PWRLEAK: Exploiting Power Reporting Interface for Side-channel Attacks on AMD SEV

Wubing Wang[1], Mengyuan Li[1], Yinqian Zhang[2], and Zhiqiang Lin[1]

[1] The Ohio State University, Columbus OH 43210, USA
wang.11488, li.5733, lin.3021@osu.edu
[2] Southern University of Science and Technology, Shenzhen, Guangdong, 518055, China
yinqianz@acm.org

Abstract. An increasing number of Trusted Execution Environment (TEE) is adopting to a variety of commercial products for protecting data security on the cloud. However, TEEs are still exposed to various side-channel vulnerabilities, such as execution order-based, timing-based, and power-based vulnerabilities. While recent hardware is applying various techniques to mitigate order-based and timing-based side-channel vulnerabilities, power-based side-channel attacks remain a concern of hardware security, especially for the confidential computing settings where the server machines are beyond the control of cloud users. In this paper, we present PWRLEAK, an attack framework that exploits AMD's power reporting interfaces to build power side-channel attacks against AMD Secure Encrypted Virtualization (SEV)-protected VM. We design and implement the attack framework with three general steps: (1) identify the instruction running inside AMD SEV, (2) apply a power interpolator to amplify power consumption, including an emulation-based interpolator for analyzing purposes and a more general interrupt-based interpolator, and (3) infer secrets with various analysis approaches. A case study of using the emulation-based interpolator to infer the whole JPEG images processed by libjpeg demonstrates its ability to help analyze power consumption inside SEV VM. Our end-to-end attacks against Intel's Integrated Performance Primitives (Intel IPP) library indicates that PWRLEAK can be exploited to infer RSA private keys with over 80% accuracy using the interrupt-based interpolator.

1 Introduction

Private data is becoming an important asset for our society and personal life. More and more data-driven applications and technologies are introduced to release the value of data to the greatest extent, which, however, can potentially put personal or technical data at risk of leakage. Such concerns are especially perceptible in the cloud computing domain, where numerous cloud tenants rent cloud services from cloud service providers, upload and process sensitive data on the cloud. Potential safety hazards can occur in two ways: (1) malicious cloud tenants steal data from other cloud tenants, and (2) a curious or malicious cloud service provider monitors or steals data from its cloud tenants directly. Therefore, there is an urgent demand for the industries to come up with techniques that can guarantee data security in the cloud computing environment.

Trusted Execution Environment (TEE) is one of those techniques that meets the above requirement and has already been adopted in mainstream cloud service providers, such as Google cloud [17], Microsoft Azure [37], and Amazon AWS [1]. With the help of trustworthy hardware (*e.g.*, CPU and memory encryption engine), TEE provides hardware-guaranteed isolation to protect cloud user's data from other cloud users or even the cloud service provider. When TEEs are enabled, even the highest privilege software (*e.g.* operating system, hypervisor, *etc.*) cannot directly access cloud user's data. With such promising security guarantee, CPU vendors, such as Intel, AMD or ARM, all released server-level processors that support TEE features to protect VMs running on the cloud, including the current available AMD Secure Encrypted Virtualization (SEV) [2], and the upcoming Intel Trust Domain Extensions (TDX) [4] and ARM Confidential Computing Architecture (ARM CCA) [3].

However, recent research [27,24,31] showed that different types of side-channel attacks could be exploited to steal TEE-protected secrets. Among different types of side-channel attacks, the power side-channel attack plays a very important role, where the untrusted cloud service provider can easily collect the power consumption of TEE and steal secrets. Platypus [31] first examines the threat of software-based power-based side-channel attacks in cloud-based TEEs. Using power consumption reporting interfaces, Platypus successfully demonstrates that attackers can obtain power consumption with instruction-level granularity through APIC interrupts [45], and can use fine-grained power data to carry out a series of end-to-end attacks, including breaking KASLR and breaking constant-time cryptographic implementations (AES-NI) used by Intel SGX. The feasibility of power-based side-channel attacks in SEV was also discussed in [31].

Inspired by previous work, in this paper, we aim to explore the power-based side-channel attacks in the AMD SEV environment. We introduce PWRLEAK, a software-based power side-channel framework that analyzes power consumption in AMD SEV-only VMs (excluding the newer SEV-ES [21] and SEV-SNP [6] versions). Specifically, PWRLEAK uses the AMD power-reporting features to monitor the program execution inside AMD SEV VMs, and then infers secrets from the VMs. We first analyze the possibility of using power information to distinguish instructions in SEV. We show that different instructions have different power consumption, and the same instruction with various operands also has distinguishable power differences. Based on such observation, PWRLEAK makes use of page-table-based controlled channel [24] to intercept VM's execution in real-time and then infers the secret inside the SEV VM by inferring executed instructions and their operands based on distinguishable power consumption.

We test two interpolators that could amplify and produce distinguishable power consumption of a single instruction on the AMD platform: an emulation-based interpolator and an interrupt-based interpolator. The emulation-based interpolator can acquire higher-resolution power information by emulating the execution of instructions, which acts as an ideal tool to compare power consumption for analysis purposes. The more general interrupt-based interpolator ports an existing APIC-based amplifier introduced by Platypus [31] to AMD platform and can amplify the power information with interrupts by forcing the re-execution of instructions. To demonstrate the capability of PWRLEAK, we showed that the emulation-based interpolator can be used to analyze power consumption inside SEV VM and recover JPEG images processed by libjpeg

library. We further demonstrated that PWRLEAK could steal RSA private keys from the Intel IPP library with over 80% accuracy using the interrupt-based interpolator. To the best of our knowledge, PWRLEAK is the first power-based side-channel attack that extracts secrets from AMD SEV-protected VMs. The prototype of PWRLEAK has been made public available at github.com/OSUSecLab/PWRLEAK. The contributions of the paper are summarized as follows:

- **An instruction-level power consumption study inside SEV VM.** We measure the power-based information leakage towards AMD SEV VMs, and figure out that the power information can be used to differentiate instructions and their operands running inside AMD SEV VMs.
- **Test power interpolators on AMD SEV.** We test two power interpolators that take advantage of the instruction emulation function or the advanced programmable interrupt controller (APIC) to amplify the power consumption of a single instruction. We show that these two interpolators are useful to amplify and analyze the energy consumption of executed instructions in SEV VMs.
- **A new attack on AMD SEV VM.** We also propose PWRLEAK, a new power attack framework on AMD SEV VMs, and successfully steal secrets from a VM protected by the baseline SEV version. The feasibility and limitations of similar attacks but in newer versions of SEV (SEV-ES and SEV-SNP) and the corresponding countermeasures are also discussed in the paper.

Responsible disclosure. We disclosed the proposed findings and attacks to AMD in April 2023. At the time of writing, AMD has acknowledged our findings and provided a tracking ID for future communications. However, as discussed in Sec. 6, neither attack method presented in this paper could be directly conducted against the newer versions of SEV (*e.g.*, SEV-ES and SEV-SNP). The emulation-based interpolator acts as an analysis tool and is not expected to work for SEV-ES or SEV-SNP. For the interrupt-based interpolator, our paper makes use of it to demonstrate that power-based side-channel attacks can work in SEV, but we did not conduct relevant experiments in SEV-ES or SEV-SNP. We can foresee that there may be a lot of additional noise caused by additional protections enabled by SEV-ES and SEV-SNP, such as register encryption or ownership check, which prevents PWRLEAK from working directly. Therefore, VM protected by SEV-ES or SEV-SNP will not be affected by PWRLEAK.

