Do Not Trust Power Management: A Survey on Internal Energy-based Attacks Circumventing Trusted Execution Environments Security Properties

GWENN LE GONIDEC, UMR 6285, Lab-STICC, Univ. Bretagne-Sud, France
GUILLAUME BOUFFARD, National Cybersecurity Agency of France (ANSSI), France
JEAN-CHRISTOPHE PRÉVOTET, UnivRennes, INSA Rennes, CNRS, IETR-UMR 6164, France
MARIA MÉNDEZ REAL, UMR 6285, Lab-STICC, Univ. Bretagne-Sud, France

Over the past few years, several research groups have introduced innovative hardware designs for Trusted Execution Environments (TEEs), aiming to secure applications against potentially compromised privileged software, including the kernel [10, 63]. Since 2015 [94], a new class of software-enabled hardware attacks leveraging energy management mechanisms has emerged. These internal energy-based attacks comprise fault [86], side-channel [46] and covert channel attacks [28]. Their aim is to bypass TEE security guarantees and expose sensitive information such as cryptographic keys. They have increased in prevalence in the past few years [9, 24, 40]. Popular TEE implementations, such as ARM TrustZone and Intel SGX, incorporate countermeasures against these attacks. However, these countermeasures either hinder the capabilities of the power management mechanisms or have been shown to provide insufficient system protection [9, 55]. This article presents the first comprehensive knowledge survey of these attacks, along with an evaluation of literature countermeasures. We believe that this study will spur further community efforts towards this increasingly important type of attacks.

CCS Concepts: • **General and reference** → **Surveys and overviews**; • **Security and privacy** → **Hardware attacks and countermeasures**; *Embedded systems security*.

1 Introduction

To protect sensitive assets, modern mobile computing systems rely on Trusted Execution Environments (TEEs). TEEs offer a secure runtime execution environment where sensitive applications are executed, co-located with a Rich Execution Environment (REE), which may include modern and complex Operating Systems (OSes) such as Android or iOS. Relying on hardware mechanisms, TEEs isolate the execution of sensitive applications from the rest of the system, *i.e.*, from the untrusted REE. They are widely used to secure critical applications in System on Chips (SoCs) embedded in powerful and complex devices, from smartphones to high-end servers. These SoCs are powered by complex Central Processing Units (CPUs) with advanced micro-architectures, such as memory virtualization, asymmetric processing units, multiple cache levels, and speculative or out-of-order execution. TEE and REE share the same physical resources

Authors' Contact Information: Gwenn Le Gonidec, owen.le-gonidec@univ-ubs.fr, UMR 6285, Lab-STICC, Univ. Bretagne-Sud, Lorient, France; Guillaume Bouffard, guillaume.bouffard@ssi.gouv.fr, National Cybersecurity Agency of France (ANSSI), Paris, France; Jean-Christophe Prévotet, jean-christophe.prevotet@insa-rennes.fr, UnivRennes, INSA Rennes, CNRS, IETR-UMR 6164, Rennes, France; Maria Méndez Real, maria.mendez-real@univ-ubs.fr, UMR 6285, Lab-STICC, Univ. Bretagne-Sud, Lorient, France.

ACM 1558-3465/2025/5-ART
https://doi.org/10.1145/3735556

(processing units and memory). Designing innovative hardware-based protection mechanisms for next-generation TEEs is an active research field. Over the past few years, several research groups have leveraged the capabilities of the RISC-V open Instruction Set Architecture (ISA) for this purpose [10, 45, 77].

TEEs aim to protect the system against attacks where the attacker may have full control over the REE, including the OS. They typically do not provide protection against an attacker who has physical access to the device [12, 18, 63]. However, since the mid-2010s, numerous studies have demonstrated various methods by which a remote attacker can compromise the main security properties of a TEE, utilizing hardware attacks initiated from software without needing physical access to the target [25]. They rely on the manipulation of software-accessible interfaces to affect hardware components. This emerging class of attacks stems from the complexity of the systems on which TEEs are implemented, which attempt to balance performance, power constraints and security. At the intersection of software and hardware attacks, they benefit from the best of both approaches. They take advantage of hardware attack techniques developed over the last two decades and, as software attacks, they can be executed remotely and on a massive scale. This makes them a realistic and significant threat.

Of particular concern are attacks that maliciously exploit embedded power management mechanisms, referred to as *internal energy-based attacks* in this survey. These attacks can be employed to extract sensitive assets such as cryptographic keys or to bypass authentication procedures. They enable the injection of timing faults in a Fault Injection Attack (FIA) [58, 86], extraction of secret information through a Side-Channel Attack (SCA) [14, 46, 88], and the creation of a covert communication channel utilized by a Trojan and a spy [5, 28]. Such attacks have been conducted against some of the most widely deployed TEE designs, namely Arm TrustZone [5, 86, 88] and Intel Software Guard Extension (SGX) [46, 51, 58].

Among commercial TEE implementations, Intel SGX has introduced a mitigation for energy-based FIAs. However, this approach involves restricting power management mechanisms, significantly limiting the primary objective of fine-grained energy optimization [58]. Arm [2] recommends that vendors of TrustZone-based TEEs (*e.g.*, Qualcomm or Samsung) implement similar mitigation; however, to our knowledge, no official vendor documentation confirms an actual implementation. This countermeasure is ultimately unsustainable, as it leads to power waste and fails to address the root cause of energy-based attacks. For energy-based SCAs [88] and covert attacks, Intel has implemented a noise-based countermeasure [31]. However, recent studies have demonstrated that it does not fully prevent such attacks [9]. RISC-V-based TEE designs do not embed any protection against internal energy-based attacks [3, 45, 77, 90].

Consequently, it is argued that existing TEEs are vulnerable to this significant threat. A promising area of research is the development of new countermeasures that can effectively counter internal energy-based attacks while preserving the full capabilities of power management mechanisms.

This article provides a comprehensive overview of this new category of hardware energy-based attacks, analysing their capabilities, limitations, and evolution over the past few years. To the best of our knowledge, this is the first work to review this emerging category of attacks. Additionally, the initial attempts at countermeasures are reviewed.

The remainder of this article is organized as follows: Section 2 provides background information on physical attacks, power management mechanisms, and TEEs. Subsequently, Section 3 delves into internal energy-based attacks, detailing their methods, results, and limitations. In Section 4, the first published and implemented countermeasures against energy-based attacks are analysed, highlighting their shortcomings and providing an overview of the challenges involved in their practical implementation. In Section 5, the issue is discussed from a broader perspective, through the analysis of the practical exploitability of internal energy-based attacks in recent devices, and the study of energy-based attacks beyond the scope of CPUs and TEEs. Finally, Section 6 concludes this article and offers insights into future research directions.

2 Background

2.1 Trusted Execution Environments

Some privileged components, such as REE, have extensive attack surfaces that make them challenging to secure. For example, the Linux kernel comprised over 27 million source lines of code in 2020, with nearly two thousand vulnerabilities disclosed [80]. Therefore, these privileged components cannot be relied upon to ensure the security of critical programs. This realization gave rise to the concept of relying on a minimal set of security-oriented components to form a Trusted Computing Base (TCB). The aim is to keep the TCB as small as possible, providing hardware-assisted mechanisms to isolate critical programs from the main computing environment. Since the late 1990s, separate co-processors known as Trusted Platform Modules (TPMs) have been utilized for this purpose [62]. However, programs running on a TPM cannot benefit from the full power of the SoC's components. The necessity to extend this protection to third-party programs on rich, performance-oriented processors led to the development and standardization of TEEs in the early 2010s [73].

A TEE is a secure and performance-oriented environment comprising memory, storage, and processing capabilities, isolated from the rest of the system, often referred to as the REE [16]. While a TPM relies on executing security-critical programs on an external, isolated component, in a TEE, both Trusted Applications (TAs) and untrusted programs run on the same CPU and share the same hardware resources. These resources' access is typically mediated by a dedicated component known as the *Secure Monitor*. The TEE utilizes on-chip hardware mechanisms to isolate and provide integrity and confidentiality to security-critical programs in systems where privileged software, such as kernels and hypervisors in the REE, is untrusted. This includes servers used by stakeholders for data storage and computation outsourcing (cloud computing), where trust in the cloud provider may be lacking. This scenario is a typical use case for Intel SGX, Intel Trust Domain Extensions (TDX) and AMD SEV, which are proprietary TEE designs. In the embedded market, Arm TrustZone provides hardware support for TEEs to SoC designers. In the embedded world, TEEs can be used for various applications as well, such as secure telemetry, biometry and digital rights management. More recently, numerous research papers have leveraged the open RISC-V ISA to propose new hardware designs for TEE support, such as Sanctum [10] and Keystone [45], among others.

TEE implementations vary based on the hardware and software mechanisms they employ to secure trusted applications. A notable distinction between Arm TrustZone and Intel SGX-type TEEs (which encompass most academic proposals [3, 10, 77]), is that in the former, the entire system is divided into two *worlds*; while the latter aims to secure individual applications known as *enclaves*. Figure 1 shows a simple implementation of a dual-world based TEE similar to TrustZone [63]. Secure OSes and potentially hypervisors operate in the trusted world. In addition, the Secure Monitor may be able to reserve certain peripheral devices and areas of memory for the exclusive use of the trusted world. This approach offers significant flexibility: TEE implementations can prioritize providing extensive functionalities to trusted applications or aim for the smallest possible TCB, as demonstrated in studies [63]. However, in the latter approach, untrusted user-level applications, referred to as enclaves, can be individually isolated without needing to trust the kernel and other privileged software. In this protection model, there is a risk of malicious enclaves being spawned.

Another essential feature of TEE is attestation, which ensures that it operates on a physically certified device. This serves various purposes, such as allowing a trusted application to verify that the host system

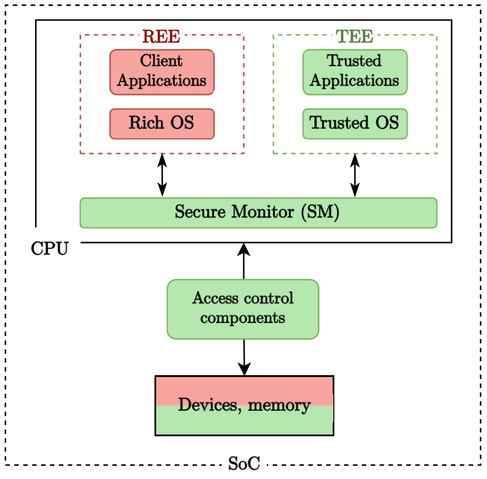

Fig. 1. Example implementation of the main components in a dual-world-based TEE. Red colored components are untrusted, green colored components are trusted. Devices and memory are divided in trusted and untrusted portions.

is running the latest version. Additionally, TEEs rely on security primitives like Secure Boot, typically enforced by firmware, which forms part of the TCB of the system.

Recently, an important area of research has emerged in the design of innovative hardware mechanisms for next-generation TEE with various objectives, such as adaptation to real-time constraint [90], flexibility [3] and robustness against advanced attack schemes, *e.g.*, controlled-channel attacks relying on page faults [77].

2.2 Power management mechanisms

Optimal power management is crucial in digital systems design, particularly for complex CPUs, enabling energy savings and improved thermal management. Among power management technologies, fine-grained control mechanisms such as Dynamic Voltage and Frequency Scaling (DVFS) have been employed for decades. DVFS involves adjusting the supply voltage and operating frequency, which together have a cubic impact on power consumption, based on system load to meet performance targets. In this article, a combination of operating frequency and supply voltage is referred to as an Operating Point (OPP) (known as *P-States* in Intel architecture terminology). In a typical DVFS implementation, either the OS or the hardware CPU requests a frequency adjustment based on workload estimation. The operating voltage is subsequently adjusted based on a set of vendor-defined OPPs. These sets are defined by manufacturers to ensure device safety and can extend beyond the actual operating limits of the device [97]. These limits vary due to factors such as ageing, temperature, and manufacturing uncertainties [86], making it impractical for hardware designers to accurately estimate them in advance. In addition to OPPs scaling mechanisms, the ability to query power consumption, temperature, and operating frequency is crucial for power management. Energy-aware programs benefit from software interfaces that allow them to monitor these metrics, and it is also directly used by hardware components to implement turbo frequencies and power capping [76].

Figure 2 provides an overview of the main components utilized in a typical DVFS implementation within a SoC featuring an application multi-core CPU, both for OPP scaling and sensing. It represents one

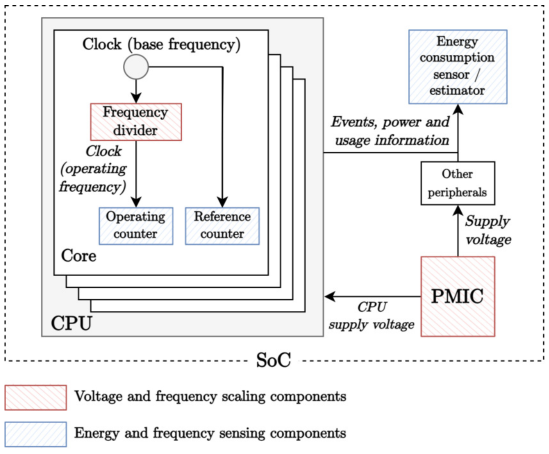
Clock (base frequency)
Frequency divider
Clock (operating frequency)
Operating counter
Reference counter
Core
CPU
Energy consumption sensor / estimator
Events, power and usage information
Other peripherals
Supply voltage
PMIC
CPU supply voltage
SoC
Voltage and frequency scaling components
Energy and frequency sensing components

Fig. 2. A SoC DVFS implementation, illustrating typical components used for sensing operating frequency and supply voltage.

implementation among many possibilities. Indeed, hardware implementations of such power management mechanisms can vary significantly from vendor to vendor and are often undocumented.

