is rapid enough to reveal changes in

GPU workloads, it cannot enable us to pry more detailed activities inside each workload. In the specific case of website fingerprinting attack, if some finer-grained information can be acquired, it may help distinguish webpages that generate similar sequences of GPU workloads during their rendering. By contrast, the emanations whose strength is AM-modulated by GPU memory accesses inevitably contain information about the modulating memory activities of each workload. Thus, given the webpages which are often misclassified as each other when the DVFS-induced EM side-channel vulnerability is exploited, they become more discernible from each other when using the AM-modulated EM emanations.

Notice that, the EM side-channel vulnerability presented in this section and the DVFS-induced one cannot be exploited at the same time. We find that the AM-modulated vulnerability emerges only when the performance level is fixed. Therefore, although the AM-modulated EM side-channel vulnerability can be leveraged to achieve more effective attacks, it is the DVFS-induced one that will be exploited in many realistic situations, because, most of the time, a user will not change the settings on GPU DVFS which is active by default. However, as we have seen in this section, if the GPU DVFS is turned off, the other EM vulnerability becomes available for being exploited.

IX. Countermeasures

As described in Section VIII, the straightforward mitigation approach that disables GPU DVFS by fixing the performance level fails to effectively prevent information leakage from many GPUs. In this section, we propose several other countermeasures that we believe can potentially mitigate the EM side-channel vulnerabilities of GPU.

The first mitigation direction is to try to significantly lower the SNR of the exploitable EM emanations that an attacker can measure. To achieve this, we may have different options (e.g., using an RF device to generate some EM noise over the targeted frequency range), but a plausible and attainable method is to shield the computer for reducing the intensity of the emitted EM signals. For example, we may be tempted to tape some shielding Faraday fabric on the computer case sides. We have performed such experiments and found that, if we just naively rely on the patterns of stripe appearance on the spectrograms for exploitation, this can indeed make far-off attacks much harder or even impossible. *However, if we apply the techniques proposed in Section V first, we can still acquire the target signal's appearance/disappearance patterns even when the SNR becomes very low, which defeats the purpose of having EM shielding in such a simple way.* Therefore, more carefully engineered EM shielding computer cases are needed. Moreover, hardware manufacturers should devote more efforts to minimizing EM emissions in their future product designs, as replied by NVIDIA to us.

The second mitigation direction is to make the granularity of the EM side-channel information much coarser. For this purpose, we may try to reduce the sensitivity of GPU DVFS to workload changes. In such a case, the performance level may not be switched during activities like webpage rendering, and thus the leaked information will be coarse-grained so that very little sensitive information can be inferred. Although this method does not try to eliminate side-channel information, it can reduce the entropy of that side-channel and thus reduce the overall exploitability. Since this will hurt energy efficiency, a problem is how to find a good trade-off between security and power consumption.

In addition to reducing the SNR of the EM signals and increasing the granularity of the side-channel information, a third direction is through obfuscation. For instance, when a sensitive activity is using GPU, we may deliberately generate some random GPU workloads to disrupt its original GPU use pattern such that the predictability is reduced. With respect to the website fingerprinting attacks studied in this paper, we may implement a thread in the browser to randomly use GPU for some dummy computation during webpage rendering via certain interfaces like WebGL. In terms of Chrome in particular, we may simply create an extension to send random requests to the GPU process which is the specific process in Chrome for managing interactions with GPU.

X. Related Work

EM emanations generated by computing devices have been extensively exploited to carry out various side-channel attacks, e.g., stealing cryptographic keys [2], [12]–[14] and inferring private information [10], [25], [32]. (EM emanations have also been exploited to construct covert communication channels for data exfiltration [16], [61].) As mentioned in [4], many of the EM side-channel attacks are only short-range given the fact that most exploitable EM emanations are weak near-field ones. To perform long-range attacks, Camurati *et al.* have discovered that wireless communication signals may be exploited, as such RF signals can be inadvertently AM-modulated (in a cascaded manner) by the EM emanations from the digital circuit of the mixed-signal chip due to substrate noise coupling [4]. (Later, Wang *et al.* improved Camurati's work by using deep learning-based analysis techniques [58].) Moreover, Callan *et al.* have observed that exploitable information may be carried by some strong EM signals (generated by certain voltage regulators or clocks) in the form of AM-modulation [3], which has been leveraged in attacks and defenses [43], [50], [61], [64].