2 Background

2.1 AMD Secure Encrypted Virtualization (SEV)

AMD first introduced Secure Encrypted Virtualization (SEV) in 2016 [2], which is a hardware-based technology designed to protect virtual machines (VMs) against both privileged software attackers and physical attackers on a remote platform. To protect the confidentiality of guest VMs' code and data, SEV provides necessary isolations for data (*e.g.*, cache and TLB) within the CPU chip, and encrypts VM's memory using memory encryption [49]. AMD later introduced SEV-ES (Encrypted State, the second generation of SEV [21]) to add additional protection towards VM's unencrypted register

states during VM-hypervisor world switch. Lately, in order to add additional memory integrity protection and defend against several controlled-channel attacks (page table manipulation attacks [48]), AMD introduced the third generation of SEV on Zen 3 architecture, called SEV Secure Nested Paging (SEV-SNP [6]). Due to the strong security guarantee and user-friendly mode provided, AMD SEV has already been adopted by some public cloud service providers, including Google Cloud [17] and Microsoft Azure [37].

2.2 Hardware Power Reporting Feature

The hardware power reporting interfaces provided by commodity processor vendors, such as Intel and AMD, allow software to monitor and control CPU's power consumption. The reporting interfaces related to power consumption specified in the AMD manual [53] include each CPU core's effective frequency and power consumption:

The Effective Frequency, which monitors the real CPU frequency of each core with the Max Performance Frequency Clock Count (MPERF) and Actual Performance Frequency Clock Count (APERF) `MSR` registers. These two registers can be accessed in kernel mode using `rdmsr` and `wrmsr` instructions. Users can calculate the effective frequency of a core over a software-determined window of time.

Processor Core Power Consumption, which provides power consumption for a given core over a software-determined time interval in `MSR_CORE_ENERGY_STAT` MSR. The value of the register is the cumulative energy consumption of a given CPU core. The sampling interval of `MSR_CORE_ENERGY_STAT` is 1 *ms*. Compared to the power interfaces in Intel Processors, which have a 50 μs sampling interval, AMD processors sample power consumption in a much coarser granularity.

2.3 Power-based Side-channel Attacks

Power-based side-channel attacks exploit the collected power information to distinguish victim's behaviors or infer secret from the victim. The power-based side-channel attacks can be further classified as *hardware-based power attacks* and *software-based power attacks*.

Hardware-based Power Side-channel Attacks. The *hardware-based power attacks* can usually acquire power consumption data with a higher granularity using an independent device. Existing attacks showed that power consumption data collected in this way could help an attacker identify the executed instruction [46,43,42] or infer the execution trace of programs [16]. Lately, researchers showed that hardware-based power side-channel attacks could successfully steal the RSA private key algorithm [20,54] or AES private keys [38,13,41].

Software-based Power Side-channel attacks. Software-based power side-channel attacks rely on software-based power reporting interfaces to collect power consumption data. There are many efforts to explore software-based power side channels in smartphones [12,36], which could fingerprint application being used or identify user's movement. Recent work in the past few years has used software-based power side channels

to break isolation protections provided by modern desktop or server processors. The two most relevant papers related to this article include Platypus [31], which focuses on attacking Intel SGX, and another software-based power side-channel attack [30], which focuses on AMD CPUs. On Intel CPUs, Platypus attacks [31] showed for the first time that attackers could distinguish different executed instructions and their operands by collecting power consumption information from the Intel Running Average Power Limit (RAPL) interface. With the help of APIC-timer interrupt, the attacker could get execution control with instruction-level granularity. These side-channel information could later be used to leak secret keys from the constant-time AES-NI implementation used by Intel SGX, break KASLR, and establish a time-independent convert channel. The power consumption of different instructions, but in AMD 's Zen microarchitecture, and the feasibility of power-based side-channel attacks in SEV was also studied or discussed in the paper. Lipp *et al.* [30] later demonstrated the danger of power-based side-channel attacks in modern AMD processors through several end-to-end attacks, which successfully broke KASLR, stole kernel secrets and established a covert channel from unprivileged attackers. In their attacks, they combined power consumption with `prefetch` to infer system states and steal secrets. Inspired by those existing papers, in our paper, we focus on AMD's Zen microarchitecture and explore the feasibility of stealing secrets from AMD SEV-protected VMs using software-based power side-channel attacks.

2.4 Common Power Analysis Methods

Simple Power Analysis method and *Cross Correlation Analysis* method are two methods widely used in power-based side channel attacks. Simple Power Analysis (SPA) is a technique that differentiates various operations by distinguishing individual power patterns [52]. Based on the method used to recognize and distinguish power patterns, the SPA attack can be further categorized into two types: the visual SPA attack [33] and the template-based SPA attack [11]. The visual SPA attack manually inspects and recognizes the difference in the power traces, and the template-based SPA attack uses the extracted mathematical statistic template to analyze the power traces. Cross Correlation Analysis (CCA) [34] uses the correlation coefficient to measure power traces to differentiate two inputs. Specifically, if two inputs are similar, the correlation coefficient value is high; otherwise, the correlation coefficient value is low.

Power consumption sampling with instruction-level granularity. To achieve a sampling rate with instruction-level granularity, previous attacks [31,27] usually utilized APIC timer interrupts to allow the target to execute a single or multiple instructions before being halted. On Intel platforms, SGX-STEP [45] first showed that an attacker could use APIC timer interrupts to execute zero-step or single-step SGX enclaves with instruction-level granularity. Platypus [31] then first combined this timer interrupt-based technique together with the power consumption interface to reveal the relationship between power consumption and different instructions. A similar methodology was also studied on the AMD platform to monitor the states of SEV VMs. CipherLeaks attack [27] used the APIC timer interrupt to step AMD SEV VM's execution inside an instruction page. In this paper, we collect the power consumption at the instruction level by adopting the same APIC-based sampling method presented in Platypus [31].

3 Exploring Power Consumption Leakage

This section consists of several experiments that aim to exploit the ability of hardware power reporting interfaces in AMD platform and collect the ground truth of the relationship between the power consumption with behaviors of a SEV-protected VM. All the experiments were conducted on a blade server with an 8-Core AMD EPYC 7251 Processor. The host OS runs Ubuntu 64-bit 18.04 with kernel version 4.20.0. The guest VMs run the same kernel version and are configured with 4 virtual CPUs, 4 GB memory, and 30 GB local disk as SEV official GitHub repository suggested [8].

3.1 Synchronous Power Measurement

To accurately measure power consumption, we first introduce a synchronous power measurement method to collect the ground truth and explore the relationship between instructions and the corresponding power consumption. More specifically, we first modified the `CPUID` handler in the KVM to act as an indicator of the start and end of a power trace, so that we could accurately locate the measured behaviors inside the VM. Then, for each instruction to be tested, we used two `CPUID` instructions to indicate the start and end points. We executed each instruction $100,000$ times inside the VM, and measured the overall power consumption on the CPU core. By dividing the total power consumption by $100,000$, we could know the power consumption of that instruction. Although the power consumption of `CPUID` is also included in the results, it is negligible compared to the power consumption caused by $100,000$ repeated instructions.