Energy consumption metrics can be retrieved by various means, depending on the platform considered. For instance, in Android-based devices, the power supply module communicates battery status to privileged software. On x86 processors, energy consumption can be measured from software using the Running Average Power Limit (RAPL) interface. Depending on the context, this interface does not always rely on physical power sensors; it may instead utilize a model based on architectural events communicated by peripherals [6].

The operating frequency is correlated to power consumption, as both are tied by DVFS mechanisms. It is typically measured by differentiating the values of incrementing counters; one incrementing with a reference frequency and the other incrementing in proportion to the actual performance[1] [30].

Frequency regulation in a CPU is handled by frequency dividers using Phase-Locked Loops (PLLs). Historically, voltage regulation has been predominantly managed by an off-chip Power Management Integrated Circuit (PMIC), as represented in Figure 2. However, on-chip Integrated Voltage Regulator (IVR) are becoming increasingly common [4], allowing for fast voltage transitions. Power is brought to the SoC components by a shared Power Distribution Network (PDN). At the software level, interaction with the frequency and voltage regulators is facilitated through dedicated OS modules. For instance, in Linux-based systems, the OS kernel requests frequency changes through modules such as `cpufreq`, either using an automatic governor or manually setting an operating frequency through Model-Specific Registers (MSRs). Voltage and current regulators can be controlled via a dedicated framework [15].

High voltage is required to achieve high operating frequencies. If the voltage is insufficient, timing conditions at the transistor level may not be met, resulting in device instability and clock glitches [23]. Although supply voltage is typically adjusted based on operating frequency, it is possible to independently adjust them by directly writing to dedicated MSRs. This presents an entry point for internal energy-based FIA, which intentionally triggers clock glitches by configuring a high clock frequency and a low supply voltage.

[1]See: `aperfmperf.c` in the Linux kernel source code.

Finally, energy management mechanisms are evolving, becoming more aggressive and sophisticated. While a core cluster used to be typically supplied by a single power rail, per-core supply voltage domains are becoming more common, as implemented in some recent Intel architecture-based CPU [78]. Additionally, the time required to change supply voltage is decreasing. The first energy-based FIA were conducted on systems where a change in voltage took about a millisecond to complete [58]. Nowadays, IVRs with fast voltage transitions are increasingly common. Intel's hardware-managed performance states significantly reduce the latency of both frequency and voltage changes [13, 78]. The literature demonstrates IVRs with sub-microsecond voltage transition delays [36]. Therefore, one can expect energy management mechanisms to continue evolving in this direction with faster, finer, more sophisticated energy management strategies.

2.3 Hardware attacks

Hardware attacks exploit the electronic behaviour of components executing sensitive or critical applications. In the case of SCA, rather than breaking complex mathematical algorithms such as cryptographic schemes protecting secret data, an attacker analyses side-channel signals generated during the execution of the victim application. Examples include the target device's electromagnetic emanation [8], power consumption [50], or computation time [19]. These signals reveal the target architecture and microarchitecture state, allowing an attacker to deduce the instructions and/or data computed, thereby compromising target confidentiality. Similarly, in FIAs, an attacker can introduce faults into computation or memory by inducing glitches within the electronic device through external sources, potentially resulting in controlled instruction skipping, such as during an authentication process. In cryptographic applications, an attacker may fault the result of specific instructions to infer cipher keys using dedicated methods such as Differential Fault Analysis (DFA). In both cases, SCA or FIA, the objective is typically to retrieve secret information.

Traditionally, hardware attacks have required physical access to the hardware device, such as through external sensors or sources of glitches. Consequently, hardware attacks were not always considered in most device threat models. However, a new type of software-induced hardware attack has emerged, allowing an attacker application to perform hardware attacks from within the device. By exploiting microarchitectural components shared by the victim and attacker applications, such as memory caches [75], communication interconnections [79], or more recently, power management components, this powerful new type of attack greatly extends the threat surface.

2.4 Timing constraints in digital systems

Digital systems are ruled by timing constraints. As shown in Figure 3, a synchronous circuit (such as an instruction) typically consists of two D-Flip-Flops (DFFs), which value changes upon receiving the synchronizing clock signal every t_{clock}. In-between those two is the combinational logic of the instruction. The output logic signal needs to be constant for t_{setup} during the clock signal's receiving in order for the output DFF to correctly register it. The input signal also needs to be constant for t_{DFF} after receiving the clock signal. Therefore, the combinational logic has to generate the output signal within t_{logic}, this time being ruled by the Equation (1):

$$t_{logic} \leq t_{clock} - t_{DFF} - t_{setup} \tag{1}$$

Failure to meet these constraints results in the output DFF's value to be corrupted. Assuming t_{DFF} and t_{setup} are constant, two things can cause these errors: either the clock frequency being to o high, or the combinational logic being too slow. The latter notably occurs when the system's supply voltage is insufficient, as the switching time of transistors is proportional to their supply voltage.

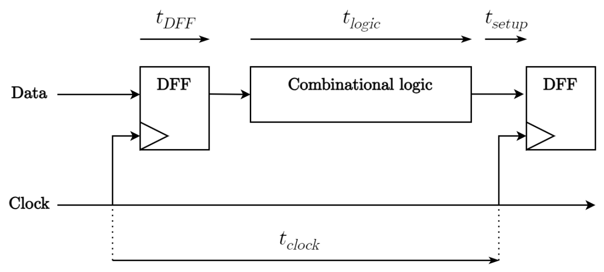

Fig. 3. Timing constraints in a digital system.

3 Internal Energy-based Attacks

As detailed in Section 2.3, physical access to the victim device enables potent power and clock-based attack scenarios, which have been studied and utilized against secure microcontrollers. Recently, several studies have shown that similar attacks can be executed internally using software-accessible energy management mechanisms against complex CPUs. This shift is relatively new, with such attacks first demonstrated in 2015 for SCAs [94] and in 2017 for FIAs [86]. Consequently, they pose an emerging threat that may become increasingly dangerous with further discoveries and when combined with other attack vectors.

Sensitive assets embedded in TEEs are common targets for internal energy-based attacks in recent literature [46, 86]. Earlier attacks often targeted untrusted programs [94]. However, with the widespread adoption of TEEs, research has shifted toward a privileged attacker model where trusted programs and data are the targeted assets, which are protected from direct access by the rich OS. Energy management mechanisms – voltage and frequency regulators and sensors – are shared across security domains, spanning both TEEs and REEs. An attacker with full control over the REE can exploit these energy management mechanisms as a side channel to access trusted assets. The methods by which these mechanisms can provide attack vectors for FIAs, SCAs, and covert communication are described in the following sections.

3.1 Attacker model

This section describes the attacker's objectives and capabilities for conducting internal energy-based attacks from within the system. These attacks employ methods similar to the physical attacks described in Section 2.3: they monitor programs using external power sensors and/or manipulate their execution using external sources of glitches. However, they differ significantly in a crucial aspect: internal attacks utilize software-accessible energy management mechanisms, eliminating the need for the attacker to physically access the device. While traditional physical attacks involve the attacker utilizing off-chip equipment to extract information or manipulate the execution flow, internal attacks are entirely software-driven, thus expanding the threat model to include remote attackers. This significantly increases the attack surface. Moreover, these internal attacks are more cost-effective, carry no risk of damaging the targeted platform, and pose a serious potential for widespread exploitation [82]. Typically, the attacker first identifies a vulnerability on their own copy of the target device. Then, malware is developed to exploit this vulnerability, carrying out the attack scenario defined in the previous phase. This malware can then be deployed for remote exploitation.

3.1.1 *Attacker's objectives.*

The attacker's goal is to compromise key security properties of the TEE and trusted applications/enclaves. Namely, the targeted security properties are 1) data confidentiality (*e.g.*, stealing cryptographic keys), 2) integrity (*e.g.*, altering processed data), 3) availability (*e.g.*, conducting Denial-of-Service attacks), or 4) authenticity (*e.g.*, loading a self-signed application into the TEE). This assumption implies that the CPU core currently running the TEE is the ultimate target. Therefore, energy-based attacks targeting hardware accelerators [93], Graphics Processing Units (GPUs) [72] or Field Programmable Gate Arrays (FPGAs) [41] are outside the scope of this work. However, attacks using on-chip FPGAs to target the CPU, such as FPGA-to-CPU undervolting FIAs [49], are included.

3.1.2 Attacker's capabilities. Our assumptions regarding the resources and capabilities of an attacker for internal energy-based attacks are based on the TEE Protection Profile (PP) [22] defined by the GlobalPlatform standards organization. This document serves as the primary reference for describing and evaluating the security of a TEE implementation. Attacker models are defined in its appendix. Our model is primarily derived from the *hardware vulnerabilities exploited from software* [22, Annex A, Section A.4.1] attack path. For energy-based attacks, the SCA and FIA attack paths defined in the PP are also considered. Thus, the main resources and capabilities of our assumed attacker are the following:

- Full control over the REE, including the OS, its drivers, and access to debugging tools.
- Comprehensive documentation of the targeted TEE implementation, including its hardware resources and cryptographic schemes.
- The ability to load a malicious trusted program in an attack on a cryptographic application allows the attacker to invoke operations with a targeted key, obtaining thousands of measurements of those operations.
- The capability to control the REE to minimize noise due to signal processing and manage calls to the TEE in a SCA.
- The use of side-channel analyses, including non-energy-related SCAs such as cache attacks, to find a trigger signal in a FIA.

Attackers are remote; they do not have physical access or proximity to the victim's device but can execute malicious software on it. However, they may utilize physical equipment on a copy of the device for profiling purposes before conducting the actual attack. For example, physical sensors can be employed to establish a correlation between power consumption and data on their own device during the profiling phase. Subsequently, during the actual attack, the developed model can be exploited on the victim's device through a remote attack using software-accessible power sensors embedded in the chip.

3.2 Fault injection attacks

Internal energy-based FIAs rely on inducing a modification of the supply voltage and frequency on the target CPU. As described in Section 2.4, abrupt changes in these parameters can result in clock glitches, leading to faults in processed data or instructions. Typically, an unstable state where faults occur is reached when the supply voltage is low and the clock frequency is high. Above a certain threshold, faults occur in critical parts of the system, which produces a reset. The gap in which faults occur may be thin, depending on the device. The attacker can induce a glitch by 1) transiently increasing the frequency to an unstable value, known as *overclocking* attacks [86], or 2) reducing the power domain's supply voltage while maintaining a high clock frequency, referred to as *undervolting* attacks [68]. Table 1 provides an overview of internal energy-based FIAs published recently, highlighting their distinctive characteristics. In this section, we provide an overview of these attacks and their limitations, starting with attacks that maliciously exploit DVFS mechanisms and then addressing the specific case of FPGA-to-CPU undervolting attacks.

Table 1. Overview of FIAs exploiting software-accessible energy management mechanisms.

Attack	**Target platform (Target TEE)**	**Compromised security properties (assets)**
CLKSCREW [86]	Qualcomm Krait (Arm TrustZone)	Confidentiality (extract AES keys from the TEE), Integrity (fault an RSA key during verification), Authenticity (load a self-signed application into TrustZone)
VoltJockey [67, 68]	Qualcomm Krait (Arm TrustZone) [67] Skylake CPUs (Intel SGX) [68]	Confidentiality (extract AES keys from the TrustZone & SGX), Authenticity (load a self-signed application into TrustZone)
Plundervolt [58]	Skylake CPUs (Intel SGX)	Confidentiality (extract AES & RSA keys from the TEE), Integrity (out-of-Bounds memory access)
V0ltpwn [37]	Skylake CPUs (Intel SGX	Integrity (fault SHA-256)
Noubir *et al.* [61]	Exynos 5422 & Kirin 960	Availability (denial-of-service attack)
FPGA-to-CPU undervolting attacks [24, 48, 49]	Heterogenous SoCs (CPU and programmable layer)	Integrity (fault multiplications and AES encryption), Confidentiality (recover AES secret keys)

3.2.1 DVFS attacks. DVFS FIAs involve installing a malicious kernel module or driver that accesses the hardware voltage and frequency regulators. By independently adjusting these parameters, the attacker causes the device to operate beyond its specified limits. This type of attack represents an emerging threat first demonstrated in [86]. In this study, attackers overclock the device to induce a transient clock glitch, which is used to compromise the security of Qualcomm Secure Execution Environment (QSEE), an Arm TrustZone-based TEE. Additionally, as discussed in Section 2.2, in most DVFS implementations, each core (or cluster) has its own frequency domain. Attackers exploit this feature to target a specific core (or cluster) without affecting their attack program. Through this method, the authors successfully extract secret AES keys via DFA. They also demonstrate loading a self-signed application into the TEE by breaking the RSA certificate verification. Subsequent research has shown that transient undervolting can achieve similar results [68] and that Intel SGX can also be compromised through energy management mechanisms by directly manipulating the corresponding MSR [37, 58, 68]. As shown in Table 1, the target platforms encompass a range of devices, from smartphone processors (*e.g.*, Qualcomm Krait) to high-end Intel CPUs. Arm TrustZone and Intel SGX are shown to be vulnerable to FIA. These attacks illustrate various use cases, compromising different security properties of the victim device and targeting diverse assets.