Similar to our work, Sehatbakhsh *et al.* have also exploited EM emanations involved with power management for attacks very recently [51]. While there are similarities, fundamental differences exist. *First*, our work leverages the effect of DVFS on GPU clocks, whereas their work uses the effect of demand-based switching on CPU voltage regulators. *Second*, our work also studies how to increase the SNR of the EM emanations of interest to make them still exploitable even in the presence of EM shielding and strong noise contamination. *Third*, aside from the overlap in terms of inferring keystroke timing, the two works demonstrate different attacks, where the work presented in [51] builds covert channels for data exfiltration and our work showcases website fingerprinting. *Finally*, our work illustrates

that disabling DVFS as a countermeasure may not be fully effective.

Given the rising popularity of modern GPUs, research on their security implications has begun drawing attentions in recent years. Several studies exploit possible residues in GPU memory that may not be properly cleared after use. Pietro *et al.* have shown that the lack of memory-zeroizing operations can be exploited to attack CUDA AES implementations [47], and the memory residual leakage vulnerabilities in virtualized and cloud computing environments have been investigated by Maurice *et al.* [33]. Furthermore, Zhou *et al.* have studied how to extract raw images from GPU memory residues [65]. In [27], Lee *et al.* have successfully inferred the web browsing history by examining GPU memory dumps.

Other than relying on GPU memory residues, many attacks exploit possible logical or physical side-channel information of GPU to breach confidentiality. Jiang *et al.* have studied the timing differences due to contentions on shared GPU caches or memory banks and leveraged such timing side-channels to break AES implementations on GPU [20], [21]. Luo *et al.* have demonstrated that CUDA RSA implementations are also susceptible to timing side-channel analysis [29]. Aside from logical side-channel information, Luo *et al.* have also used GPU power consumption traces for cryptanalysis of CUDA AES [30], while Gao *et al.* have used near-field localized EM emanations from GPU for the same purpose [11]. In addition to subverting GPU-accelerated cryptosystems, Naghibijouybari *et al.* have studied GPU side-channel attacks in a more general context, where they used performance counters or resource tracking APIs to measure shared GPU resource contentions and exploited such information to fingerprint websites, infer user activities, and reverse engineer neural networks [41].

GPUs can also be a new venue of which malware takes advantage to increase its stealthiness [56]. In [26], Ladakis *et al.* have implemented a piece of GPU-based malware that directly monitors the keyboard buffer from GPU via DMA to log keystrokes, and in [66], Zhu *et al.* have conducted a more comprehensive study on such a topic. In [40], Naghibijouybari *et al.* have showed that malware may exploit contentions on shared GPU resources to construct covert channels for data exfiltration. In [7], Davidov and Oldenburg have also exploited GPU's EM emanations, but they only showed a malware-enabled covert channel on an AMD GPU.

Website fingerprinting and/or keystroke timing attacks have been investigated to a great extent in the past. Other than network traffic analysis, an attacker may exploit certain logical side-channel information, such as cache timings [45], [53], memory footprints [19], storage usage statistics [22], shared event loops [57], and interrupt timings [28], to achieve such attacks. Like our work, physical side-channel information has also been studied for this malicious purpose [6], [15], [63]. In particular, an EM-based website fingerprinting attack is shown feasible in a mobile setting by Matyunin *et al.* [32], in contrast to which, our work focuses on scenarios where discrete GPUs are used.

XI. Conclusion

In this paper, we have presented our newly discovered EM side-channel vulnerability that exists in many modern GPUs and conducted two case studies, website fingerprinting and keystroke timing attacks, to demonstrate that this new EM vulnerability is highly exploitable. Even though the root cause of this vulnerability is identified as the commonly used DVFS feature in GPU, we have shown that simply disabling DVFS by setting GPU to a specific performance level may not be an effective countermeasure since another AM-modulated EM vulnerability emerges. We have also discussed some potential mitigation approaches.

As research on information leakage vulnerabilities of GPU has just started lately, we believe that the currently disclosed ones and their exploitation represent only the tip of the iceberg, and many other exploitable ones lurk in the darkness. The study carried out in this paper argues for more rigid evaluation on the security of GPU from different perspectives.

Acknowledgment

This work is supported in part by the National Science Foundation (NSF) grants CNS-2147217 and CNS-1739328. The authors would like to thank the anonymous reviewers for their comments and suggestions that help us improve the quality of the paper.

References

[1] Y. Abe, H. Sasaki, S. Kato, K. Inoue, M. Edahiro, and M. Peres, "Power and Performance Characterization and Modeling of GPU-Accelerated Systems," in *2014 IEEE 28th International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2014, pp. 113–122.

[2] 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 (USENIX Security 18)*, 2018, pp. 585–602.

[3] 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)*, 2015, pp. 592–603.

[4] 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)*, 2018, pp. 163–177.

[5] X. Chen, Z. Xu, H. Kim, P. Gratz, J. Hu, M. Kishinevsky, and U. Ogras, "In-Network Monitoring and Control Policy for DVFS of CMP Networks-on-Chip and Last Level Caches," *ACM Transactions on Design Automation of Electronic Systems*, vol. 18, no. 4, Oct. 2013.