3.2 Instruction Power Consumption

Our experiment results showed that even under the protection of AMD SEV, different instructions, different operands, and different loaded data all produce distinguishable power consumption-based side-channel information.

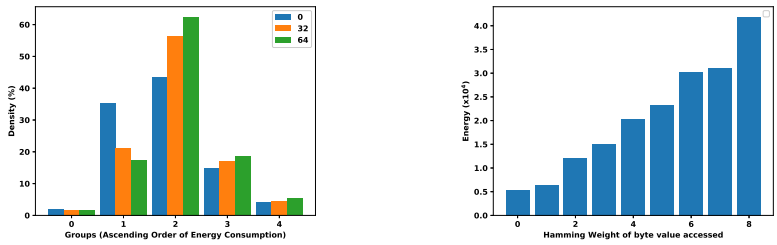
Instruction	Core Power ($10^4 mW$)	Instruction	Core Power ($10^4 mW$)
aesdec xmm1, xmm2	4.266	inc r64	1.488
aesdeclast xmm1, xmm2	4.166	mov mem, r64	1.700
aesenc xmm1, xmm2	5.340	mov r64, r64	0.387
aesenclast xmm1, xmm2	5.251	clflush mem	69.600
aesimc xmm1, xmm2	1.123	xor r64, r64	1.444
pclmullqlqdq xmm1, xmm2	7.150	fscale	40.599
dec r64	1.511	nop	0.554
imul r64, r64	4.633	rdrand r64	29.540

Table 1: Energy consumption of instructions.

Distinguishing Instructions. With the synchronous power measurement approach, we measure the power consumption of the instructions $100,000$ times and calculate the median power consumption of each instruction. We choose instructions that are commonly used and related to cryptography for demonstration. The data are reported in Table 1. There are four columns in this table; the first and the third column are the instructions been tested, and the second and the fourth column are the core power consumption for the given instruction. For instance, running instruction `aesdec` $100,000$ times consumes 4.266 x 10^4 milliwatts. The differences in power consumption among

the various instructions are noticeable. For instance, the `aesdec` instruction between two registers has 4.26 x 10^4 core power consumption, while the `aesenc` instruction between two registers has 5.34 x 10^4 core power consumption.

Distinguishing Operands. The power information can also be used to differentiate operands of the same instruction. Furthermore, we collected and explored the power difference caused by different operands of the same instruction. Our evaluation suggests that the exact value of the operands is hard to be distinguished. However, operands with different Hamming weights can be differentiated. We use the `imul` instruction to demonstrate this result. For the operand (64-bit), we selected operands with Hamming weights of 0, 32, and 64 bits. We measured each operand $100,000$ times and got the maximum and minimum value of energy consumption. After dividing the energy interval into five even groups, we counted the number of results for each interval. As shown in Fig. 1(a), the x-axis is the group number, with the energy consumption of each group sorted by ascending order; the y-axis is the percentage of the result that each group contains. In summary, the energy difference in the operand with various Hamming weights is observable; the operand with a lower Hamming weight has relatively less energy consumption.



(a) `imul` (b) `movb`

Fig. 1: Power difference because of operand's hamming weights.

Distinguishing Loaded Data. Similarly, we measured the relationship between the data loaded from the cache and their power consumption. We used the `movb` instruction to demonstrate this experiment, which could read one byte of data from the cache line. We generated 256 different pieces of data, which cover all possible values for a byte, and categorized them into nine different groups based on the Hamming weight (*e.g.* 0, 1, 2, ..., 8). After measuring the power consumption that loads every data $100,000$ times, we calculated the average power consumption of the data in each of the nine groups and presented the result in Fig. 1(b). The x-axis is the Hamming weight of each group, and the y-axis is the average power consumption of each group. The results suggest that loading the data with larger Hamming weights could consume more energy, which is distinguishable.

4 PwrLeak Design

In this section, we present PwrLeak, an attack framework to differentiate power consumption caused by instructions running inside SEV-protected VMs, and steal secrets from the victim VM.

4.1 Threat Model

In this paper, we consider the same threat model as AMD SEV's threat model [2], where the adversary is a privileged software attacker who does not know the data protected by the SEV-enabled guest VM, and cannot control any program running inside the guest VM. We further assume that the adversary has the pre-knowledge of the target program's binary running inside the VM (*e.g.*, a specific cryptography library), including detailed information such as control flow and function calls of the target program.

4.2 Overview of PWRLEAK

Even though our synchronous power measurement (discussed in Sec. 3) suggests that different behaviors inside SEV VMs lead to different power consumption. There are two main challenges left for a real world power-based side-channel attack. First, the synchronous power measurement approach used to measure repeated instructions is not practical. Second, the low power consumption sampling rate (1 ms) in AMD processors also limits the practicality of a real attack. To overcome these two challenges, we introduce PWRLEAK, whose general components are shown in Figure 2. *Instruction Identification* is a component used to locate some target instructions in a specific program running inside VMs. *Power Interpolator* is a component that can amplify the power consumption of target instructions, so that the amplified power data is sufficient for PWRLEAK to distinguish different instructions via AMD's coarse-grain hardware power reporting interfaces. *Power Attack* is a component that can run offline, possibly on a separate machine, and infers secrets by analyzing the collected power data.

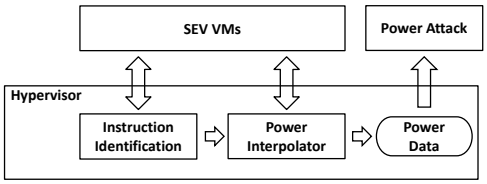


Fig. 2: PWRLEAK Overview.

Instruction Identification. To locate a specific instruction of the program running inside AMD SEV VMs, we use both the page-level memory access pattern [51] and the APIC single-step [45]. We first locate the page of the target instruction with the page-level memory access pattern, then use APIC single-step to further locate the target instruction inside this page. Need to note that the attacker can directly check the `rip` register to get the number of executed instructions. For a newer version such as SEV-ES or SEV-SNP, the attacker may need to check the ciphertext of the `rip` register to distinguish a single-step from a zero-step.

Power Interpolator. After locating the instruction, we measure the power consumption of these instructions by monitoring the core energy consumption MSR register (`MSR_CORE_ENERGY_STAT`). As the register is updated in a related low rate (*e.g.*, $1ms$), it is hard to measure the power consumption of a small gadget of instructions

(*i.e.* one instruction). To solve this problem, in Sec. 4.4, we test two power interpolators that can be used to amplify the power consumption of a single instruction.

Power Attack. Finally, the attackers conduct a power attack by analyzing the power consumption. As we assume that the attacker has knowledge about the program binary to be attacked, the secret about this program can be inferred by distinguishing different operations in critical locations.

4.3 Instruction Identification

To perform the attack, the adversary first needs to pinpoint a specific location (*e.g.*, an instruction inside its instruction page) in the program. In this work, we use the page sequence matching to pinpoint an instruction page, and use the SEV VM single-step to step to an instruction.

Page Sequence Matching. Previous works [51] have proved that the page sequence caused by page faults can be used by the adversary to successfully locate a certain instruction page of a target program. Similarly, in the SEV environment, the adversary collects the page fault sequence for the known target binary and uses this sequence to identify the target instruction page.