These studies present powerful attack scenarios but also reveal numerous limitations. In DVFS attacks, achieving precise timing for glitch injection is not always feasible, depending on how quickly the victim device can switch between different OPPs. In [86], attackers were able to induce a glitch within a 65 000-cycle time window, targeting a specific part of the victim program. However, in Intel architectures examined in [37, 58], over 500 000 instructions are executed between consecutive OPP change requests, making transient glitch injection impractical. Attackers exploit the fact that certain instructions, such as multiplications [58] and vector operations [37], are more likely to cause faults, allowing them to target specific parts of a program to some extent. When the victim device does not exhibit such behaviour, supply voltage manipulation can still be employed for denial-of-service attacks [61]. Therefore, a short OPP switch delay is favourable for FIAs. Additionally, voltage regulators require significantly more time to change the supply voltage compared to PLLs for changing the operating frequency. Thus, undervolting attacks, while being reportedly more difficult to prevent, offer less transience and precision compared to overclocking.

Secondly, not all SoCs give the OS direct access to voltage and frequency regulators. For devices lacking an open-source OS or detailed technical documentation, the corresponding registers remain unknown. The attacks on Intel SGX [37, 58, 68] were made possible by prior research efforts that uncovered the MSR used for voltage control [58].

Third, each hardware device possesses unique characteristics that can make fault injection via voltage and frequency control challenging, if not impossible. The unstable region that triggers faults without requiring a device reboot or freeze may be very narrow on some CPUs. With coarse-grained voltage control precision (*e.g.*, 5 mV steps), reaching this region may prove unattainable, as observed in AMD Zen processors [70].

3.2.2 FPGA-to-CPU undervolting attacks. Recently, the use of programmable layers such as FPGAs to fault the CPU in heterogeneous platforms has been studied. This follows earlier studies which showed that FPGAs can be used to synthesise power-hungry circuits which provoke significant voltage drops when their activation is toggled at the resonance frequency of the PDN [41]. This vulnerability has mostly been studied in the context of multi-tenant cloud FPGAs for FPGA-to-FPGA attacks. However, the voltage drops affects all components in the same die, notably the CPU. A software attacker can leverage this vulnerability, along with the CPU DVFS mechanisms, to perform an FPGA-to-CPU undervolting fault attack. These software-based FPGA-to-CPU attacks have been studied in [24, 48, 49]. Although these works haven't yet explicitly targeted TEE assets, acting more as proof-of-concept attacks on bare-metal setups, they could be used in future work for remote attacks on TEEs since they target the underlying CPU. In [24], it is shown that this method can achieve precise fault injection, with fault models that were not observed in CPU-to-CPU undervolting attacks, namely instruction skips and data during transfer from DDR memory to caches. It is observed in [48] that FPGA-induced attacks offer substantial improvement in timing precision compared to DVFS attacks. FPGA-to-CPU attacks have been used to successfully setup DFA on an AES encryption occurring on the CPU [24, 48]. Thus, these attacks provide new fault models that could be exploited, together with the previously described DVFS attack, to jeopardize TEE security properties.

3.3 Side-channel attacks

Several studies demonstrate various malicious uses of energy management mechanisms targeting different assets. In the next paragraphs, we briefly describe the primary attack vectors that can be exploited for SCAs.

The first attack vector is a direct reading of energy-related metrics to carry. Attacks exploiting this vulnerability have evolved significantly over the last decade. The first published attacks exploiting this principle targeted unprivileged access to interfaces: battery status on Android phones [21, 52, 66, 94], and power consumption through the `cpufreq` module on Linux-based systems [14]. When accessed by an unprivileged user, these interfaces have a low refresh rate, ranging from ten to hundreds of milliseconds. Despite this low resolution, they can reveal information such as geolocation, website fingerprints, or passwords.

Table 2. Overview of Side-Channel Attacks exploiting software-accessible energy management mechanisms. The resolution column defines the minimum time between 2 consecutive measurements. (P): With privileged access; (NP): With unprivileged access.

Attack	**Target Platform**	**Attack Vector**	**Time Resolution**	**Compromised assets**
Battery level reading attacks [21, 52, 66, 94]	Android smartphones	Unprivileged battery level reading	100 ms to 175 ms	Geo-localisation, running applications, website fingerprints.
DF-SCA [14]	Linux-based systems (x86 and Arm)	Unprivileged use of the `cpufreq` module	10 ms	Website fingerprints, keystrokes.
Attacks on RAPL counters [9, 46, 51]	x86 CPUs	Energy consumption reading using RAPL counters	50 µs (P) - 1 ms (NP)	AES keys from an SGX-enclaved program, Kernel Address Space Layout Randomization (KASLR) addresses.
Frequency throttling attacks [47, 88]	x86 CPUs, Arm SoCs, GPUs	Manipulation and analysis of the frequency throttling mechanism	—	Cryptographic keys (*e.g.* AES) from enclaved programs, KASLR addresses, pixel sniffing.
Collide+Power [40]	Any	Analysis of the power variation during replacement of attacker-controlled data by victim data, using direct measurement or frequency throttling.	—	Any value in a co-located memory (*e.g.* caches).

More recently, some studies describe how a privileged attacker can access energy consumption metrics, notably the RAPL interface on x86 systems [46, 51]. Privileged attackers benefit from a significantly higher refresh rate, with metrics being updated every millisecond or less. This enables more powerful attack scenarios, notably targeting TEE assets: attackers are able to retrieve secret AES keys from trusted programs.

The second attack vector is explored in [47, 84, 88, 89], where the attacker analyzes the behavior of frequency throttling automatically triggered under certain conditions to maintain a balance between power, frequency, and temperature. These frequency adjustments are shown to be both instruction- and data-dependent [47]. An attacker can trigger them by subjecting the processor to heavy workloads. Moreover, a privileged user can manipulate the power budget threshold at which throttling occurs, facilitating the exploitation of this vulnerability. This methodology has been employed to compromise AES encryption within an SGX enclave [47]. This vulnerability also extends to Arm SoCs, as demonstrated in [84], potentially making it exploitable against TrustZone as well. It is noteworthy that program execution time is directly proportional to clock frequency. Therefore, in attacks where clock frequency serves as the side channel, attackers can measure the execution time of a dummy program instead of directly accessing frequency metrics, as demonstrated in [88]. This renders frequency throttling attacks difficult to detect and prevent.

The third attack vector is used in [98]. It supposes that the attacker can access an FPGA that is using the same PDN as the victim CPU. Power consumption variations across the PDN are mirrored as transient supply voltage drops for the FPGA, which induces delays in its combinational logic. Thus, the attacker can load specific circuits in the FPGA such as time-delay lines, and use them as power monitors for the rest of the system. This method allows them to monitor the CPU's power consumption variations with higher temporal resolution than the previously mentioned methods. In [98], it is shown that an attacker can break constant-time implementations of RSA keys using this technique. The potential use of this attack vector in other scenarios requiring greater precision is still to be determined in future work.

Finally, in [40], Kogler *et al.* demonstrate that these software-based power SCAs can reveal arbitrary values in shared memory components, such as caches. They observe that power consumption subtly varies when an attacker-controlled value is replaced by a victim program's value. These variations depend on the Hamming distance between the two values. By using two inverted attacker values and analyzing the impact of their eviction on power consumption, they amplify this subtle leakage signal. With this method, an attacker can leak single-bit differences in the victim data. This attack is CPU-agnostic and can be executed with either direct measurement or frequency throttling to monitor power consumption.

Table 2 provides an overview of the software SCAs based on energy management mechanisms outlined in this section. It is evident that despite relying on low-resolution measurements, some attacks can still infer sensitive information. Moreover, internal energy-based SCAs appear to be more varied than FIAs in terms of potential attack scenarios. They are also evolving rapidly, with increasingly refined attack patterns emerging over the years.

3.4 Covert communication

Several studies have demonstrated that frequency and power management mechanisms can function as covert communication channels between a Trojan and a malicious monitoring program. Following the adversary model presented in Section 3.1, we consider covert communication in the context of a TEE. These attacks can be used to transfer (sensitive) information from an attacker-controlled *sender* program running within the TEE and a *receiver* program running outside the TEE (*i.e.*, within the REE) without any other component detecting this conversation. The sender program may be a self-signed application launched in the TEE using an FIA, as detailed in Section 3.2, or exploiting a vulnerability such as a backdoor. This is therefore a more advanced threat model than in the previous sections, where the attacker was confined to the REE.

While some covert-channel attacks found in the literature present an identical or similar attacker model [5], most consider an unprivileged attacker. Since these methods could be applicable to attacks within the TEEs, we still cover these attacks in this section. We limit our analysis to attacks where both, sender and receiver, are running co-located on the same CPU. Although there are other threat models for energy-based covert communication, such as FPGA-to-FPGA frequency covert channels [20], we do not consider them in this article because their applicability to attack TEEs is less obvious and has not been proven.

In power-management-based covert channel attacks, the attacker hijacks power management modules to establish a communication method based on frequency modulation. The sender program can trigger these frequency changes in two main ways. The first involves taking direct control of hardware frequency regulators [1, 7]. The second consists in subjecting the device to various workloads, resulting in voltage or frequency modulation through mechanisms such as Adaptive Voltage Scaling (AVS), DVFS, and current management techniques [28, 38]. Then, the receiver program can measure the frequency either by directly accessing dedicated registers [1, 5] or by measuring the execution time of a fixed set of instructions [28, 54].

Table 3. Overview of covert-channel attacks exploiting software-accessible energy management mechanisms. (P): With privileged access; (NP): With unprivileged access.

Attack	**Target Platform**	**Sender**	**Receiver**	**Maximum Throughput**
Benhanhi *et al.* [5]	Platform-agnostic	(P) Modifies the frequency using configuration registers	(P) Monitors the frequency by directly reading the clock register	$6\,\mathrm{kbit\,s^{-1}}$
Alagappan *et al.* [1], Miedl *et al.* [54], *DVFSSpy* [69]	Platform-agnostic	(NP) Subjects the CPU to different workloads to modulate its frequency	Reads the frequency: • (P) Using dedicated kernel access [1] • (NP) Using a fixed instruction sequence [54] • (NP) Using `cpufreq` [69]	$20\,\mathrm{bit\,s^{-1}}$ [1], $2\,\mathrm{bit\,s^{-1}}$ [54], $28\,\mathrm{bit\,s^{-1}}$ [69]
POWERT Channels [38], *TurboCC* [35]	Platform-agnostic [38], Intel CPUs [35]	(NP) Subjects the CPU to heavy workloads to reduce the receiver's power budget	(NP) Subjects the CPU to heavy workloads and monitors the execution time to assess the power budget	$121\,\mathrm{bit\,s^{-1}}$ [38], $61\,\mathrm{bit\,s^{-1}}$ [35]
IChannels [28]	Intel CPUs	(NP) Executes power-hungry instructions to trigger current management mechanisms	(NP) Executes power-hungry instructions and measures the throttling period	$2.9\,\mathrm{kbit\,s^{-1}}$
Uncore Encore [26]	Intel CPUs	(NP) Creates different amounts of Last-Level Cache (LLC) traffic to trigger the *"uncore"* frequency scaling	(NP) Measures uncore components (*e.g.*, LLC) access delay	$46\,\mathrm{bit\,s^{-1}}$
Bossuet *et al.* [7]	Heterogeneous platforms	(P) Manipulates the frequency of a FPGA using its configuration registers	(P) Reads the modified frequency using a sensor implemented in the FPGA	$1.3\,\mathrm{kbit\,s^{-1}}$ (CPU-to-CPU)
Yue *et al.* [96]	Mobile phones	(NP) Modulates Ethernet traffic (interpacket gap) using frequency scaling	(NP) Monitors the received traffic's interpacket gap	$45\,\mathrm{bit\,s^{-1}}$

Table 3 provides an overview of existing energy-based covert channel attacks. The reported throughput is obtained in the best possible conditions, in particular when the noise caused by the activity of other programs is very low. As such, it should be considered as an indicator of the overall potential of an attack approach, and not as an absolute or unsurpassable value. It is also highly dependent on the characteristics of the targeted platform [38].

Literature endeavors such as Benhanhi's *et al.* [5] show that simple attack schemes, where privileged attackers directly use the platform's frequency configuration registers, can achieve a significant throughput. However, methods that rely on subjecting the processor to varying workloads to influence the CPU frequency are also valuable contributions, showing that most energy management mechanisms can be exploited indirectly to create a covert channel. They prove that automatic balancing of power, frequency and temperature in modern systems can be a fundamental vulnerability.