[6] S. S. Clark, H. Mustafa, B. Ransford, J. Sorber, K. Fu, and W. Xu, "Current Events: Identifying Webpages by Tapping the Electrical Outlet," in *European Symposium on Research in Computer Security (ESORICS)*, 2013, pp. 700–717.

[7] Davidov, Mikhail and Oldenburg, Baron, "TEMPEST@Home - Finding Radio Frequency Side Channels," https://duo.com/labs/research/finding-radio-sidechannels#section9, 2020.

[8] Q. Deng, D. Meisner, L. Ramos, T. F. Wenisch, and R. Bianchini, "MemScale: Active Low-Power Modes for Main Memory," in *Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2011, pp. 225–238.

[9] V. Dhakal, A. M. Feit, P. O. Kristensson, and A. Oulasvirta, "Observations on Typing from 136 Million Keystrokes," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 2018, pp. 1–12.

[10] M. Enev, S. Gupta, T. Kohno, and S. N. Patel, "Televisions, Video Privacy, and Powerline Electromagnetic Interference," in *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS)*, 2011, pp. 537–550.

[11] Y. Gao, H. Zhang, W. Cheng, Y. Zhou, and Y. Cao, "Electro-Magnetic Analysis of GPU-Based AES Implementation," in *Proceedings of the 55th Annual Design Automation Conference (DAC)*, 2018.

[12] D. Genkin, L. Pachmanov, I. Pipman, and E. Tromer, "Stealing Keys from PCs Using a Radio: Cheap Electromagnetic Attacks on Windowed Exponentiation," in *International Conference on Cryptographic Hardware and Embedded Systems (CHES)*, 2015, pp. 207–228.

[13] ——, "ECDH Key-Extraction via Low-Bandwidth Electromagnetic Attacks on PCs," in *Topics in Cryptology - CT-RSA 2016*, 2016, pp. 219–235.

[14] 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)*, 2016, pp. 1626–1638.

[15] D. Genkin, M. Pattani, R. Schuster, and E. Tromer, "Synesthesia: Detecting Screen Content via Remote Acoustic Side Channels," in *Proceedings of the 2019 IEEE Symposium on Security and Privacy (S&P)*, 2019, pp. 853–869.

[16] 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 (USENIX Security 15)*, Aug. 2015, pp. 849–864.

[17] 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)*, 2017, pp. 1095–1108.

[18] K. B. Hardin, J. T. Fessler, and D. R. Bush, "Spread Spectrum Clock Generation for the Reduction of Radiated Emissions," in *Proceedings of IEEE Symposium on Electromagnetic Compatibility (EMC)*, 1994, pp. 227–231.

[19] S. Jana and V. Shmatikov, "Memento: Learning Secrets from Process Footprints," in *Proceedings of the 2012 IEEE Symposium on Security and Privacy (S&P)*, 2012, pp. 143–157.

[20] Z. H. Jiang, Y. Fei, and D. Kaeli, "A Complete Key Recovery Timing Attack on a GPU," in *2016 IEEE International symposium on high performance computer architecture (HPCA)*, 2016, pp. 394–405.

[21] ——, "Exploiting Bank Conflict-Based Side-Channel Timing Leakage of GPUs," *ACM Transactions on Architecture and Code Optimization*, vol. 16, no. 4, Nov. 2019.

[22] H. Kim, S. Lee, and J. Kim, "Inferring Browser Activity and Status through Remote Monitoring of Storage Usage," in *Proceedings of the 32nd Annual Conference on Computer Security Applications (ACSAC)*, 2016, pp. 410–421.

[23] W. Kim, M. S. Gupta, G.-Y. Wei, and D. Brooks, "System Level Analysis of Fast, Per-Core DVFS using On-Chip Switching Regulators," in *2008 IEEE 14th International Symposium on High Performance Computer Architecture (HPCA)*, 2008, pp. 123–134.

[24] M. Kosaka, "Inside Look at Modern Web Browser."

[25] M. G. Kuhn, "Electromagnetic Eavesdropping Risks of Flat-Panel Displays," in *Proceedings of the 4th International Conference on Privacy Enhancing Technologies (PETS)*, 2004, pp. 88–107.

[26] E. Ladakis, L. Koromilas, G. Vasiliadis, M. Polychronakis, and S. Ioannidis, "You Can Type, but You Can't Hide: A Stealthy GPU-based Keylogger," in *Proceedings of the 6th European Workshop on System Security (EuroSec)*, 2013.

[27] S. Lee, Y. Kim, J. Kim, and J. Kim, "Stealing Webpages Rendered on Your Browser by Exploiting GPU Vulnerabilities," in *Proceedings of the 2014 IEEE Symposium on Security and Privacy (S&P)*, 2014, pp. 19–33.