To trigger and monitor the page-level access pattern, we first clear the *present* bit in the page table entry for all pages mapped to the VM. Then we monitor the `VMEXIT` event (*e.g.*, the `handle_exit` function in the kernel). When the `VMEXIT` is triggered by a page fault, we collect the corresponding page address. Finally, PWRLEAK identifies the specific instruction page using the pre-collected page access pattern.

Unlike the traditional page table walk, SEV adopts a nested page table to maintain the address transmission between the guest virtual address (GVA) and the host physical address (HPA). Specifically, the nested page table consists of a guest page table and a host page table. The guest page table maintains the mapping between GVA and the guest physical address (GPA), and is within the protection of SEV. The host page table maintains the mapping between GPA and HPA and is under the hypervisor's control. Thus, the privileged adversary knows the mapping between GPA and HPA, but will not directly know the relationship between GVA and GPA. Therefore, instead of directly using the GPA to construct the page access pattern, PWRLEAK uses the address interval between two consecutive pages. For instance, we let p_i be the GPA of the i_{th} page access. Then, the page access pattern can be presented as the following: $S = \{p_1 - p_0, p_2 - p_1, \cdots, p_i - p_{i-1}, \cdots\}$.

SEV VM Single-step. To further improve the attack, it is necessary to narrow down the granularity of the attack to several instructions. PWRLEAK uses the APIC-based single-step to identify the instruction in which we are interested. Similarly to the method introduced in Platypus [31], SGX-Step [45] and Cipherleaks [27], we use the APIC interrupt to force the VM to `VMEXIT` after a single instruction. By carefully setting the APIC interrupt's timer, the program running inside the SEV VM can be single-stepped.

In particular, as shown in Figure 3, the first APIC interrupt arrives when the SEV VM is executing instruction 0, which raises a `VMEXIT` after instruction 0 is retired. After the `VMEXIT` is handled properly by the hypervisor, the hypervisor will execute the `VMRUN` instruction. After all guest states are restored, the next instruction 1 in the

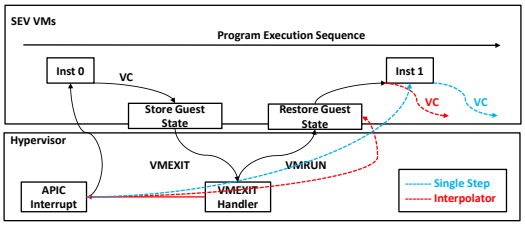


Fig. 3: APIC for the single-step and interpolator.

guest VM will be executed and the instruction pointer will be advanced. Meanwhile, the attacker can set the APIC timer in the `VMEXIT` handler, so that the next APIC interrupt arrives when instruction 1 is executing (shown in blue lines in Figure 3). Because of the second interrupt, another exception will be raised after instruction 1 is retired, which forces another `VMEXIT` and is trapped by attacker-controlled `VMEXIT` handler. In this way, the attacker can single-step the SEV VM.

In SEV-ES and SEV-SNP, the instruction pointer to be interrupted is encrypted and stored in an area called the VM Save Area (VMSA) during `VMEXIT`. Thus, the attacker cannot directly trace the execution of instructions. However, the attacker could still know whether the instruction pointer is advanced by monitoring the change of the encrypted instruction pointer inside VMSA. With the knowledge about the program context, the attacker knows the distance to the instruction in which the attacker is interested, therefore, allows the attacker to single-step the target instruction.

4.4 Power Interpolator

Limited by AMD's coarse-grained power consumption interface, a power interpolator is introduced to amplify the power consumption of the target instruction. PWRLEAK tests two interpolators: emulation-based and the existing interrupt-based interpolators. The emulation-based interpolator can be used to emulate a single instruction multiply times to amplify its power consumption, which is used by as an analysis tool to collect power consumption from a SEV VM. It's important to note that the emulation-based interpolator is only compatible with SEV VM and cannot be used with SEV-ES or SEV-SNP VM whose register states are encrypted during VMEXIT. The interrupt-based interpolator is a more general method that could be potentially applied to SEV-ES and SEV-SNP but with lots of noise. More discussion of their pros and cons, as well as their applicability in SEV-ES and SEV-SNP is covered in Sec. 6.

Emulation-based Interpolator. Emulation-based interpolator modifies the kernel instruction emulation function to amplify the power consumption. KVM emulates the execution of the instruction that raises the exception in its handler using an emulation function to ensure that the same exception doesn't raise right after `VMRUN`. This function (`x86_emulate_instruction`) emulates the execution behavior of the instruction, for instance, accessing a specific memory location.

To deploy the emulation-based interpolator, PWRLEAK implements our own instruction emulation function by extending the `x86_emulate_instruction` function. In its original implementation, only instructions that access memory are emulated. We further extend the capability of the emulation function by emulating other instructions. Particularly, for those instructions that do not have memory access, we obtain the

corresponding system states (*e.g.*, the value of registers) from the virtual machine control block (VMCB), and retrieve the instruction to be emulated using single-step and the program context . Finally, we execute the instruction directly in the hypervisor.

To apply this emulation-based interpolator for power analysis, PWRLEAK first hooks the `VMEXIT` handler. When the target instruction is interrupted, the `VMEXIT` trampoline handler is called. PWRLEAK forces the instruction to be emulated multiple times in the trampoline handler, surrounded by the instruction that reads the power consumption. The emulation-based interpolator can be used as a tool to analyze power-based vulnerabilities by precisely collecting the power consumption of specific instructions. Even the emulation-based approach works well in SEV, it cannot be recognized as an effective attack method because an attacker can directly read the values of registers in SEV, and such information leakage will cause more severe leaks. Meanwhile, this approach is limited by SEV-ES and SEV-SNP. SEV-ES and SEV-SNP would encrypt the Virtual Machine Save Area (VMSA), which stores all VM's state-related data. It would prevent the emulation-based interpolator from obtaining the register values. Therefore, the emulation-based interpolator cannot work with in the machine that supports SEV-ES and SEV-SNP.

Interrupt-based Interpolator. The interrupt-based interpolator is considered to be a more general approach. In this paper, we have shown that it is feasible to use this approach in the baseline SEV, and similar approaches may also potentially work with SEV-ES or SEV-SNP. Specifically, by setting a value to the APIC timer, the attacker can control where the APIC interrupt arrives. The single-step makes the interrupt arrive when executing the first instruction after the `VMRUN`. Similar to the single-step, by setting a small APIC interval, the interrupt-based interpolator makes the interrupt arrive at the SEV VM within the `VMRUN`; thus, the exception will be raised before the next guest instruction has been executed, and the instruction pointer of the guest VM will not be advanced (shown in red lines in Figure 3). In our experiment setup, the attacker conservatively underestimates the APIC interval and can directly verify whether the instruction is zero-stepped by checking the unencrypted `rip` register inside the SEV VM's VM control block. While this paper did not test for it, attackers may still be able to determine whether a zero-step or single-step occurred in SEV-ES or SEV-SNP environment through other side-channel information, such as observing changes in the ciphertext of the `rip` register, or by monitoring performance counters. With the interrupt-based interpolator approach, the attacker can force the `VMRUN` or target instruction in the VM to be executed multiple times for measurement purposes [31]. However, the root reason of distinguishable power consumption of an instruction amplified by interrupt-based interpolator is not verified by the paper. The power consumption difference could be introduced by different hamming weight in VMCB or be introduced by transient execution of the next instruction.