The first published attacks of this type rely on the automatic adjustment of OPPs according to the processor activity by Dynamic Frequency Scaling (DFS) governors [1, 54]. With the evolution of power management mechanisms, covert channels have been adopting more sophisticated approaches, leading to better results, and illustrating that increasingly complex energy management mechanisms is leading towards more and more vulnerabilities. More specifically, automatic frequency adjustments occur due to the PDN being shared across all SoC components. Having limited resources, the PDN has to allocate them to the various programs running in the CPU while taking into account parameters such as the temperature, power consumption and frequency. Therefore, the activity of one component impacts the amount of power allocated to another [38]. On the CPU, these restrictions take the form of frequency throttling. When several programs exhibit simultaneous high activity, power management circuits proceed to an automatic reduction of their frequency. This mechanism has also been exploited in SCAs, as described in Section 3.3. While DFS automatically adjusts frequency with a period of 10 ms at best in published work [69], these throttling mechanisms can be triggered more than once every millisecond [28, 38], allowing higher throughput. Moreover, in [28], Haj-Yahya *et al.* offer an advanced characterisation of current management mechanisms in recent Intel processors. As Table 3, this extensive characterisation allows to develop a methodology for covert communication that can achieve a throughput orders of magnitude greater than previous work that rely on similar vulnerabilities. This shows that energy-based covert communication attacks can be significantly improved by thoroughly studying exploitable power management mechanisms and adapting the attack scheme to the platform under consideration.

In addition to maximum throughput, stealth is an important goal in covert communications attacks. Power management mechanisms can be leveraged in creative ways to design stealthy communication schemes that are difficult to detect. In [96], Yue *et al.* observe that DVFS-induced frequency adjustments influence the network traffic of Android devices, with lower frequencies leading to lower inter-packet sizes in Ethernet communications. Authors leverage this side-effect to design a covert-channel over network communications using DVFS. In [7], Bossuet *et al.* show that on heterogeneous platforms, programmable logic can serve as a proxy for covert communication between two CPU programs. With this method, the attacker manipulates the frequency of an FPGA rather than the frequency of his own processor cluster. This makes the attack potentially more difficult to detect, in addition to achieving high throughput due to the high capacity of frequency sensors implemented in FPGAs.

In summary, the exploitation of power management mechanisms is a new and promising vector for covert communication attacks. Although other attack vectors for covert communication, such as shared cache [74], can achieve a throughput several orders of magnitude higher than the energy-based attacks listed in Table 3, the literature on energy-based covert communications shows increasingly sophisticated attack schemes [7, 28] and creative ways of using power management to design stealthy attacks [96]. Energy-based covert channel attacks remain therefore a promising and exciting area of research and are likely to continue to improve in the future.

4 Countermeasures

As observed in Section 3, internal energy-based attacks, which encompass FIAs, SCAs, and covert communication, are potent, can be executed remotely, and are cost-effective as they do not require specialized equipment. Therefore, they pose a substantial threat to all commercial TEE designs.

Despite the numerous demonstrations of these emerging attacks recently, many innovative TEE designs have yet to incorporate protections against them. This lack of protection is due to such vulnerabilities being perceived as hardware flaws [3, 10] or because their mitigation is believed to be unrelated to TEE protection mechanisms [45, 77]. However, internal energy-based attacks do not exploit flaws in a single hardware block that can be remedied by strengthening its design. Instead, they capitalize on a fundamental vulnerability in the TEE protection model: hardware regulators, accessible by the REE without protection, inherently serve as a side channel. This side channel is particularly potent, yielding results comparable to those of energy and clock-based physical attacks, which are well-recognized and considered significant threats. Nonetheless, internal energy-based attacks cannot be classified in the same category as traditional physical attacks regarding the threat model. This is primarily because they do not utilize off-chip equipment but rather leverage components already embedded in the hardware device, thereby circumventing their primary objective of energy optimization. When no countermeasures are in place, any attacker with kernel-level privileges in the untrusted world (*e.g.*, runs in the REE), assumed in most TEE adversary models, can execute this type of attack against an application running within the trusted world. Consequently, addressing this vulnerability at its root necessitates designing TEE software and hardware mechanisms to either eliminate or render this side channel unusable for potential attackers.

Nevertheless, the most widely used TEE implementations, Intel SGX and Arm TrustZone, embed mitigation against the FIA and some SCAs outlined in Section 3, although they hinder the use of power management mechanisms. Moreover, several studies have proposed proof-of-concept countermeasures against energy-based FIAs [39, 56, 97]. These proof-of-concept studies explore various approaches to mitigate energy-based attacks but have not yet been implemented in real-world TEEs. Given the similarity of these remote energy-based attacks to traditional hardware attacks, well-known methods such as masking against SCAs and redundancy against FIAs can also be used to counter them [87]. All of these avenues offer potential inspiration for next-generation TEE designs to incorporate tailored countermeasures to protect against energy-based attacks.

In this section, we provide an overview and insights into existing approaches to counter internal energy-based attacks. First, we classify the countermeasures into three categories and provide the criteria we use to evaluate their potential in Section 4.1. We then examine existing implementations and proof-of-concept countermeasures against CPU-to-CPU energy-based fault attacks in Section 4.2, SCAs in Section 4.3 and covert-channel attacks in Section 4.4, highlighting the limits of each approach. We discuss the specific case of FPGA-to-CPU attacks in Section 4.5. Finally, we give a brief conclusions on countermeasures against internal energy-based attacks in Section 4.6.

4.1 Countermeasures classification

Our classification of countermeasures against energy-based attacks is based on the main components involved in these attacks and their interrelations. The attack path is similar for both FIAs and SCAs. Attacks are facilitated by three vulnerabilities represented in Figure 4. To prevent energy-based attacks (both FIAs and SCAs), a countermeasure must mitigate at least one of them. Based on this observation, countermeasures can be classified into one of three classes, each corresponding to a vulnerability.

(A): Attackers, using their root privilege in the REE, can exploit power management hardware; whether to manipulate the supply voltage and frequency in an FIA or to monitor energy-related metrics in an SCA. This includes accessing software power management interfaces, or using devious means such as frequency throttling mechanisms and FPGAs to sense power consumption or to induce a voltage drop in the CPU. We define countermeasures as belonging to class **(A)** if their approach denies the attacker access to the power management hardware. Existing countermeasures in this category usually operate at software or firmware level.

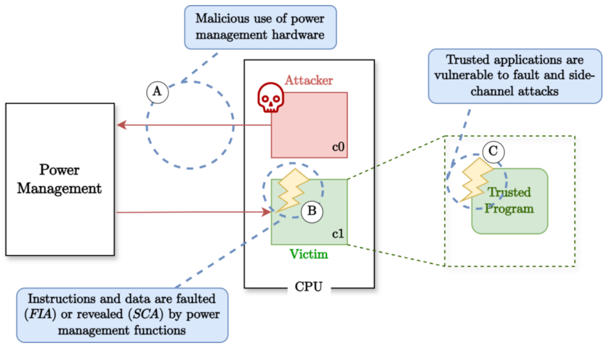
Malicious use of power management hardware
A
Attacker
c0
Power Management
B
c1
Victim
CPU
Trusted applications are vulnerable to fault and side-channel attacks
C
Trusted Program
Instructions and data are faulted (*FIA*) or revealed (*SCA*) by power management functions

Fig. 4. Main steps and components involved in internal energy-based attacks.

(B): Power management mechanisms are directly linked to the target processor. First, they can provoke glitches if they don't supply the correct voltage and frequency. Second, the energy-related metrics that they collect (*e.g.*, real-time power consumption, temperature) reveal information on which data are currently being processed, and how. Indeed, power consumption is data- and instruction-dependent. We define countermeasures as belonging to class **(B)** if they aim to prevent power management mechanisms from damaging the victim processor, *i.e.*, by preventing faults due to incorrect voltage and frequency supply and stopping data and instructions from leaking through the power side channel. This can be achieved either by modifying power management mechanisms or by hardening the processor. Most of these countermeasures require hardware modifications.

(C): The victim's trusted program is sensitive to what happens on the underlying processor. If the processor faults, the program will encounter unexpected behaviour. Plus, its algorithm and secret values are processed by the leaky processor. We define countermeasures as belonging to class **(C)** if their objective is to harden trusted programs against the considered attack, for instance, by making them fault-tolerant or by designing a constant-power implementation of their algorithm. Therefore, this category of solutions operates at the level of the sensitive application.

In the following sections, we use **(F)** to designate countermeasures approaches against energy-based fault attacks and **(S)** for those against side-channel attacks; followed by their class **(A)**, **(B)**, or **(C)**.

Criteria for assessing a countermeasure's potential. Besides the classification based on the previous paragraph, we use the following in our analysis to evaluate the potential of a countermeasure. The first criterion is the complexity of implementing and deploying the proposed solution. Depending on the chosen approach, a countermeasure can be more or less difficult to implement on the considered systems. Software solutions are practical in that regard because they can be deployed on existing devices. Meanwhile, countermeasures that require heavy hardware modifications are more suited for devices that have a security-oriented design, such as embedded systems used in critical applications.

A second criterion is the effectiveness of the countermeasure in negating the considered attacks. Indeed, while some solutions deal with the root cause of the vulnerability, effectively preventing the attack from happening, others focus on mitigating the consequences of an attack, or on reducing the attack's efficiency. Plus, it is important to note that some countermeasures have not been fully implemented, or have not been tested against actual attacks, making it difficult to guarantee and to evaluate their efficiency in practice.

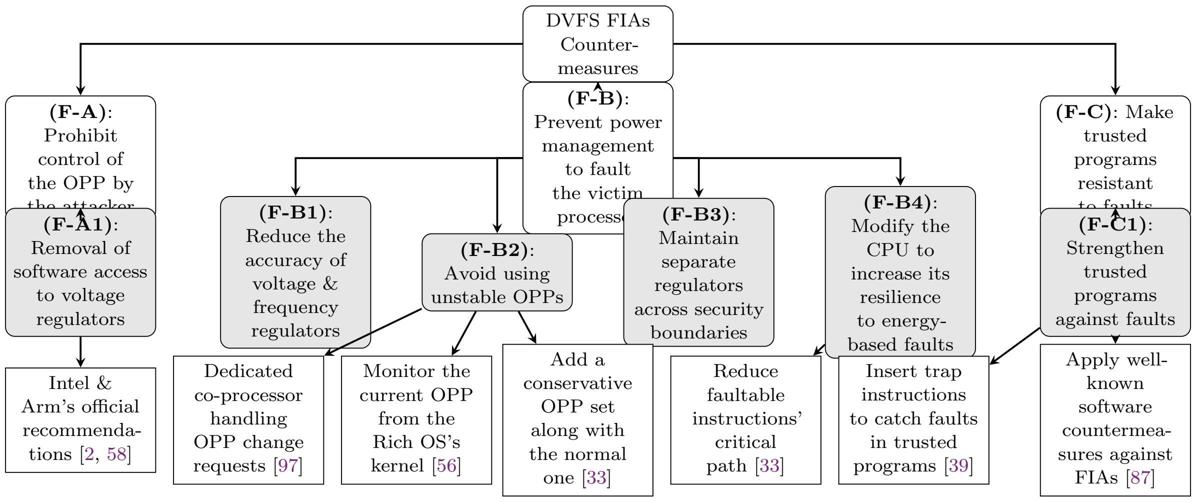
DVFS FIAs Counter-measures
(F-A): Prohibit control of the OPP by the attacker
(F-B): Prevent power management to fault the victim process
(F-C): Make trusted programs resistant to faults
(F-A1): Removal of software access to voltage regulators
(F-B1): Reduce the accuracy of voltage & frequency regulators
(F-B2): Avoid using unstable OPPs
(F-B3): Maintain separate regulators across security boundaries
(F-B4): Modify the CPU to increase its resilience to energy-based faults
(F-C1): Strengthen trusted programs against faults
Intel & Arm's official recommendations [2, 58]
Dedicated co-processor handling OPP change requests [97]
Monitor the current OPP from the Rich OS's kernel [56]
Add a conservative OPP set along with the normal one [33]
Reduce faultable instructions' critical path [33]
Insert trap instructions to catch faults in trusted programs [39]
Apply well-known software countermeasures against FIAs [87]

Fig. 5. Existing and potential approaches for countermeasures against DVFS FIAs.

Finally, the overhead required by the countermeasure needs to be considered. Countermeasures may affect the efficiency of the system, causing performance overhead for some programs, increasing their code size, and or the system power consumption.

In the following sections, we describe in depth the existing countermeasures against energy-based attacks, classify them and compare them based on the aforementioned criteria. First, we delve into countermeasures against FIAs, and second we study countermeasures against side-channel and covert-channel attacks.

4.2 FIAs countermeasures

The existing literature has proposed many approaches to counter or mitigate energy-based FIAs. Some of them have been implemented in commercial TEEs, or in proof-of-concepts described in publications and patents. These countermeasures greatly vary in their objectives, effectiveness, implementation abstraction level and overhead. They can consist of modifications in the device's software, hardware, or a combination of both. Given this great variety of countermeasures, we propose to summarize, analyse and compare them.

In addition, Figure 5 gives a global overview of existing countermeasures approaches and their implementation based on the classification given in Section 4.1. This graph shows that, while several countermeasures implementations rely on a similar principle, some approaches are still unexplored. In the next paragraphs, we describe them more in-depth and compare their objectives and their potential.