[28] M. Lipp, D. Gruss, M. Schwarz, D. Bidner, C. Maurice, and S. Mangard, "Practical Keystroke Timing Attacks in Sandboxed JavaScript," in *European Symposium on Research in Computer Security (ESORICS)*, 2017, pp. 191–209.

[29] C. Luo, Y. Fei, and D. Kaeli, "Side-Channel Timing Attack of RSA on a GPU," *ACM Transactions on Architecture and Code Optimization*, vol. 16, no. 3, Aug. 2019.

[30] C. Luo, Y. Fei, P. Luo, S. Mukherjee, and D. Kaeli, "Side-Channel Power Analysis of a GPU AES Implementation," in *Proceedings of the 2015 33rd IEEE International Conference on Computer Design (ICCD)*, 2015, pp. 281–288.

[31] K. Ma, X. Li, W. Chen, C. Zhang, and X. Wang, "GreenGPU: A Holistic Approach to Energy Efficiency in GPU-CPU Heterogeneous Architectures," in *Proceedings of the 2012 41st International Conference on Parallel Processing (ICPP)*, 2012, pp. 48–57.

[32] N. Matyunin, Y. Wang, T. Arul, K. Kullmann, J. Szefer, and S. Katzenbeisser, "MagneticSpy: Exploiting Magnetometer in Mobile Devices for Website and Application Fingerprinting," in *Proceedings of the 18th ACM Workshop on Privacy in the Electronic Society (WPES)*, 2019, pp. 135–149.

[33] C. Maurice, C. Neumann, O. Heen, and A. Francillon, "Confidentiality Issues on a GPU in a Virtualized Environment," in *International Conference on Financial Cryptography and Data Security (FC)*, 2014, pp. 119–135.

[34] X. Mei, L. S. Yung, K. Zhao, and X. Chu, "A Measurement Study of GPU DVFS on Energy Conservation," in *Proceedings of the Workshop on Power-Aware Computing and Systems (HotPower)*, 2013.

[35] Micron Technology, Inc., "GDDR5 SGRAM Introduction."

[36] ——, "GDDR6: The Next-Generation Graphics DRAM."

[37] A. K. Mishra, R. Das, S. Eachempati, R. Iyer, N. Vijaykrishnan, and C. R. Das, "A Case for Dynamic Frequency Tuning in On-Chip Networks," in *2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2009, pp. 292–303.

[38] A. Miyoshi, C. Lefurgy, E. Van Hensbergen, R. Rajamony, and R. Rajkumar, "Critical Power Slope: Understanding the Runtime Effects of Frequency Scaling," in *Proceedings of the 16th International Conference on Supercomputing (ICS)*, 2002, pp. 35–44.

[39] J. V. Monaco, "SoK: Keylogging Side Channels," in *Proceedings of the 2018 IEEE Symposium on Security and Privacy (S&P)*, 2018, pp. 211–228.

[40] H. Naghibijouybari, K. N. Khasawneh, and N. Abu-Ghazaleh, "Constructing and Characterizing Covert Channels on GPGPUs," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2017, pp. 354–366.

[41] 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)*, 2018, pp. 2139–2153.

[42] R. Nath and D. Tullsen, "The CRISP Performance Model for Dynamic Voltage and Frequency Scaling in a GPGPU," in *Proceedings of the 48th International Symposium on Microarchitecture (MICRO)*, 2015, pp. 281–293.

[43] A. Nazari, N. Sehatbakhsh, M. Alam, A. Zajic, and M. Prvulovic, "EDDIE: EM-Based Detection of Deviations in Program Execution," in *Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA)*, 2017, pp. 333–346.

[44] Newegg, "Best Selling Computer Cases," https://www.newegg.com/d/Best-Sellers/Computer-Cases/s/ID-7.

[45] 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)*, 2015, pp. 1406–1418.

[46] P. Perera and V. M. Patel, "Deep Transfer Learning for Multiple Class Novelty Detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[47] R. D. Pietro, F. Lombardi, and A. Villani, "CUDA Leaks: A Detailed Hack for CUDA and a (Partial) Fix," *ACM Transactions on Embedded Computing Systems*, vol. 15, no. 1, Jan. 2016.

[48] 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, no. 1, pp. 34–42, 2017.

[49] L. Ruff, R. A. Vandermeulen, N. Görnitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," in *Proceedings of the 35th International Conference on Machine Learning (ICML)*, vol. 80, 2018, pp. 4393–4402.

[50] 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)*, 2018, pp. 1–8.