4.5 Power Attack

After amplifying and measuring the power consumption of the instructions with the power interpolator, PWRLEAK uses the power information to infer the secret. For operations with noticeable differences in power consumption, PWRLEAK uses the Simple

Power Analysis (SPA) attack to infer the secret. PWRLEAK uses the cross correlation analysis (CCA) to infer the secret in those applications for which SPA fails.

5 Evaluation

In this section, we evaluate PWRLEAK using two case studies. We first present how to use only the emulation-based interpolator to analyze the power leakage from libjpeg, and then demonstrate the attack targeting an RSA implementation with the interrupt-based interpolator. The evaluation settings are identical as the experiment settings in Sec. 3, excepting the attacker now doesn't control the SEV-protected VM.

5.1 Infer Images from libjpeg

Libjpeg is a widely used image-rendering library that offers lossy image compression and decompression implementations. The input of the libjpeg library is a bitmap image. The decoding of a JPEG image transfer a bitmap image into blocks with 8x8 pixels with three steps: *decompression*, *dequantization*, and *inverse discrete cosine transformation* (IDCT). The encoding procedure transfers blocks to a bitmap image with *discrete cosine transform*, *quantization*, and *compression*. The JPEG image is shown on the screen based on the decoded pixels. With enough information about each 8x8 pixels, adversaries can recover the whole JPEG image.

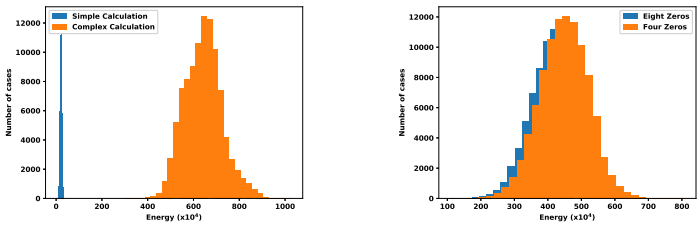
```
1  int jpeg_idct_islow(j_decompress_ptr cinfo,
2  jpeg_component_info * compptr, JCOEFPTR coef_block,
3  JSAMPARRAY output_buf, JDIMENSION output_col) {
4    ...
5    /* Pass 1: process columns from input. */
6    inptr = coef_block;
7    for (ctr = DCTSIZE; ctr > 0; ctr--) {
8      if (inptr[DCTSIZE*1]==0 && inptr[DCTSIZE*2]==0 &&
9      inptr[DCTSIZE*3]==0 && inptr[DCTSIZE*4]==0 &&
10   inptr[DCTSIZE*5]==0 && inptr[DCTSIZE*6]==0 &&
11   inptr[DCTSIZE*7]==0) {
12   ---- Simple Calculation ---
13   continue;}
14   --- Complex Calculation ---
15   }
16   /* Pass 2: process rows from work array. */
17   wsptr = workspace;
18   for (ctr = 0; ctr < DCTSIZE; ctr++) {
19 #ifndef NO_ZERO_ROW_TEST
20     if (wsptr[1]==0&&wsptr[2]==0&&wsptr[3]==0&&
21     wsptr[4]==0&&wsptr[5]==0&&wsptr[6]==0&&
22     wsptr[7]==0) {
23       ---- Simple Calculation ---
24       continue; }

26 #endif
27     --- Complex Calculation ---
28     --- (with a lot of multiplication calculations) ---
29  }}
```

Fig. 4: IDCT function in libjpeg.

In IDCT algorithm [29], there are two loops to handle a block (*i.e.* eight columns and eight rows). A simple calculation applies when all elements in a row or a column are zeros; otherwise, a complex calculation with more page faults applies. The attacker can then infer the value of each block by normalizing the number of data-page faults. To mitigate this vulnerability, libjpeg (version 6b, Figure 4) implemented the flag `NO_ZERO_ROW_TEST`. When the flag is enabled, all rows use complex calculation, thus page faults can not infer data in rows. Thus increases the difficulty of using only page fault information to recover JPEG images.

In this experiment, we demonstrate that the power information can further be exploited to infer rows in JPEG images on the IDCT implementation of the newest libjpeg library. We first present that the power-based attack is also useful when the program is vulnerable to order-based attacks. Using the emulation-based interpolator to measure

the power consumption of the simple and the complex calculations (wherein the input is a column with four pixels equals 0) and using the SPA to analyze the results. Particularly, we amplified some target instructions $100,000$ times, measured the power consumption, and inferred the instruction and the secret. The results are presented in Figure 5(a), which indicates that the power of the simple and complex calculations can be easily distinguished with the emulation-based interpolator.



(a) Columns Processing (b) Rows Processing

Fig. 5: Energy Consumption of Emulation-based Interpolator.

Then, we demonstrate the power information is another covert channel when order-based attacks is mitigated. To analyze the energy consumption of rows of JPEG images, we use emulation-based approach to simulate/collect the power consumption. We measured the power consumption of each instruction $100,000$ times and amplified each one $100,000$ times with the emulation-based interpolator. Finally, we applied the SPA to analyze the results. The results are presented in Figure 5(b), where the blue bars are the rows with all bits (except the first one) as zeros, and the orange bars are the rows with the Hamming weight equal to 4. The result indicates that these two conditions are discernible. This is because one of the most energy-consuming calculations is multiplication, which consumes much less energy when multiplying by 0. While this paper did not reconstruct the JPEG image, attackers may still be able to recover the whole image as the columns and rows with all bits as zeros is discernible [51].

5.2 Steal Private Exponent in RSA

RSA is a widely used asymmetric cryptographic algorithm. Modular exponentiation is one of the most important components of the RSA algorithm. To mitigate attacks such as SPA [33] and DPA [22], the modular exponentiation algorithm with message blinding [14,44] was discovered (as shown in Algorithm 1) by introducing a random variable. However, attackers could still exploit this algorithm when they can discover the correlation between the private exponent and the power consumption.

Here we targeted at a non-constant time RSA implementation of Intel's Integrated Performance Primitives (Intel IPP) library [19] with modular exponentiation algorithm with message blinding, and we exploited both emulation-based and interrupt-based interpolators to infer the private exponent. The RSA implementation in the IPP library first calls the `ippsRSA_Decrypt` / `ippsRSA_Encrypt` function and then selects the actual function (*e.g.*, `gsRSAprv_cipher`) for the encryption and the decryption based on the instruction set supported by the CPU. We evaluated the effect of the power-based attacks on it with a 512-bit RSA private exponent.

Interrupt-based Interpolator. Firstly, we try to exploit the interrupt-based interpolator on a modular exponentiation algorithm with message blinding. As the algorithm runs inside the SEV, neither plaintext nor ciphertext are available to adversaries. Thus, the CCA attack [34,23] is selected. The underlying assumption of the CCA attack is that the correlation coefficient of the power consumption between two "$z = z * T[d_i]$ *mod* n" operations would be higher if the values of two $T[d_i]$s are the same. Otherwise, the correlation coefficient would be relatively low.