(F-A): Prohibit control of power management by the attacker

***(F-A1)**: Removing software access to regulators.*

This approach consists in disabling any software access to voltage regulators, even from the privileged rich OS. Indeed, to execute a FIA as described in Section 3, the attacker loads a malicious kernel module that directly accesses hardware regulators. Therefore, this solution disarms the attacker and effectively prevents DVFS FIAs. This countermeasure is recommended by manufacturers for TEE implementations that have been targeted in this manner [2, 58]. More precisely, Arm recommends prohibiting the kernel from having *direct and independent control of clock and voltage* and suggests performing regular checks on the requested OPP using a trusted entity such as firmware [2]. To our knowledge, no public document indicates that vendors (*e.g.*, Qualcomm, Samsung, etc.) have implemented this countermeasure in their TEE. Furthermore, in response to undervolting attacks on SGX, Intel has disabled access to the voltage control MSR when SGX is enabled. This solely requires a firmware update, which is included in the TCB attestation. An enclave can request this attestation to verify the effectiveness of the update. Thus, this solution is simple to deploy and highly practical for massively used TEEs.

However, this countermeasure runs counter to the primary objective of power optimization techniques. We assert that it may not be sustainable in the long term. Indeed, with this countermeasure, voltage management cannot be performed by the Rich OS and must be delegated to another component, such as hardware power management subsystems. To have no software control over voltage scaling implies that the system can only use the few OPPs defined by the manufacturer. This vendor-prescribed OPPs may diverge significantly from a device's actual operating limits [97]. It also precludes the ability to adjust them dynamically for precise control over the device's power consumption. In [33], the effects of undervolting the device under these vendor-prescribed OPPs (while still staying within the operating limits) are studied. It is shown that this slight undervolting can bring significant improvements: 11.0 % efficiency gain, up to 20.8 % if the software's energy awareness is improved as well.

In summary, this solution is simple to implement and fully prevent DVFS-based FIAs. Plus, it does not add any performance overhead to the system. However, it impedes the utilization of DVFS to its fullest extent for enhancing performance, minimizing energy consumption, and optimizing temperature. This is crucial in resource-constrained embedded devices, as well as in servers where effective power and temperature management are vital. Furthermore, it does not address the underlying vulnerability: there may be additional undocumented methods to manipulate voltage and frequency from the kernel [58], including manipulating FPGAs to cause voltage drops on heterogeneous platforms [49]. As a result, other countermeasures have been explored in the literature.

(F-B): Prevent power management mechanisms to fault the target processor

(F-B1)*: Reduce the accuracy of regulators.*

As stated in Section 3.2, the accuracy of voltage and frequency regulators directly affects the attacker's control over the induced glitch. First, if the voltage and frequency steps are too large, then the attacker may not be able to find an unstable operating area that would trigger a fault. Second, if delays required to switch the voltage and frequency from one step to another are too long, then the attacker loses precision over the fault, *i.e.*, loses the ability to target a specific part of the victim program. The impact of tweaking these parameters on the feasibility of DVFS FIAs has not yet been studied. On the downside, this prevents energy-aware designers from using undervolting and similar techniques to their full potential for optimising efficiency. In addition, it goes against the general trend in the development of power management mechanisms towards faster and more precise control of the device's supply voltage and frequency.

(F-B2)*: Enforce operating limits to avoid using unstable OPPs.*

FIAs exploiting power management mechanisms rely on glitches occurring when the processor operates beyond its specified operating limits. Therefore, enforcing these limits in hardware would render such attacks impractical. However, implementing this in reality may pose challenges. Firstly, determining the actual operating limits of a device is only feasible after a post-manufacturing testing phase, making it practically difficult to enforce hardware-enforced hard limits [86]. Secondly, various factors contribute to the fact that even two similar devices manufactured on the same wafer may have different limits, such as variations and uncertainties during the manufacturing process, temperature fluctuations, and ageing effects. As a result, either a broad security margin must be applied, which restricts the optimal utilization of DVFS, or the limits must be dynamically updated.

Therefore, this approach is more complex than **(F-A1)**, which disables software control over the OPP altogether. However, it is less detrimental to energy efficiency, as long as the enforced range is less restrictive than the OPPs defined by the manufacturer. This range must strike a balance between safety (by being far enough away from the operating limits to prevent faults) and energy efficiency (by being as close to the limits as possible, allowing the OS to optimize energy consumption through undervolting). The performance overhead and ease of implementation of the countermeasures under consideration are also important factors to consider.

Several works propose countermeasure implementations that follow this approach. They first make a characterization of the device's stability depending on their supply voltage and frequency, by submitting it to various OPPs. This characterization is then used to determine a reasonable range, which is then enforced by software or hardware means.

In [56], a kernel module polls the voltage and frequency registers to ensure that they do not cross the stability threshold. This proposition is less restrictive on DVFS than Intel's approach, allowing undervolting to some extent. The stability threshold is defined as the highest voltage for which there is a frequency at which a fault can occur. The module prohibits any voltage below this threshold, regardless of frequency. As a result, it is more restrictive on DVFS than approaches that define the limit on both voltage and frequency [33, 97]. Although this new kernel module can be unloaded by an attacker who controls the rich OS, an attestation can be used to verify its integrity. This approach is purely software and simple to implement. Plus, the authors observe that it introduces a low-performance overhead: on average 0.28 % performance loss over the `SPEC2017` benchmarks. However, the authors do not indicate how to dynamically update the threshold to account for the effects of ageing. Besides, it is unclear whether this countermeasure fully prevents DVFS FIAs. For instance, a fault could strike between two polls made by the kernel module.

In [97], a hardware-level approach is proposed. This appears to make the countermeasure robust, as it is unlikely that an attacker could compromise the integrity of a separate hardware component. It relies on a lightweight machine-learning model, trained based on the previously mentioned device characterization. This model is used to predict whether an OPP is safe to use or not. It runs on dedicated *blacklist-core* coprocessors, one per CPU core, independent of both the CPU and the PMIC. The coprocessors intercept OPP change requests from the CPU and only apply them if they are predicted to be stable. To account for the effects of temperature and ageing, the restrictiveness of the blacklist core is dynamically controlled by a decision boundary. It is updated when a fault occurs during runtime. The implementation of this countermeasure requires hardware changes, *i.e.*, to add blacklist cores on the SoC and to route calls to the PMIC to them. It also requires software changes in the TEE trusted code, *e.g.*, to implement a behaviour for the decision boundary. Deploying this countermeasure is therefore more difficult than purely software solutions, as it requires the involvement of various actors, from the hardware integrator to the software developer. Regarding performance, the authors point out that the blacklist-core adds a negligible overhead to frequency and voltage change requests. One of these cores consumes 10.5 mW at 250 MHz. This static overhead in the device's energy performance should be weighed against the benefits of this countermeasure compared to simply restricting the usable operating points to a vendor-defined discrete state, using Intel *SpeedShift* technology, for instance [78].

In [33], Juffinger *et al.* present a hardware-software co-design introducing a second set of OPPs to the device. This supplementary set offers more aggressive energy savings compared to the original one, featuring lower supply voltages for the same operating frequencies while maintaining high performance. It is based on the observation that some instructions, *e.g.*, multiplications and vector operations, are more likely to fault than others when x86 systems are submitted to undervolting. The OS is modified to switch to the conservative OPP set when executing such instructions. The authors of this work point out that it is not primarily intended as a countermeasure against DVFS attacks, but as a tool to safely improve energy efficiency. Nevertheless, we believe that this approach could potentially be used in an actual countermeasure with a few modifications. It can be implemented solely by software changes. The main implementation challenge is to choose the component responsible for the OPP curve: it cannot be the attacker-controlled kernel, but it must control each instruction to account for its faultiness. Hardware-controlled OPP curve changes, as in Intel's *SpeedShift* technology, could also be explored. Another challenge is that such a countermeasure could degrade the performance of the system, which has to stall during each OPP curve switch. A voltage change can take up to a millisecond on some devices. Juffinger *et al.*, propose to mitigate this risk by introducing a time limit, which prevents curve switches from being too frequent. They also note that this method gives better energy savings when programs are optimized to avoid using faultable instructions.

***(F-B3)**: Maintain separate hardware regulators.*

Another approach is to use separate regulators so that they are not shared between trusted and untrusted programs, depending on whether a CPU core is in a secure or non-secure state [86]. Implementing such a solution poses several challenges. First, maintaining multiple physical regulators per core may incur significant costs. Second, at the software level, access to the regulators must be restricted based on the core and execution environment of the program issuing the request. This necessitates a comprehensive power management solution across different software layers in the trusted world, which could introduce performance overhead and enlarge the size of the TCB. Reserving one or several CPU cores exclusively for trusted programs, equipped with dedicated hardware regulators, would simplify the implementation of this countermeasure, as suggested in [67]. However, this approach would essentially resemble using a TPM or a coprocessor. As demonstrated in Section 2.1, this contradicts the goals of TEEs, which strive to offer isolation while utilizing the same hardware resources for both trusted and untrusted programs.

Overall, this approach to prevent energy-based FIAs poses significant challenges to implement while respecting the TEE design philosophy. Meanwhile, it doesn't offer any obvious advantages over the **(F-B2)** approach, for which some implementations seem capable of countering the vast majority, if not all, DVFS fault attacks. Furthermore, the design of a proof-of-concept demonstrating the feasibility and effectiveness of this approach remains an open challenge: to the best of our knowledge, no publicly available work has done so.

***(F-B4)**: Increase the resilience of the processor to power-management-induced faults.*

As shown in Section 2.4, clock glitches and voltage glitches occur because the processor instructions' critical paths get too large compared to the clock frequency. Therefore, relaxing some instructions' critical paths can prevent such glitches from occurring. This method is explored in [33], in which, as mentioned in **(F-B2)**, a second set of OPPs is added along the default, conservative one. On x86 systems, undervolting provokes specific types of instructions to fault, notably multiplications, which are very frequently used in many types of programs. To avoid having to switch to the conservative OPP curve at each multiplication, Juffinger *et al.*, propose to make multiplication resilient to undervolting faults instead, by relaxing their critical path. As a result, each multiplication takes one additional clock cycle to complete. This does not significantly affect performance because of instruction pipelining. This approach involves heavy hardware modifications, directly into the CPU's arithmetic logic units. Therefore, it would be impractical to apply it to every instruction. However, as demonstrated in [33], it can be an efficient tool when combined with other approaches.

Other possibilities, not limited to DVFS FIAs, can be derived from existing hardware countermeasures against faults in general. For instance, adding redundancy to operations can detect and mitigate all types of faults. At the hardware level, there are many possibilities to implement this. For instance, by duplicating the entire instruction stream. Alternatively, methods based on control-flow integrity enforcement can be explored [91]. Moreover, leveraging the RISC-V ISA opens up opportunities for hardware and cross-layer countermeasures that operate at both, the compiler and hardware levels [44, 53, 95]. As of now, there have been no published attempts to utilize these methods to mitigate internal energy-based attacks. However, they can be used to protect the device against them, as they do for other types of FIAs. This type of approach requires heavy modifications and may induce significant overhead. They are suitable for devices with high-security requirements that need to protect against a variety of threats, including but not limited to energy-based attacks.

(F-C): Harden trusted programs and hardware against faults

Countermeasures outlined in the preceding paragraphs effectively prevent attackers from injecting faults into trusted programs, thereby thwarting all known DVFS FIAs. However, in this paragraph, we delve into alternative approaches aimed at securing trusted programs against these attacks. These strategies involve modifying trusted programs themselves to render them resilient to faults. While these countermeasures do not preempt DVFS attacks, they either detect faults upon occurrence and rectify their effects or render faults impractical for exploitation by attackers. Although these countermeasures typically introduce more performance overhead than the aforementioned approaches, this overhead is confined to TEE programs, with little impact on the rest of the system. Another notable distinction is that this type of countermeasure assumes that developers of trusted software bear the responsibility for safeguarding their programs from potential attackers. Consequently, they are only relevant in cases where TEE vendors have not provided lower-level mitigation against FIAs from the outset.

A countermeasure proposed in [39] detects undervolting FIAs using a compiler extension. It involves inserting *trap* instructions into the victim code. As mentioned in Section 3, undervolting Intel processors increases the likelihood of certain instructions being faulted (multiplications, vector operations). However, this countermeasure introduces significant overhead in terms of both execution time and code size of the protected program. For instance, in the tested scenarios, it resulted in a performance overhead of 148.4 % and a code size overhead ranging from 50 % to 150 % to mitigate 99 % of DVFS FIAs. Additionally, this approach is tailored specifically for undervolting-based attacks on x86 Intel platforms and may not generalize well to other architectures.