[51] N. Sehatbakhsh, B. B. Yilmaz, A. Zajic, and M. Prvulovic, "A New Side-Channel Vulnerability on Modern Computers by Exploiting Electromagnetic Emanations from the Power Management Unit," in *2020 IEEE*

International Symposium on High Performance Computer Architecture (HPCA), 2020, pp. 123–138.
[52] C. Shen, T. Liu, J. Huang, and R. Tan, "When LoRa Meets EMR: Electromagnetic Covert Channels Can Be Super Resilient," in *Proceedings of the 2021 IEEE Symposium on Security and Privacy (S&P)*, 2021.
[53] A. Shusterman, L. Kang, Y. Haskal, Y. Meltser, P. Mittal, Y. Oren, and Y. Yarom, "Robust Website Fingerprinting Through the Cache Occupancy Channel," in *28th USENIX Security Symposium (USENIX Security 19)*, Aug. 2019, pp. 639–656.
[54] D. X. Song, D. Wagner, and X. Tian, "Timing Analysis of Keystrokes and Timing Attacks on SSH," in *USENIX Security Symposium (USENIX Security 01)*, 2001.
[55] O. Starov and N. Nikiforakis, "Xhound: Quantifying the fingerprintability of browser extensions," in *Proceedings of the 2017 IEEE Symposium on Security and Privacy (S&P)*, 2017, pp. 941–956.
[56] G. Vasiliadis, M. Polychronakis, and S. Ioannidis, "GPU-Assisted Malware," *International Journal of Information Security*, vol. 14, no. 3, pp. 289–297, Jun. 2015.
[57] P. Vila and B. Kopf, "Loophole: Timing Attacks on Shared Event Loops in Chrome," in *26th USENIX Security Symposium (USENIX Security 17)*, Aug. 2017, pp. 849–864.
[58] R. Wang, H. Wang, and E. Dubrova, "Far Field EM Side-Channel Attack on AES Using Deep Learning," in *Proceedings of the 4th ACM Workshop on Attacks and Solutions in Hardware Security (ASHES)*, 2020, pp. 35–44.
[59] Z. Wang, W. Yan, and T. Oates, "Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline," in *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 1578–1585.
[60] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for Reduced CPU Energy," in *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 1994.
[61] Z. Zhan, Z. Zhang, and X. Koutsoukos, "BitJabber: The World's Fastest Electromagnetic Covert Channel," in *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2020.
[62] K. Zhang and X. Wang, "Peeping Tom in the Neighborhood: Keystroke Eavesdropping on Multi-User Systems," in *USENIX Security Symposium (USENIX Security 09)*, 2009, pp. 17–32.
[63] Z. Zhang, S. Liang, F. Yao, and X. Gao, "Red Alert for Power Leakage: Exploiting Intel RAPL-Induced Side Channels," in *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security (ASIA CCS)*, 2021, pp. 162–175.
[64] Z. Zhang, Z. Zhan, D. Balasubramanian, B. Li, P. Volgyesi, and X. Kousoukos, "Leveraging EM Side-Channel Information to Detect Rowhammer Attacks," in *Proceedings of the 2020 IEEE Symposium on Security and Privacy (S&P)*, May 2020.
[65] Z. Zhou, W. Diao, X. Liu, Z. Li, K. Zhang, and R. Liu, "Vulnerable GPU Memory Management: Towards Recovering Raw Data from GPU," *Proceedings on Privacy Enhancing Technologies (PETS)*, vol. 2017, no. 2, pp. 57–73, 2017.
[66] Z. Zhu, S. Kim, Y. Rozhanski, Y. Hu, E. Witchel, and M. Silberstein, "Understanding The Security of Discrete GPUs," in *Proceedings of the General Purpose GPUs (GPGPU)*, 2017, pp. 1–11.

APPENDIX A

In this paper, we carried out all the experiments using an SDR device, USRP B210, and an ultra-wideband directional antenna, RFSPACE UWB-3. As shown in Figure 15, the antenna is directly connected to the SDR device via a coaxial cable without any other amplifier/filter front-end modules in-between. The bandwidth we need here is 25 MHz, and the USRP B210 can provide 56 MHz of instantaneous bandwidth in the frequency range of 70 MHz to 6 GHz, which is more than sufficient for our needs. The GNU Radio framework is used to capture and process the signal of interest.

The modern-looking computer case with a translucent side panel (which is acrylic) is AeroCool Cylon RGB Mid Tower. The monitor is HP VH240A. The motherboard installed in this case is ASUS PRIME Z270-P. The CPU used in this system

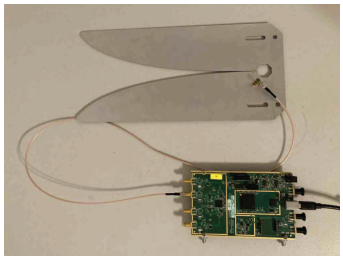

Fig. 15: Signal measurement equipment: USRP B210 and RFSPACE UWB-3

is Intel i5-6500T. The power supply unit (PSU) installed in this case is Apevia ATX-JP1000 Jupiter 1000W.