Algorithm 1: Modular Exponentiation

```
Input: x, n, d=(d_{e-1}, d_{e-2}, ..., d_2, d_1, d_0),
a=r - 1 mod n (r is a random number)
Output: x^d mod n
begin
    T[0] ← a, T[1] ← x*r mod n, z ← r mod x
    foreach i = e - 1 to 0 do
        z ← z*z mod n, z ← z*T[d_i] mod n
    end
    z=z*a mod n
    return z
end
```

Power Trace Collection. For verification purposes, we use a relatively short RSA private exponent (512-bit) to decrypt a ciphertext in this experiment. To collect the power traces of the RSA operation, we focus on the "$z = z * T[d_i]$ *mod* n" operation in each iteration. For each instruction of this operation, we apply the interrupted-based interpolator to amplify the power consumption of each instruction *N* times (zero-stepping). As N increases, the precision of instruction's power consumption also becomes greater. Then, we organize all the power information collected into our defined format (as shown in Definition 1). In total, $3,000,000$ power traces were collected.

***Definition* 1** *Let $\mathcal{P}i = \{\mathcal{P}i_0, \mathcal{P}i_1, \cdots, \mathcal{P}i_r\}$ be the power trace of i_{th} execution of the RSA decryption operation, and in total r power traces collected. $\mathcal{P}i_j = \{e_{i,j,0}, e_{i,j,1}, \cdots, e_{i,j,k}\}$ corresponding to the power information of the j_{th} iteration of the multiplication operation ("$z = z * T[d_i]$ mod n") for the i_{th} power trace. This multiplication operation has k instructions, and $e_{i,j,k}$ indicates the power consumption of the k_{th} instruction in it.*

Correlation Calculation. To calculate the correlation coefficient, we first randomly select a jth bit as the reference bit. Then we calculate the Pearson correlation coefficient of power consumption between the jth bit and all other bits with the equation shown in Equation 1 [50]. The result of this equation is the correlation coefficient of the power consumption for the same instruction between different bits in the private exponent.

$$\rho(\mathcal{P}i_{j1}, \mathcal{P}i_{j2}) = \frac{\sum_{i=0}^{r-1}(e_{i,j1,1}e_{i,j2,1}) - \frac{\sum_{i=0}^{r-1} e_{i,j1,1} \sum_{i=0}^{r-1} e_{i,j2,1}}{r}}{\sqrt{(\sum_{i=0}^{r-1} e_{i,j1,1}^2 - \frac{(\sum_{i=0}^{r-1} e_{i,j1,1})^2}{r})}\sqrt{(\sum_{i=0}^{r-1} e_{i,j2,1}^2 - \frac{(\sum_{i=0}^{r-1} e_{i,j2,1})^2}{r})}} \quad (1)$$

An example of this algorithm is shown in Figure 6. Each column is the power consumption of an instruction in the operation "$z = z * T[d_i]$ *mod* n". For instance, the first red column is the power consumption of the first instruction in the operation when processing bit 0, and the second red column is the power consumption of the first instruction when processing bit j. We calculate the power correlation between the same instruction in different bits; thus, 512 correlations are calculated for each instruction. When the operation "$z = z*T[d_i]$ *mod* n" consists of k instructions, $512*k$ correlations are calculated in total.

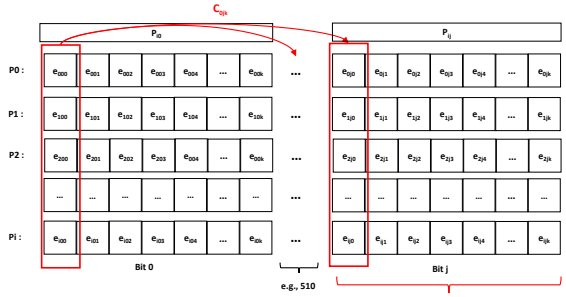


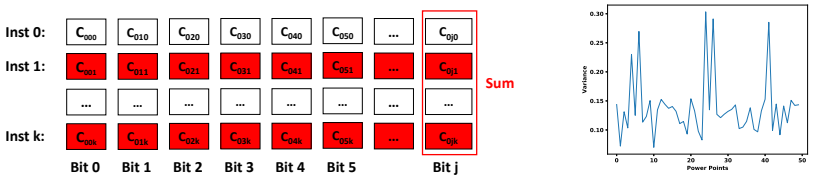
Fig. 6: Correlation Calculation Algorithm.

Exploit Key Bits. We use the correlation coefficient to distinguish bits in the private exponent [47]. In particular, a higher correlation coefficient indicates that two bits (*e.g.*, $T[0]$ or $T[1]$) are the same. A relatively low correlation coefficient means that either two bits are different, or a significant amount of noise is included in the power information. As the noise introduced by the APIC interrupt and the system (*e.g.*, random time delay, random clock, *etc.*) could affect the result of the correlation coefficient, we only keep those instructions that have a high correlation coefficient. In particular, the following two steps are used:

- We filter the noise with an intermediate value, the variance. As shown in Figure 7a, each row consists of correlation values of the same instruction from various bits. We first calculate the variance for each instruction (each row), then keep those instructions with a relatively higher variance. A higher variance (peaks in Figure 7b) means that the power information of this instruction can help infer the private exponent.
- Then, we add up the correlation coefficient of these selected instructions (*e.g.*, rows in red in Figure 7a) for each bit, then deduce the private exponent based on the value of the summed correlation coefficients with the threshold-based approach. A higher sum of the correlation coefficient means that this bit has a higher chance to be the same as the reference bit. A lower correlation indicates that this bit has higher probability to be opposite to the reference bit. Thus, an attacker can use correlation analysis to infer the value of each bit and have different levels of confidence in the predicted value of each bit.

Evaluation Results. We randomly generated a 512-bit RSA key using openssl and used the two interpolators discussed above to recover the private exponent. To recover this RSA key (private exponent), we recorded $3,000,000$ traces in total.

Interrupted-based Interpolator. The interrupted-based interpolator method could correctly infer 427 out of 512 bits of the private exponent by comparing with the correct key. With only correlation values, it is hard to know which bits of the key do not recover correctly. When comparing this result with other works with a recovery rate greater than 90% [5], the lower recovery rate of the interrupt-based interpolator indicates that this approach is affected by some noise. Such noise might be averaged out with the increasing number of traces collected, we leave this for future work. However, the time needed for collecting power consumption data also increases with the number of traces, and in the experiment described in the paper, collecting 3 million traces took around 80 hours.



(a) Noise Filtering. (b) Correlation Coefficient.

Fig. 7: Interrupt-based Interpolator.

Emulation-based Interpolator. With the same approach discussed above, we measured the power consumption of each instruction in the operation "$z \leftarrow z$*$T[d_i]$ *mod* n" with the emulation-based interpolator. In particular, for the evaluation purpose, we generated the RSA key pair with a 512-bit modulus. For each instruction, we applied the emulation-based interpolator to amplify the power consumption $100,000$ times. In total, $1,000,000$ different power traces were collected. After applying the CCA attack, we successfully recovered the private exponent without an error.

6 Discussion

Countermeasures. The power-based side-channel attack needs to gather fine-grained power information during run-time in order to analyze and infer secret using the collected data. Thus, adding additional noise may be one potential way to prevent power-based side-channel attacks. The power-based side-channel attack could be prevented if the hardware or the VM itself could add noises during run-time to hide the power consumption pattern caused by different instructions and operands. The hardware manufacturers can also use a microcode patch to disable the hardware reporting interfaces of TEE's power assumption to prevent such attacks. For example, affected by two general power-based side-channel attacks [31,30], AMD has added restrictions on accessing power-consumption interfaces in newer kernel versions, and Linux has also removed some related drivers that could potentially cause leakage [35].