Other methods, that are non-specific to energy-based attacks, can be used to secure trusted programs. Indeed, since FIAs have been studied for decades, many well-known software techniques can be used to secure programs against faults in general. Redundancy and error detection codes, such as parity checks, are commonly utilized to both detect and mitigate the effects of faults. This can be achieved through software means by duplicating instructions at compilation, manually implementing redundancy checks in the code software, or even duplicating the entire encryption or decryption process, although the latter incurs significant overheads [82]. Another approach is a software implementation of infection, wherein a completely erroneous cypher is produced in the event of a fault during encryption, rendering the output cypher unusable for attackers. Tao *et al.* explore the potential use of these three countermeasures (parity codes, redundancy-based methods and infectious computation) against DVFS FIAs targeting AES encryption in [87]. They determine that the most cost-effective countermeasure, in terms of both performance and code size overheads, is temporal redundancy (*i.e.*, running the encryption process twice serially), with overheads of 34.18 % and 12.04 % in performance and code size, respectively. However, they do not empirically assess the robustness of this countermeasure against actual overclocking or

Table 4. Comparison of published countermeasures against DVFS fault attacks.
[1]NE: Not evaluated against a practical implementation of an attack.

Countermeasure	**Implementation**	**Attack Detection / Prevention Rate**	**Undervolting Allowed**	**Overhead**
Removal of software access to voltage regulators [2, 58]	Software (microcode update)	100 %, may leave other possible attack paths [58]	□	None (only removes software access to a register)
Monitor the OPP currently in use [56]	Software (REE kernel module)	NE[1]	◪	0.28 % global performance overhead
Insert trap instructions to catch faults [39]	Software (Compiler extensions for TAs)	>99 %, no false positives	■	150 % on performance and +50-150 % code size for TAs
Temporal redundancy for cryptographic software [87]	Software (Algorithmic modifications on cryptographic TAs)	NE[1]	■	34 % on performance, +12 % code size, +128 bit bits on the data structure (on AES-128)
Prevent the transmission of unstable OPPs to the PMIC [97]	Hardware (co-processor)	99.69 %, with 0.92 % false positives	■	+10.5 mW per CPU core, +1 µs latency on each OPP change
Use a conservative OPP set alongside an efficient one [33]	Hardware (modifications on the CPU)	NE[1]	■	3.8 % performance loss on average

undervolting attacks. In [29], Huang *et al.*, propose to substitute the most *faultable* instructions with *safer* ones. Some instructions cannot be replaced, and for some others, the substitutes do not provide satisfying protection against DVFS FIAs (the fault rate is reduced by about half or less).

In summary, countermeasures that modify trusted code to make it resilient against faults have the advantages of being easy to deploy and of leaving the untrusted part of the system unaltered. However, they usually add a lot of overhead to the protected code in terms of performance and code size. Besides, they do not prevent attacks from happening, but only mitigate their consequences. In that regard, other solutions **(F-A)** and **(F-B)** that deal with the root cause of the vulnerability will be more efficient most of the time. Thus, this type of solution seems more suitable when the system and the TEE do not offer protection against energy-based FIAs. In that sense, it highlights the necessity of native, embedded countermeasures that deal with the vulnerability at its root.

Conclusion on existing countermeasures against DVFS fault attacks.

In the previous paragraphs, we investigated potential approaches to counter DVFS attacks, highlighting how they differ in their objectives and methods. Table 4 summarizes countermeasure implementations proposed in the literature [33, 39, 56, 87, 97], as well as the one implemented in Intel processors and recommended by Arm [2, 58]. They are compared according to the criteria presented in Section 4.1, in addition to another criterion specific to the DVFS fault attack countermeasure: whether or not it allows undervolting. Indeed, the main drawback of the countermeasure implemented by Intel [58] is that it prohibits the operating system from lowering the supply voltage beyond restrictive vendor-defined OPPs, resulting in energy loss. The use of undervolting can lead to energy efficiency improvements of up to

20 % [33]. Countermeasures proposed in the literature attempt to solve this problem by protecting devices without limiting DVFS capabilities. Most don't restrict usable OPPs at all [39, 87], or come as close as possible to the operational limits for voltage and frequency [33, 97]. The countermeasure proposed in [56], despite allowing some undervolting, still restricts the set of OPPs that can be used, by defining the limit solely in terms of frequency, irrespective of voltage. It is therefore an intermediate solution regarding this criterion.

Existing countermeasures vary in their impact on the system. Software approaches tend to either limit power management mechanisms [58] or add significant performance overhead to TAs [39, 87]. Meanwhile, existing proposals for hardware countermeasures are more focused on improving the energy efficiency of the platform while protecting the CPU from undervolting-induced faults [33, 97]. Therefore, the optimal countermeasure for a given platform depends on which aspects of the system can be compromised to provide protection against these attacks.

4.3 Side-channel attacks countermeasures

Unlike FIAs, SCAs are passive and thus more challenging to detect and thwart. In the following, the existing approaches to counter internal energy-based SCAs are studied according to the classification given in Section 4.1. This is represented, as well as existing implementations of these approaches, in Figure 6. It can be observed that compared to energy-based FIAs, few countermeasures have been proposed and implemented for SCAs. Yet, many more SCAs attack scenarios have been demonstrated. This gap in the literature suggests that countermeasures for internal energy-based SCAs is a promising avenue for future research.

(S-A): Prohibit malicious access to power management interfaces

To counter the first category of side-channel attacks, based on direct reading of the power consumption or battery level, these interfaces may simply be made inaccessible to all untrusted programs. However, said interfaces are extensively used by energy-aware programs and frameworks. Thus, simply deactivating them would be hurtful to the energy efficiency of the system. We advocate for the use of less restrictive countermeasures such as those presented in **(S-B)** and **(S-C)**.

For the other category of energy-management-related SCAs, which employs frequency throttling as the side channel [47, 84, 88, 89], attackers can facilitate the attack by altering the power or temperature threshold at which throttling occurs [47]. This can be effectively mitigated using TEEs by restricting access to corresponding configuration registers or files to trusted programs only. Such restriction compels attackers to increase the system workload to induce throttling, which is less practical. However, this comes in contradiction with one of the main reasons for which this throttling mechanism exists in the first place: power clamping, *i.e.*, allowing the user to define an energy budget over a given period of time [71].

We also point out that power management mechanisms may be maliciously used in much more ways for SCAs than for FIAs. Creative and unexpected attack vectors can be exploited. For instance, using an embedded programmable hardware component as a precise power sensor [7], or using thermal measurements instead of power traces [55]. Therefore, finding all possible vulnerabilities and forbidding their access may be challenging for SoC designers. There may always be new ways to exploit power management mechanisms.

(S-B): Prevent power management from leaking information on the processed data and instructions

To thwart attacks like *Platypus* [46], which derive information from direct readings of instantaneous power consumption or frequency, several solutions can make the side channel challenging for attackers to

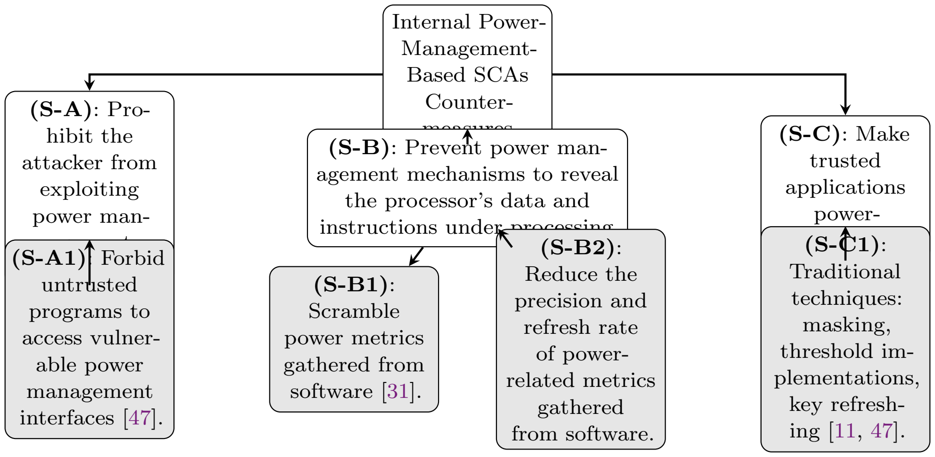
Internal Power-Management-Based SCAs Counter-measures
(S-A): Prohibit the attacker from exploiting power man-
(S-A1): Forbid untrusted programs to access vulnerable power management interfaces [47].
(S-B): Prevent power management mechanisms to reveal the processor's data and instructions under processing
(S-B1): Scramble power metrics gathered from software [31].
(S-B2): Reduce the precision and refresh rate of power-related metrics gathered from software.
(S-C): Make trusted applications power-
(S-C1): Traditional techniques: masking, threshold implementations, key refreshing [11, 47].

Fig. 6. Mind-Map: Existing and potential approaches and implementations for countermeasures against power management based side-channel and covert-channel attacks.

exploit. Intel's approach involves scrambling software-accessible power traces when SGX is enabled [31]. Power traces sent directly to hardware components remain unaffected. The scrambling is achieved by randomizing the values sent to the software-accessible voltage MSRs. This allegedly makes this internal power side-channel unusable for the attacker, but it also induces a variation in the energy reporting of about 5 % to 10 %, which can potentially harm some energy-aware programs which need precise data on the energy consumption. This imprecision is to be put in perspective with the inherent approximation of the estimation-based RAPL energy reporting [27]. However, recent works show that Intel's countermeasure is not sufficient to counter power-based SCAs. In [9], Chowdhurry *et al.*, bypass Intel scrambling countermeasure by combining a power-based SCAs with a *transient execution attack*. This category of attacks uses architectural vulnerabilities to replay the victim program's instructions. Thus, the attackers can gather thousands of power measurements for the targeted instructions. With enough of them, Intel's noise-based countermeasure becomes inefficient.

The same method could potentially be transposed for frequency throttling attacks. Randomly tweaking the operating frequency may prevent the attacker from inferring information from measurements. A similar approach has been proposed against physical power attacks in [81]. With a finite set of frequency OPPs, it still may be possible for the attacker to infer information even when noise is added to the operating frequency.

Another potential approach is to reduce the granularity of software-accessible counters and embedded sensors to hinder energy-based SCAs. However, as shown in Section 3.3, attackers may still infer sensitive information even with a low-resolution channel (*e.g.*, one measurement every 10 ms).

In summary, this type of countermeasures are interesting and leads to mitigating energy-based SCAs. Indeed, they are less restrictive for power management than simply deactivating these interfaces, while being allegedly efficient to prevent this type of attack [27]. They do not totally prevent the attacker from gathering information but offer a statistical-based countermeasure. Therefore, there may still be ways to exploit such scrambled or imprecise data. In addition, the application of these methods for frequency throttling attacks is still to be studied. Finally, this approach has the same drawback as for those of the

(S-A) category: given the variety of ways to exploit power management mechanisms for SCAs, it may be challenging to implement a mitigation for all of them.

(S-C): Prevent trusted programs from revealing information

Traditional algorithm-level countermeasures against power analysis (both physical and remote) can be employed to increase the difficulty of exploiting these attacks. This type of countermeasure concentrates on making the attacks harder to exploit by obscuring results or concealing sensitive information. These solutions encompass masking, threshold implementations [60], and key refreshing, which may be implemented in trusted programs to shield them against both types of attacks [11, 47]. Their potential in the specific case of internal power-management-based SCAs has yet to be assessed. Given that this type of countermeasure requires to modify the algorithms themselves, it can imply some overhead, contrary to previously mentioned approaches. However, it has the advantage of leaving power management mechanisms themselves unaltered, which benefits energy-aware programs. This type of countermeasure also tends to focus solely on cryptographic algorithms, whereas TEEs encompass a much wider variety of programs.

4.4 Covert-channel attacks countermeasures

Covert-Channel attacks are similar to SCAs in the sense that they use devious means as interfaces for communication. In SCA, a victim program inadvertently leaks secrets through this interface, whereas in covert communication, it is voluntarily used to transmit information. Therefore, some of the approaches described in Section 4.3 against SCAs could also be used to thwart covert-channel attacks. Namely, **(S-A)** and **(S-B)**, which aim to prevent side-channels caused by power management in general. Some of these approaches have been mentioned in the literature as potential countermeasures, but have not yet been implemented and evaluated in practice. For example, scrambling frequency metrics by adding noise or reducing their precision [38]. These approaches would however significantly slow down covert communication based on frequency modulation. However, it is worth noting that SCAs and covert channels use different power-related metrics as information vectors. At present, covert communications only use frequency variations, while the most powerful SCAs—for which Intel has implemented scrambling—use energy consumption information as their main side channel [9, 46]. Therefore, while countermeasures for SCAs and covert channels are similar in their approach, they would differ in their implementation, resulting in a different impact on the power management of the device. For example, adding noise to frequency seems significantly more difficult and more impactful than scrambling power consumption measurements collected from software. Indeed, as Table 3 shows, in covert-channel attacks, frequency is often measured not by accessing specific registers whose reported values could be scrambled, but by measuring the execution time of a fixed set of instructions. Consequently, countering these attacks would necessitate random adjustments of the actual operating frequency of the processors, instead of simply scrambling the value reported by a register, which would degrade the performance and predictability of the system.