The all-metal computer case is Thermaltake Versa H22 Mid Tower. The monitor is HP VH240A. The motherboard installed in the case is ASRock Z270 Killer SLI. The CPU used in this system is Intel i3-6100. The PSU installed in this case is Thermaltake Smart 700W.

When we performed the experiments reported in Section IV-C, we used AMD GPU driver for Linux (AMDGPU 20.20) to set the performance level to the second lowest one. However, NVIDIA GPU driver does not allow us to fix the GPU to a specific performance level except for the highest one. To ensure the appearance of the signals of interest, we played YouTube videos to make sure that the performance level bumps up to the second lowest one frequently.

APPENDIX B

Here we use an example to demonstrate the effectiveness of using the techniques described in Section V to overcome the relatively low SNR issue. We use AMD RX 5700 and Linux OS in this example.

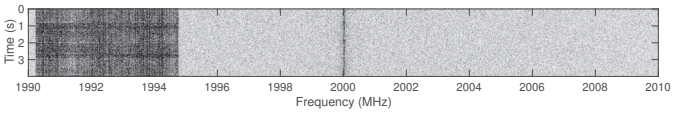

Fig. 16: Spectrogram corresponding to a scenario where the periodic performance level change is invisible

We have RX 5700 in the all-metal computer case, and place it far away from the antenna (about 6 meters). We make a script which uses the AMD GPU driver to periodically switch the performance level between the lowest and the second lowest. Figure 16 shows the corresponding spectrogram. On the spectrogram, we cannot visibly find any patterns of stripe appearance. If we simply use such spectrograms for attacks like website fingerprinting, it is very unlikely that the attacks can be successfully mounted.

By contrast, Figure 17 shows the time series derived using the techniques described in Section V. From the figure, we can clearly see the designed periodic peak appearance pattern.

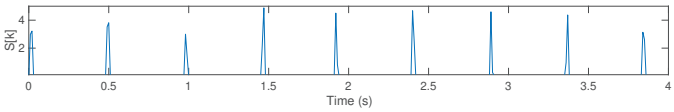

Fig. 17: Derived time series in which peaks appear periodically and match the desired pattern

APPENDIX C

Given a target GPU, there may be some other GPUs close to it, and they all have the same second lowest WCK frequency. In such a scenario, an attacker can first try to avoid picking up the disturbing signals emitted from the nearby GPUs by an appropriate placement of the directional antenna. We have experimentally verified this against the following setup – two RX 580 machines are placed close to each other (less than 1 meter apart), where one of them is the target that has a cursor blinking in the Google search box and the other one plays a Youtube video; and the directional antenna is placed 3 meters away from both machines. We confirm that we can find several appropriate placements to pick up only the signals of interest of the target, and the corresponding spectrogram is shown in Figure 18 (A).

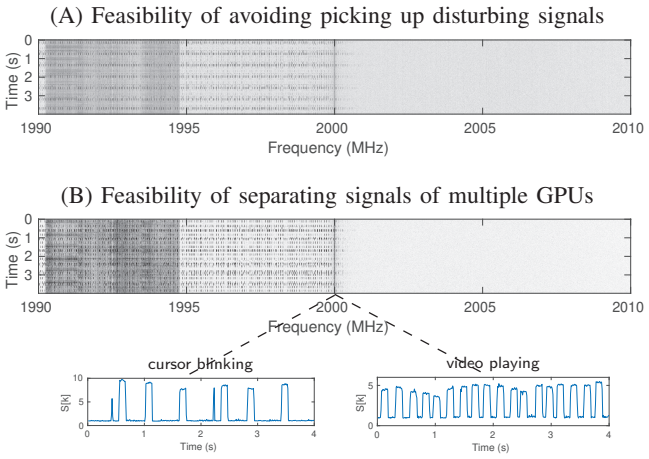

Fig. 18: Examples of dealing with disturbing signals generated by nearby GPUs

Nevertheless, the attacker may be restricted from arbitrarily positioning the antenna, and therefore the signals emitted from the nearby GPUs may be picked up clearly that can strongly contaminate the spectrum. Continuing with the example above, Figure 18 (B) shows the spectrogram where we have signals of both RX 580 GPUs picked up. Compared to Figure 18 (A), we can see that the desired pattern becomes hardly recognizable. Interestingly, we find that the time series derivation technique described in Section V-A can help us precisely separate signals generated by multiple identical or similar GPUs. The reason is that, due to oscillator imperfection, the f_c of a GPU is nearly unique and often differs from the f_c of another one by several kHz, namely multiple k^{th} DFT bins for the f_c can be found and subsequently multiple independent $S[k]$ traces will be derived. In our example, the two f_c's are separated by 12.6 kHz (i.e., 126 bins given 100 Hz resolution) and the signals can be fully separated as shown by the bottom part of Figure 18 (B).