Comparison with Other Power Attacks. To launch a power-based side-channel attack, the CPA attack is a widely used method [10,32,9], which can resist noise. However, CPA attack mainly targets at algorithms and requires either plaintext or ciphertext, which is not the case in the AMD SEV's scenario. The most related work to us is Platypus [31], which uses Intel RAPL to break Intel SGX protection, steal private keys from constant-time cryptographic implementation (AES-NI), and study power consumption of instructions in AMD platform. Considering that different CPU hardware and TEE design (Intel SGX aims at protecting an application instead of a VM) could introduce different power pattern, PWRLEAK could be a complementary work to Platypus with similar approaches but target a different TEE design and cryptographic implementation with low-secure level (non-constant time Intel IPP library).

Future Work. Due to equipment limitations, we did not perform experiments on SEV-ES and SEV-SNP. Here we discuss the feasibility of power side-channel attacks on these machines and treat them as future work. The current version of the emulation-based interpolator could not work on SEV-ES and SEV-SNP due to the encrypted VMSA. The interrupt-based interpolator could potentially work on both SEV-ES and SEV-SNP.

However, this approach would encounter a substantial amount of unstable noise introduced by additional protection from SEV-ES or SEV-SNP. For example, in SEV-ES, there is an integrity check for the VM Save Area region (the region used to encrypt and backup registers) during each VMEXIT or VMRUN. In SEV-SNP, each memory write access in the case of a TLB miss introduces a Reverse Map Table (RMP) check. These additional protections may introduce inaccuracies in the observed energy consumption. Therefore, a precise noise cancellation algorithm or an amplifier that can further magnify the difference in power consumption may be necessary for such side-channel attacks to work in SEV-ES or SEV-SNP VMs.

7 Related Work

Other Attacks against AMD SEV. AMD SEV has been studied by both industry and academia since its first release in 2016. Faced with a strong threat model in which the entire software stack is not trusted, previous work showed that AMD SEV suffers from numerous attack surfaces, including both *incomplete system designs* [39,25,15,28,49,18] and *side-channel attacks* [27,24,48]. For *incomplete system designs*, AMD actively addresses existing attacks by providing microcode patches [7] and adding new hardware extensions (including AMD SEV Encrypted States (SEV-ES) [21] and AMD SEV Secure Nested Paging (SEV-SNP) [6]) in addition to the baseline AMD SEV. However, for *side-channel attacks*, which are not included in AMD SEV's thread model and indirectly leak secret from the SEV-protected VM [26,27,24], AMD typically does not provide fixes for such attacks. Common side-channel attacks in AMD SEV include page table-based side-channel attacks [40], cache side-channel attacks, PMC-based side-channel attacks [48], and ciphertext side-channel attacks [24]. The defense mechanisms against such side-channel attacks often involve refactoring source code to avoid certain patterns or gadgets, or adopting code with constant-time implementations.

8 Conclusion

In this paper, we have demonstrated the potency of power-based side-channel attacks in extracting secrets from AMD SEV-protected VMs. Through a series of exploratory experiments and an emulation-based interpolator, we show that adversaries can still notice the differences in the instruction and operand level with the $1ms$ coarse-grained power sampling interval provided by AMD. Additionally, we have successfully leaked a random generated RSA key in an IPP implementation using PWRLEAK.

Acknowledgments

We would like to thank the anonymous reviewers and the shepherd Moritz Lipp for their very helpful comments and feedback during revision, which have significantly improved the quality and clarity of the work. This research was partially supported by NSF award 2207202. Any opinions, findings, and conclusions or recommendations in this paper are those of the authors and do not necessarily reflect the views of the NSF.

References

1. Confidential computing: an AWS perspective . https://aws.amazon.com/blogs/security/confidential-computing-an-aws-perspective/, 2021. Aug, 2021.
2. SEV Secure Nested Paging Firmware ABI Specification. https://www.amd.com/system/files/TechDocs/56860.pdf, 2021.
3. Arm Confidential Compute Architecture . https://www.arm.com/architecture/security-features/arm-confidential-compute-architecture, 2022. Dec, 2022.
4. Intel trust domain extensions. https://www.intel.com/content/www/us/en/developer/articles/technical/intel-trust-domain-extensions.html, 2022. Dec, 2022.
5. Ebru Akalp Kuzu, Betül Soysal, Muhammet Şahinoğlu, Umut Güvenç, and Ali Tangel. New cross correlation attack methods on the montgomery ladder implementation of rsa. In *2013 3rd IEEE International Advance Computing Conference (IACC)*, pages 138–142, 2013.
6. AMD. AMD SEV-SNP: Strengthening VM isolation with integrity protection and more. *White paper*, 2020.
7. AMD. AMD Secure Encryption Virtualization (SEV) Information Disclosure (Bulletin ID: AMD-SB-1013). https://www.amd.com/en/corporate/product-security/bulletin/amd-sb-1013, 2021.
8. AMD. AMDSEV branch. https://github.com/AMDESE/AMDSEV/, 2022.
9. Paul Bottinelli and Joppe W Bos. Computational aspects of correlation power analysis. *Journal of Cryptographic Engineering*, 7(3):167–181, 2017.
10. Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In *International workshop on cryptographic hardware and embedded systems*, pages 16–29. Springer, 2004.
11. Suresh Chari, Josyula R Rao, and Pankaj Rohatgi. Template attacks. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 13–28. Springer, 2002.
12. Yimin Chen, Xiaocong Jin, Jingchao Sun, Rui Zhang, and Yanchao Zhang. Powerful: Mobile app fingerprinting via power analysis. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE, 2017.
13. Christophe Clavier, Benoit Feix, Georges Gagnerot, Mylene Roussellet, and Vincent Verneuil. Improved collision-correlation power analysis on first order protected aes. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 49–62. Springer, 2011.
14. Jean-Sébastien Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In *International workshop on cryptographic hardware and embedded systems*, pages 292–302. Springer, 1999.
15. Zhao-Hui Du, Zhiwei Ying, Zhenke Ma, Yufei Mai, Phoebe Wang, Jesse Liu, and Jesse Fang. Secure encrypted virtualization is unsecure. *arXiv preprint arXiv:1712.05090*, 2017.
16. Thomas Eisenbarth, Christof Paar, and Björn Weghenkel. Building a side channel based disassembler. In *Transactions on computational science X*, pages 78–99. Springer, 2010.
17. Google. Introducing google cloud confidential computing with confidential VMs. https://cloud.google.com/blog/products/identity-security/introducing-google-cloud-confidential-computing-with-confidential-vms, 2020.
18. Felicitas Hetzelt and Robert Buhren. Security analysis of encrypted virtual machines. *ACM SIGPLAN Notices*, 52(7):129–142, 2017.