In addition, the covert-channel attack literature mentions potential approaches for countermeasures that are specific to their attack schemes, thus different than those found for SCAs. The first proposition consists in assigning a separate power budget for each processor core [28, 38]. Indeed, some cross-core covert channels exploit the fact that a single controller manages the total power budget across all cores, correlating the power consumption (*i.e.*, activity) of one core with the power budget of the others [38]. Having per-core power management hardware (voltage regulators, PLLs and management circuit) would prevent such attacks. As discussed in Section 4.2, this countermeasure has also been considered for DVFS fault attacks, and suffers from several issues, mainly its cost. The second proposition consists in implementing a dynamic *Secure Mode* which would act on power management mechanisms when a critical program is running. In [28], Haj-Yahya *et al.* suggest to set to the maximum the voltage guardband, which is the threshold at which frequency throttling is activated. Such a countermeasure would impair cover-channel attacks, and may prove useful against frequency-throttling-based SCAs as well. However, frequency throttling is an important power and thermal management mechanism. Disabling or restricting it would increase the device's power consumption, up to 11 % according to Haj-Yahya *et al.* estimations, depending on the processor's architecture. In addition, throttling also serves a protection mechanism [32]: running the processor at a high voltage or temperature for too long can cause permanent damage. Therefore, this countermeasure should be implemented with caution, taking into account the physical characteristics and limitations of the system.

In conclusion, the literature suggests several possible approaches to counter covert channel attacks. However, they require concrete implementation in order to more accurately estimate their impact on the system energy consumption, and to better understand the technical challenges involved.

4.5 Countermeasures against FPGA-to-CPU attacks

As described in Section 3, several works showcase how an attacker can use an FPGA to exploit power management hardware and attack CPU programs, both through SCAs and fault attacks. These attacks extend from previous research, which focused on FPGA-to-FPGA attacks in a multi-tenant scenarios. To the best of our knowledge, no countermeasure exists specifically for energy-based FPGA-to-CPU attacks. However, several methods have been proposed to counter FPGA-to-FPGA attacks [42, 43, 59, 65]. These countermeasures aim to prevent an attacker from making malicious usage of FPGAs for precise power sensing and for inducing voltage drops. In that regard, they also prevent FPGA-to-CPU attacks.

Two main approaches have been proposed in the literature. The first one consists in analysing bitstreams prior to deploying them on the FPGA to potentially detect malicious circuits [42, 43], as an antivirus would do for software programs. It is designed to forbid both, the deployment of power sensors to carry out an SCA, and power-hungry circuits that can be used to induce a voltage drop. The proposed analyser looks for patterns that are known to be used in such attacks. This includes searching for primitives such as Ring Oscillators or unusual connections between the clock and data paths, among others, and carrying out timing analysis to detect runtime behaviour that would provoke high current variations [42]. This type of countermeasure presents however two main shortcomings: first, it prohibits the use of legitimate Intellectual Propertys (IPs) blocks that use potentially harmful circuits for benign purposes. Second, because it is based on the analysis of existing attacks, it is not effective against unknown attack schemes [59]. Therefore, it can't completely prevent undervolting attacks.

The second approach is to detect voltage drops at runtime and prevent them by disabling the malicious part of the FPGA. The main challenge with these countermeasures is to disable the malicious circuit fast enough to prevent the attack. Contrary to the previous approach, this is helpful only against undervolting attacks and not FPGA-based SCAs. Voltage drop detection can be achieved using voltage fluctuation sensors, similar to the ones used in SCAs. To neutralise the attacker's circuits, methods based on clock gating [65], switching all interconnects to a high impedance state [59] and partial reconfiguration [34] have been studied. At present, such countermeasures can prevent attacks leading to a crash, but not fault attacks, since faults occur faster than crashes during a voltage drop [59]. Although it can't completely prevent them, this method successfully detects FPGA-induced fault attacks, allowing the system to trigger other mitigation after the attack has occurred.

4.6 Summary on existing countermeasures against internal energy-based attacks

As discussed in previous sections, the focus on internal energy-based attacks has predominantly centred on securing processors against DVFS FIAs. However, the solutions adopted by manufacturers, namely restricting direct kernel control over OPP and relying on governor-based DVFS management, limit the optimal utilization of DVFS, resulting in energy inefficiency. Furthermore, the emergence of throttling-based attacks [88] underscores that governor-based DVFS also presents an exploitable side-channel for attackers. This approach fails to address the underlying cause of these attacks: the shared and accessible nature of energy management mechanisms by both the TEE and the REE, inherently constituting a side-channel. Besides, the implementation of separate regulators for the two domains has not been achieved yet and may prove costly. Recent works proposing specific countermeasures against energy-based fault injection attacks [39, 56, 97] offer promising avenues for securing TEEs, albeit highlighting the substantial challenges in terms of overhead and implementation complexity. Additionally, we posit that leveraging features of modern TEE designs, such as the exclusive assignment of hardware peripherals [3] or attestation, could enhance countermeasure effectiveness. Besides, this allows the TEE to stay the sole trusted actor in the system, thereby centralizing the TCB.

Meanwhile, while specific countermeasures against DVFS FIAs proposed to date [39, 97] show promise, they await implementation in actual hardware TEE technologies. Lastly, few published work has proposed countermeasures against energy-based software SCAs and covert-channel attacks beyond hardening the algorithms under attack or restricting access to their interfaces, akin to FIAs. The design and implementation of innovative and efficient countermeasures against power-management-based internal SCAs and covert communication remains an open challenge.

5 Discussion

In the previous sections, we provided an overview of internal energy-based attacks and their countermeasures from the literature perspective. However, to offer a more comprehensive understanding of this emerging research topic, it is necessary to consider these attacks from a broader point of view. Therefore, in this section, we first assess the practical exploitability of the aforementioned attacks, highlighting the main challenges that hinder their reproducibility and, consequently, complicate the accurate evaluation of the associated attack surface. Next, we extend our discussion beyond the sole targets of TEEs and CPUs, in order to reflect on broader implications and identify promising directions for future research. Finally, we summarize the key findings of this article and outline what we consider the most interesting research avenues.

5.1 Exploitability and reproducibility of internal energy-based attacks

Despite the amount of research that has been done on attacks that exploit power management mechanisms, they are difficult to reproduce in practice. Therefore, it may be complex for a manufacturer or certification body to prove and guarantee that a device is resistant against these attacks. In addition, due to the lack of standards for interaction with power management mechanisms, as well as theoretical models modeling their vulnerability according to implementation details, it is difficult to estimate the number of devices on which these attacks are currently exploitable.

Challenges in replicating existing attacks across platforms.

DVFS-induced fault attacks described in Section 3.2 are particularly difficult to reproduce on different platforms. Some work, such as CLKScrew [86] and Plundervolt [58], provides open-source material to replicate the attacks [57, 85], but these resources are tailored to specific target platforms, making them difficult to adapt to others. This limitation arises from the lack of standardization, and often documentation of the interfaces between software and hardware voltage regulators (*e.g.*, MSR `0x150` used in Plundervolt [58]). As a result, the method for accessing voltage regulators from user-level code is highly dependent on the specific platform and hardware vendor.

In addition, several factors related to the hardware implementation of power management mechanisms and PDNs influence the feasibility of undervolting fault injection attacks on CPUs. As these aspects have not been thoroughly investigated, some of them remain poorly understood. For instance, in [70], it is reported that such attacks are not exploitable on AMD Zen processors, despite the absence of documented countermeasures against DVFS-induced faults.[2] While it is technically possible to manipulate the CPU supply voltage in steps of 6.25 mV—similar to the Qualcomm platform used in CLKScrew [86]—no OPP combination has been found that causes faults without leading to a system crash or freeze. As a result, undervolting on these processors cannot be exploited for anything beyond denial-of-service attacks. This limitation may be explained by physical properties, such as the capacitance of the voltage regulators. Further research is needed to better understand how hardware-level DVFS implementations affect the feasibility of fault injection.

Challenges in assessing the global exposure to energy-based attacks.

Countermeasures against DVFS-based fault attacks, such as those implemented by Intel [58] and recommended by Arm [2], should prevent these attacks on recent devices. However, it is difficult to assert with certainty that they have been completely mitigated, as new methods of independently controlling voltage and frequency might still be discovered. By focusing on neutralizing specific attacks (*e.g.*, Plundervolt and CLKScrew) rather than addressing the root cause, vendors may leave other potential attack vectors open.

Consequently, it is currently difficult to estimate how many devices remain vulnerable to DVFS fault attacks, primarily due to the lack of documentation regarding power management hardware and its software interfaces. We advocate for the design and adoption of a standardized framework for software interactions with PMICs. Such a framework would enable manufacturers and certification bodies to evaluate device robustness against DVFS-based attacks using a common testing methodology.

Regarding SCAs that exploit power consumption sensors which, as shown in Section 3.3, constitute the most practical and powerful source of leakage, all devices exposing power-related metrics are potentially vulnerable. However, the nature of the assets at risk varies according to the implementation. As illustrated in Table 2, even low-precision access (*e.g.*, through the Linux `cpufreq` module) may allow sensitive information such as user geolocation or passwords to be inferred [94]. With higher time resolution, attackers may even compromise critical TEE assets such as cryptographic keys [46]. To the best of our knowledge, no existing model predicts the number of samples required to extract sensitive information such as an encryption key as a function of sensor sampling rate and amplitude resolution. Developing such a model would enable a more precise assessment of which assets are at risk, depending on the type of power sensor available on a given platform. This, in turn, would provide a first estimate of the number of potentially vulnerable devices.

5.2 Beyond TEEs and CPUs

TEEs are designed to protect sensitive data and programs from a powerful attacker with full privileges in the REE. However, attacks that exploit power management mechanisms, along with other hardware

[2]To our knowledge, AMD has never communicated publicly about fault attacks via DVFS, and no official documentation indicates the implementation of specific countermeasures in their processors.

vulnerabilities accessible from software, threaten this protection model [22]. Therefore, we advocate for the integration of specific countermeasures against such attacks within TEEs.

Yet, in order to build an effective protection model, the scope of energy-based attacks must be considered beyond the sole boundaries of the TEE. Recent studies show that components other than the CPU may be vulnerable to these attacks and may even serve as attack vectors against the CPU itself. In particular, fault injection via DVFS has been demonstrated on GPUs [72, 92] and on FPGA-based hardware accelerators [83, 93]. Consequently, delegating critical functions from the TEE to hardware accelerators does not inherently guarantee their protection against internal energy-based attacks. Furthermore, FPGA-to-CPU attacks have demonstrated that it is possible to compromise the CPU, and by extension the TEE, from another component in the SoC [49], as long as both components share the same PDN. This highlights the feasibility of cross-component energy-based attacks.

Such attacks can target components in both directions: from CPU to FPGA [92] and from FPGA to CPU [49], including power side-channel attacks [98] and covert channels based on frequency manipulation [7]. Therefore, a TEE limited to protecting the CPU and its data cannot ensure the integrity of the platform as a whole.

To mitigate this issue, two complementary approaches may be considered. The first is to confine critical programs within a tightly controlled subsystem that offers robust protection against energy-based attacks. Recent trends in SoC design suggest the integration of secure subsystems [17], similar to Secure Elements (SEs), dedicated to safeguarding a limited but critical set of assets. These components are often subject to advanced security evaluations, making them more resilient to hardware-level threats than general-purpose CPUs. Evaluating the effectiveness of internal energy-based attacks against such secure subsystems remains an open area for future research.

The second approach involves extending protection mechanisms to all components interconnected through the PDN, beyond the sole TEE. Some TEEs already propagate the processor's security state to peripherals; for example, TrustZone propagates the value of the NS bit over the AXI bus [63]. Exploring such mechanisms to manage power control at the SoC level, leveraging the isolation and control capabilities of TEEs, is a promising research direction. In practice, designing a security-state-aware PMIC controller appears feasible. A patent granted to Intel Corp. [64] illustrates industrial interest in this approach, although it predates the discovery of DVFS-based attacks and was not designed with them in mind. The proposal envisions the TEE as a dynamic policy manager for the PMIC, including the handling of power management interrupts.

Finally, as discussed in Section 4, existing countermeasures against internal energy-based attacks build upon techniques from multiple disciplines, including machine learning [97] and processor architecture [33]. As these attacks originate from vulnerabilities at the software/hardware boundary, effective mitigation strategies should also rely on software/hardware co-design, in the same way that recent TEE architectures address other hardware-level vulnerabilities accessible from software [3, 77].

Moreover, the most powerful internal energy-based attacks often combine power management exploitation with other techniques, such as cache attacks [86] or single-stepping [9]. This suggests that defending against them may benefit from insights and methodologies drawn from adjacent research domains. We argue that such interdisciplinary approaches are essential for the design of robust and comprehensive countermeasures against internal energy-based attacks.

5.3 Perspectives for future research

This article has reviewed various internal energy-based attacks and the countermeasures proposed so far, highlighting several open research directions. This section summarizes key areas where further investigation is required.

1) Countermeasures against SCAs, covert communications, and FPGA-to-CPU fault attacks

As discussed in Section 4, several types of internal energy-based attacks currently lack effective countermeasures. We identify the following priority areas for future research.

Side-channel attacks based on direct access to energy-related metrics.

Against such SCAs, general countermeasures such as masking [11] and scrambling can be applied at the software level. However, complementary strategies are needed. One promising direction is the definition of strict access control policies for power-related sensors, as discussed in Section 4.3. To date, the only known implementation of such a mitigation is the scrambling of RAPL counters in SGX-enabled Intel processors [31], which has nevertheless been bypassed in [9].