APPENDIX D

For classifying time series data in our website fingerprinting attack, we directly use the residual network (ResNet) model proposed in [59]. Its architecture is duplicated here in Figure 19. The details about the model can be found in [59] and its code repository (https://github.com/cauchyturing/UCR_Time_Series_Classification_Deep_Learning_Baseline).

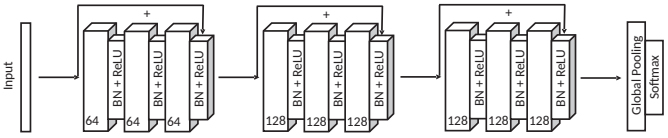

Fig. 19: The neural network model used for our website fingerprinting attack (duplicated from [59])

APPENDIX E

Table VIII lists the websites that are used in our evaluations on website fingerprinting.

TABLE VIII: List of fingerprinted websites

9gag.com	abs-cbn.com	adobe.com
amazon.com	amazonaws.com	aol.com
apple.com	archive.org	ask.com
battle.net	bing.com	blogger.com
booking.com	businessinsider.com	cnn.com
deviantart.com	dictionary.com	discord.com
duckduckgo.com	ebay.com	espncricinfo.com
exoclick.com	facebook.com	feedly.com
foxnews.com	gamepedia.com	github.com
go.com	goodreads.com	google.com
imdb.com	linkedin.com	live.com
microsoft.com	msn.com	netflix.com
office.com	paypal.com	pinterest.com
reddit.com	roblox.com	stackoverflow.com
twitch.tv	twitter.com	whatsapp.com
wikipedia.org	yahoo.com	youtube.com
zillow.com	zoom.us	

APPENDIX F

Figure 20 shows the classification results in the form of heat-map-represented confusion matrices corresponding to the nearby scenario evaluation performed in Section VI-C, and Figure 21 shows the confusion matrices corresponding to the evaluation in the first faraway scenario.

APPENDIX G

We relax the limitations in the nearby scenario setting by allowing the training data set to include examples collected at all the other seven spots when testing each of the last four spots (i.e., spots N5, N6, N7, and N8). Table IX shows the evaluation results. Compared with the results in Table V, we can observe that the accuracy is increased by more than 10% in