19. Intel integrated performance primitives. https://software.intel.com/content/www/us/en/develop/tools/oneapi/components/ipp.html.
20. Kouichi Itoh, Dai Yamamoto, Jun Yajima, and Wakaha Ogata. Collision-based power attack for RSA with small public exponent. *IEICE transactions on information and systems*, 92(5):897–908, 2009.
21. David Kaplan. Protecting VM register state with SEV-ES. *White paper*, 2017.
22. Paul Kocher, Joshua Jaffe, Benjamin Jun, et al. Introduction to differential power analysis and related attacks, 1998.
23. Ebru Akalp Kuzu, Betül Soysal, Muhammet Şahinoğlu, Umut Güvenç, and Ali Tangel. New cross correlation attack methods on the montgomery ladder implementation of rsa. In *2013 3rd IEEE International Advance Computing Conference (IACC)*, pages 138–142. IEEE, 2013.
24. Mengyuan Li, Luca Wilke, Jan Wichelmann, Thomas Eisenbarth, Radu Teodorescu, and Yinqian Zhang. A Systematic Look at Ciphertext Side Channels on AMD SEV-SNP. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1541–1541. IEEE Computer Society, 2022.
25. Mengyuan Li, Yinqian Zhang, and Zhiqiang Lin. Crossline: Breaking "security-by-crash" based memory isolation in AMD SEV. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 2937–2950, 2021.
26. Mengyuan Li, Yinqian Zhang, Zhiqiang Lin, and Yan Solihin. Exploiting unprotected I/O operations in AMD's secure encrypted virtualization. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 1257–1272, 2019.
27. Mengyuan Li, Yinqian Zhang, Huibo Wang, Kang Li, and Yueqiang Cheng. CIPHERLEAKS: Breaking Constant-time Cryptography on AMD SEV via the Ciphertext Side Channel. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 717–732, 2021.
28. Mengyuan Li, Yinqian Zhang, Huibo Wang, Kang Li, and Yueqiang Cheng. TLB Poisoning Attacks on AMD Secure Encrypted Virtualization. In *Annual Computer Security Applications Conference*, pages 609–619, 2021.
29. Libjpeg. Libjpeg version 6b Files. https://sourceforge.net/projects/libjpeg/files/libjpeg/6b/.
30. Moritz Lipp, Daniel Gruss, and Michael Schwarz. AMD Prefetch Attacks through Power and Time. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 643–660, 2022.
31. Moritz Lipp, Andreas Kogler, David Oswald, Michael Schwarz, Catherine Easdon, Claudio Canella, and Daniel Gruss. Platypus: Software-based power side-channel attacks on x86. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 355–371. IEEE, 2021.
32. Owen Lo, William J Buchanan, and Douglas Carson. Power analysis attacks on the AES-128 S-box using differential power analysis (DPA) and correlation power analysis (CPA). *Journal of Cyber Security Technology*, 1(2):88–107, 2017.
33. Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks: Revealing the secrets of smart cards*, volume 31. Springer Science & Business Media, 2008.
34. Thomas S Messerges, Ezzy A Dabbish, and Robert H Sloan. Power analysis attacks of modular exponentiation in smartcards. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 144–157. Springer, 1999.
35. Michael Larabel. AMD Energy Driver Booted From The Linux 5.13 Kernel. https://www.phoronix.com/news/Linux-5.13-AMD-Energy-Removed, 2021.
36. Yan Michalevsky, Aaron Schulman, Gunaa Arumugam Veerapandian, Dan Boneh, and Gabi Nakibly. PowerSpy: Location Tracking Using Mobile Device Power Analysis. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 785–800, 2015.
37. Microsoft. Azure and AMD announce landmark in confidential computing evolution. https://azure.microsoft.com/en-us/blog/azure-and-amd-enable-lift-and-shift-\confidential-computing/, 2021.

38. Amir Moradi, Oliver Mischke, and Thomas Eisenbarth. Correlation-enhanced power analysis collision attack. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 125–139. Springer, 2010.
39. Mathias Morbitzer, Manuel Huber, and Julian Horsch. Extracting secrets from encrypted virtual machines. In *Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy*, pages 221–230, 2019.
40. Mathias Morbitzer, Manuel Huber, Julian Horsch, and Sascha Wessel. Severed: Subverting AMD's virtual machine encryption. In *Proceedings of the 11th European Workshop on Systems Security*, pages 1–6, 2018.
41. Yongchuan Niu, Jiawei Zhang, An Wang, and Caisen Chen. An efficient collision power attack on AES encryption in edge computing. *IEEE Access*, 7:18734–18748, 2019.
42. Jungmin Park, Xiaolin Xu, Yier Jin, Domenic Forte, and Mark Tehranipoor. Power-based side-channel instruction-level disassembler. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2018.
43. Daehyun Strobel, Florian Bache, David Oswald, Falk Schellenberg, and Christof Paar. Scandalee: a side-channel-based disassembler using local electromagnetic emanations. In *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 139–144. IEEE, 2015.
44. Yen Sung-Ming, Seungjoo Kim, Seongan Lim, and Sangjae Moon. A countermeasure against one physical cryptanalysis may benefit another attack. In *International Conference on Information Security and Cryptology*, pages 414–427. Springer, 2001.
45. Jo Van Bulck, Frank Piessens, and Raoul Strackx. SGX-step: A practical attack framework for precise enclave execution control. In *Proceedings of the 2Nd Workshop on System Software for Trusted Execution*, (SysTEX'17), 2017.
46. Dennis Vermoen, Marc Witteman, and Georgi N Gaydadjiev. Reverse engineering JAVA card applets using power analysis. In *IFIP International Workshop on Information Security Theory and Practices*, pages 138–149. Springer, 2007.
47. Wunan Wan, Wei Yang, and Jun Chen. An optimized cross correlation power attack of message blinding exponentiation algorithms. *China Communications*, 12(6):22–32, 2015.
48. Jan Werner, Joshua Mason, Manos Antonakakis, Michalis Polychronakis, and Fabian Monrose. The SEVerESt Of Them All: Inference Attacks Against Secure Virtual Enclaves. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, pages 73–85, 2019.
49. Luca Wilke, Jan Wichelmann, Mathias Morbitzer, and Thomas Eisenbarth. Sevurity: No security without integrity: Breaking integrity-free memory encryption with minimal assumptions. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1483–1496. IEEE, 2020.
50. Marc F Witteman, Jasper GJ van Woudenberg, and Federico Menarini. Defeating RSA multiply-always and message blinding countermeasures. In *Cryptographers' Track at the RSA Conference*, pages 77–88. Springer, 2011.
51. Yuanzhong Xu, Weidong Cui, and Marcus Peinado. Controlled-channel attacks: Deterministic side channels for untrusted operating systems. In *Proceedings of the 2015 IEEE Symposium on Security and Privacy (SP'15)*. IEEE, 2015.
52. Shengqi Yang, Wayne Wolf, Narayanan Vijaykrishnan, Dimitrios N Serpanos, and Yuan Xie. Power attack resistant cryptosystem design: A dynamic voltage and frequency switching approach. In *Design, Automation and Test in Europe*, pages 64–69. IEEE, 2005.
53. Alan Zeichick. *Security Ahoy! Flying the NX Flag on Windows and AMD64 To Stop Attacks*. Advanced Micro Devices, March 2007.
54. Bing Zhao, Lihui Wang, Kun Jiang, Xiaobing Liang, Weijun Shan, and Jing Liu. An improved power attack on small RSA public exponent. In *2016 12th International Conference on Computational Intelligence and Security (CIS)*, pages 578–581. IEEE, 2016.