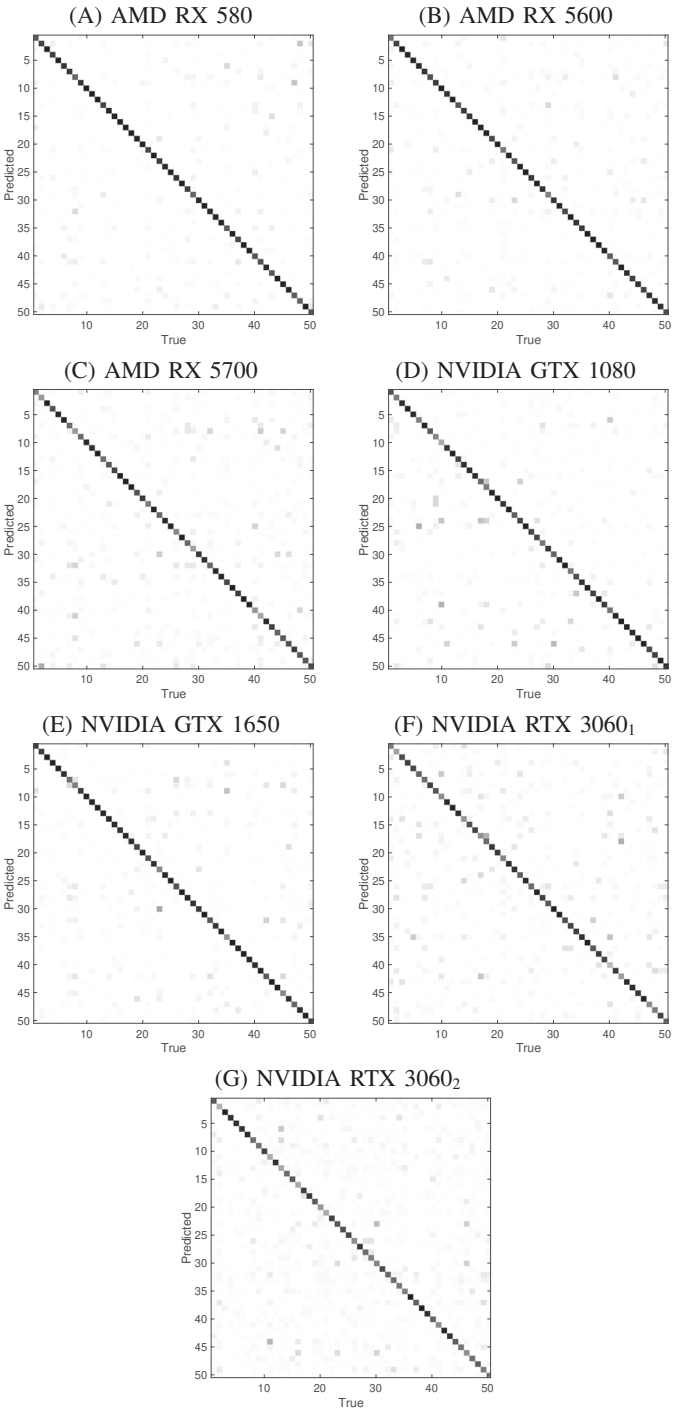

Fig. 20: Confusion matrices corresponding to the evaluation reported in Table V

many cases (e.g., in terms of NVIDIA GTX 1080, the accuracy is increased by 11.1%, and in terms of NVIDIA RTX 3060, the accuracy is increased by 11.6% in the Linux case and 13.3% in the Windows case). Similarly, Table X shows the evaluation results when we relax the limitations in the first faraway scenario. Since an attacker can freely choose different spots for profiling at his/her own place, the accuracies reported here actually represent more pragmatic results.

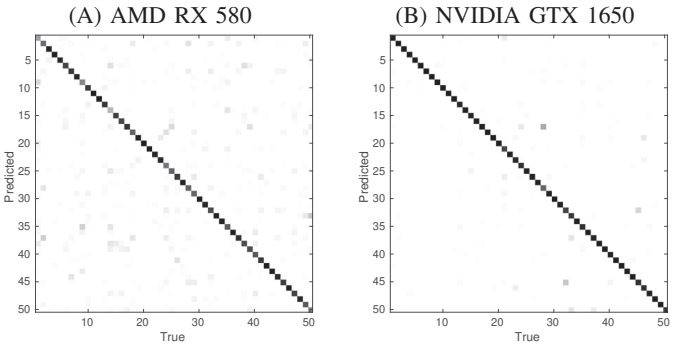

Fig. 21: Confusion matrices corresponding to the evaluation reported in "Faraway Scenario 1" of Table VI

TABLE IX: Fingerprinting accuracy in the relaxed nearby scenario (training examples are collected at all the other spots)

	RX 580	RX 5600	RX 5700	GTX 1080	GTX 1650	RTX 3060$_1$	RTX 3060$_2$
Avg.	93.0%	86.9%	80.6%	90.1%	89.9%	82.7%	77.2%
Std.	1.4%	0.7%	1.6%	3.1%	1.9%	3.9%	4.1%

TABLE X: Fingerprinting accuracy in the relaxed faraway scenario (training examples are collected at all the other spots)

	RX 580	GTX 1650
Avg.	83.4%	95.7%
Std.	2.7%	1.1%

APPENDIX H

A counterintuitive phenomenon in the website fingerprinting evaluation is that the accuracy for the NVIDIA GTX 1650 is higher at far distances than it is at near distances. One possible reason for this phenomenon is that some other EM signal in the frequency range of our interest is generated from a certain hardware component on the MSI GTX 1650 GPU card, which does not propagate far but is still relatively strong at 1 m. In other words, this additional signal is more disturbing to the EM signal of interest at 1 m. To simply verify this, we show two normalized $S[k]$ traces in Figure 22 derived from signals captured at 1 m and 3 m, corresponding to a 2Hz performance level switching. From the figure, we can see that the $S[k]$ trace at 1 m indeed has many relatively high spikes in between two expected spikes. However, due to the high complexity of the hardware stack of NVIDIA GPUs, we cannot easily pinpoint the exact component that contributes to this.

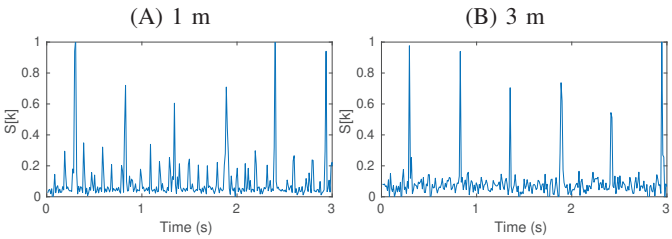

Fig. 22: Normalized $S[k]$ traces derived from the EM signals of interest measured at 1 m and 3 m against MSI GTX 1650