Side- and covert-channel attacks exploiting automatic frequency adjustments.

To the best of our knowledge, no published countermeasure exists for side-channel attacks exploiting frequency throttling, nor for covert channels based on similar power management mechanisms such as current-induced frequency modulation [28]. These attacks are recent and particularly challenging to mitigate, as they do not require access to privileged registers but simply rely on observing local frequency variations (*e.g.*, execution time). Moreover, automatic frequency scaling is part of critical mechanisms ensuring the thermal, voltage, and current stability of processors [32, 88], making their modification for security purposes complex and risky.

FPGA-to-CPU remote voltage drop attacks.

As discussed in Section 3.2.2, no current countermeasure effectively prevents FPGA-to-CPU undervolting attacks. This vulnerability is newly discovered, and existing mitigation for FPGA-to-FPGA attacks [42, 43] do not apply here. Static bitstream analysis can be circumvented [24], and dynamic detection techniques may not react in time to stop fault injection [59]. These findings call for the design of novel detection or isolation mechanisms specific to inter-component energy-based attacks.

2) SoC-wide countermeasures for DVFS fault attacks

As highlighted in Section 5.2, power management vulnerabilities arise from the interconnection of components in increasingly complex systems, where resources such as the PDN are shared while isolation is expected. Therefore, defending against DVFS-based fault attacks requires moving beyond a TEE-centric protection model and considering the platform as a whole.

One approach is to extend countermeasures across the entire SoC. In TrustZone-enabled systems, for instance, the processor's security state is propagated to peripherals via the AXI bus [63]. This concept could be extended to the design of TEE-aware power management mechanisms, enabling the secure configuration and monitoring of the PMIC based on the processor's security state.

An alternative strategy would be to rethink the TEE architecture itself. Rather than executing both trusted and untrusted code on the same CPU, as is the case in Arm TrustZone, which represents the most widespread TEE implementation, trusted execution could be offloaded to a separate processor dedicated to execute sensitive programs. In this architecture, this processor running the TEE would require a fully independent power management domain, ensuring immunity to energy-based attacks originating from untrusted components elsewhere in the system.

3) Frameworks for estimating the exploitability of internal energy-based attacks

In order to better estimate the attack surface represented by attacks based on the exploitation of power management mechanisms, as explained in Section 5.1, we advocate for the design of a standard framework for software interactions with the PMIC. Finally, it would also be beneficial to enhance the literature on this subject by carrying a theoretical model of the exploitability of power sensors for an SCA, depending on their precision and refresh rate in respect with the attack exploitation requirements.

6 Conclusion

In this article, we conducted a thorough literature review of a novel category of software-induced hardware attacks, focusing on the malicious exploitation of power management mechanisms. We also examined existing countermeasures aimed at mitigating these attacks and explored potential strategies for integrating them into TEEs. Our analysis underscores the considerable challenges inherent in safeguarding TEE implementations against energy-based attacks, highlighting this as a promising avenue for future research. Our key conclusions can be summarized as follows.

Internal energy-based attacks, including FIAs, SCAs, and covert communications, pose a significant threat to TEE implementations. This evolving threat landscape demands attention as attackers continue to develop more sophisticated and potent attack vectors. Despite the increasing prominence of these attacks, previous academic work on hardware TEE designs has largely overlooked them. Therefore, it is imperative that next-generation TEE technologies integrate robust countermeasures to address this evolving threat landscape.

Emerging countermeasures against internal energy-based FIAs, as evidenced in recent literature [39, 97], offer promising avenues to secure TEEs against these threats without overly constraining their power management mechanisms. Nonetheless, the practical implementation and testing of these countermeasures, either in commercial TEEs or in academic RISC-V based TEE designs, require further evaluation. Additionally, the exploration of mitigation strategies for energy-management-based SCAs remains an open area for future research.

Acknowledgments

This research is supported by the French National Research Agency (ANR) under the *CoPhyTEE* Young Researcher (JCJC) project contract ANR-23-CE39-0003-01.

References

[1] Murugappan Alagappan, Jeyavijayan Rajendran, Milos Doroslovacki, and Guru Venkataramani. 2017. DFS Covert Channels on Multi-Core Platforms. In *2017 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*. IEEE, Abu Dhabi, 1–6. doi:10.1109/VLSI-SoC.2017.8203469

[2] ARM. 2019. Power and Performance Management Using Arm SCMI Specification.

[3] Raad Bahmani, Ferdinand Brasser, Ghada Dessouky, Patrick Jauernig, Matthias Klimmek, Ahmad-Reza Sadeghi, and Emmanuel Stapf. 2021. {CURE}: A Security Architecture with {CUstomizable} and Resilient Enclaves. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX, 1073–1090.

[4] Rassul Bairamkulov and Eby Friedman. 2023. Placement of On-Chip Distributed Voltage Regulators. In *Graphs in VLSI*, Rassul Bairamkulov and Eby G. Friedman (Eds.). Springer International Publishing, Cham, 217–236. doi:10.1007/978-3-031-11047-4_8

[5] El Mehdi Benhani and Lilian Bossuet. 2018. DVFS as a Security Failure of TrustZone-enabled Heterogeneous SoC. In *2018 25th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE, Bordeaux, 489–492. doi:10.1109/ICECS.2018.8618038

[6] P. Bose, A. Buyuktosunoglu, J. A. Darringer, M. S. Gupta, M. B. Healy, H. Jacobson, I. Nair, J. A. Rivers, J. Shin, A. Vega, and A. J. Weger. 2012. Power Management of Multi-Core Chips: Challenges and Pitfalls. In *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, Dresden, 977–982. doi:10.1109/DATE.2012.6176638

[7] Lilian Bossuet and Carlos Andres Lara-Nino. 2023. Advanced Covert-Channels in Modern SoCs. In *2023 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, San Jose, CA, USA, 80–88. doi:10.1109/HOST55118.2023.10133626

[8] Giovanni Camurati, Sebastian Poeplau, Marius Muench, Tom Hayes, and Aurélien Francillon. 2018. Screaming Channels: When Electromagnetic Side Channels Meet Radio Transceivers. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18)*. Association for Computing Machinery, New York, NY, USA, 163–177. doi:10.1145/3243734.3243802

[9] Md Hafizul Islam Chowdhuryy, Zhenkai Zhang, and Fan Yao. 2024. PowSpectre: Powering Up Speculation Attacks with TSX-based Replay. In *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security (ASIA CCS '24)*. Association for Computing Machinery, New York, NY, USA, 498–511. doi:10.1145/3634737.3661139

[10] Victor Costan, Ilia Lebedev, and Srinivas Devadas. 2016. Sanctum: Minimal Hardware Extensions for Strong Software Isolation. In *25th USENIX Security Symposium (USENIX Security 16)*. USENIX, Austin, TX, USA, 857–874.

[11] Elke De Mulder, Samatha Gummalla, and Michael Hutter. 2019. Protecting RISC-V against Side-Channel Attacks. In *Proceedings of the 56th Annual Design Automation Conference 2019*. ACM, Las Vegas NV USA, 1–4. doi:10.1145/3316781.3323485

[12] Ghada Dessouky, Ahmad-Reza Sadeghi, and Emmanuel Stapf. 2020. Enclave Computing on RISC-V: A Brighter Future for Security?. In *1st International Workshop on Secure RISC-V Architecture Design Exploration (SECRISC-V), Co-Located with ISPASS-2020*. Boston, USA.

[13] Kevin Devey, David Hunt, and Chris MacNamara. 2021. Power Management - Technology Overview.

[14] Debopriya Roy Dipta and Berk Gulmezoglu. 2022. DF-SCA: Dynamic Frequency Side Channel Attacks Are Practical. In *Proceedings of the 38th Annual Computer Security Applications Conference*. ACM, Austin TX USA, 841–853. doi:10.1145/3564625.3567979

[15] Linux Kernel Documentation. 2008. Regulator Consumer Driver Interface.

[16] Jan-Erik Ekberg, Kari Kostiainen, and N. Asokan. 2013. Trusted Execution Environments on Mobile Devices. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security - CCS '13*. ACM Press, Berlin, Germany, 1497–1498. doi:10.1145/2508859.2516758

[17] Eurosmart. 2021. Secure Sub-System in System-on-Chip (3S in SoC) Protection Profile.

[18] Shufan Fei, Zheng Yan, Wenxiu Ding, and Haomeng Xie. 2022. Security Vulnerabilities of SGX and Countermeasures: A Survey. *Comput. Surveys* 54, 6 (July 2022), 1–36. doi:10.1145/3456631

[19] Qian Ge, Yuval Yarom, David Cock, and Gernot Heiser. 2018. A Survey of Microarchitectural Timing Attacks and Countermeasures on Contemporary Hardware. *Journal of Cryptographic Engineering* 8, 1 (April 2018), 1–27. doi:10.1007/s13389-016-0141-6

[20] Ilias Giechaskiel, Kasper Bonne Rasmussen, and Jakub Szefer. 2020. C ^{3} APSULe: Cross-FPGA Covert-Channel Attacks through Power Supply Unit Leakage. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, San Francisco, CA, USA, 1728–1741. doi:10.1109/SP40000.2020.00070

[21] Vincent Giraud and David Naccache. 2023. Power Analysis Pushed Too Far: Breaking Android-Based Isolation with Fuel Gauges. In *Advances in Information and Computer Security*, Junji Shikata and Hiroki Kuzuno (Eds.). Springer Nature Switzerland, Cham, 3–15. doi:10.1007/978-3-031-41326-1_1

[22] GlobalPlatform. 2020. TEE Protection Profile v1.3.

[23] Kamil Gomina, Jean-Baptiste Rigaud, Philippe Gendrier, Philippe Candelier, and Assia Tria. 2014. Power Supply Glitch Attacks: Design and Evaluation of Detection Circuits. In *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*. IEEE, Arlington, VA, USA, 136–141. doi:10.1109/HST.2014.6855584

[24] Mathieu Gross, Jonas Krautter, Dennis Gnad, Michael Gruber, Georg Sigl, and Mehdi Tahoori. 2023. FPGANeedle: Precise Remote Fault Attacks from FPGA to CPU. In *Proceedings of the 28th Asia and South Pacific Design Automation Conference (ASPDAC '23)*. Association for Computing Machinery, New York, NY, USA, 358–364. doi:10.1145/3566097.3568352

[25] Daniel Gruss, Clémentine Maurice, and Stefan Mangard. 2016. Rowhammer.Js: A Remote Software-Induced Fault Attack in JavaScript. doi:10.48550/arXiv.1507.06955 arXiv:1507.06955 [cs]

[26] Yanan Guo, Dingyuan Cao, Xin Xin, Youtao Zhang, and Jun Yang. 2023. Uncore Encore: Covert Channels Exploiting Uncore Frequency Scaling. In *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '23)*. Association for Computing Machinery, New York, NY, USA, 843–855. doi:10.1145/3613424.3614259

[27] Daniel Hackenberg, Thomas Ilsche, Robert Schöne, Daniel Molka, Maik Schmidt, and Wolfgang E. Nagel. 2013. Power Measurement Techniques on Standard Compute Nodes: A Quantitative Comparison. In *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, Austin, TX, USA, 194–204. doi:10.1109/ISPASS.2013.6557170

[28] Jawad Haj-Yahya, Lois Orosa, Jeremie S. Kim, Juan Gómez Luna, A. Giray Yağlıkçı, Mohammed Alser, Ivan Puddu, and Onur Mutlu. 2021. IChannels: Exploiting Current Management Mechanisms to Create Covert Channels in Modern Processors. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. 985–998. doi:10.1109/ISCA52012.2021.00081

[29] Junying Huang, Jing Ye, Xiaochun Ye, Da Wang, Dongrui Fan, Huawei Li, Xiaowei Li, and Zhimin Zhang. 2019. Instruction Vulnerability Test and Code Optimization Against DVFS Attack. In *2019 IEEE International Test Conference in Asia (ITC-Asia)*. IEEE, Tokyo, Japan, 49–54. doi:10.1109/ITC-Asia.2019.00022

[30] Intel. 2021. Intel® 64 and IA-32 Architectures Software Developer's Manual. Vol. 3, Part 16.4.

[31] Intel. 2022. Running Average Power Limit Energy Reporting CVE-2020-8694, CVE-2020-8695.

[32] Intel. 2023. What Is Throttling and How Can It Be Resolved? https://www.intel.com/content/www/us/en/support/articles/000088048.html.

[33] Jonas Juffinger, Stepan Kalinin, Daniel Gruss, and Frank Mueller. 2024. SUIT: Secure Undervolting with Instruction Traps. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24, Vol. 2)*. Association for Computing Machinery, New York, NY, USA, 1128–1145. doi:10.1145/3620665.3640373

[34] Mashrafi Alam Kajol, Sandeep Sunkavilli, and Qiaoyan Yu. 2023. AHD-LAM: A New Mitigation Method against Voltage-Drop Attacks in Multi-tenant FPGAs. In *2023 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*. IEEE, 1–6. doi:10.1109/AsianHOST59942.2023.10409460

[35] Manuel Kalmbach, Mathias Gottschlag, Tim Schmidt, and Frank Bellosa. 2020. TurboCC: A Practical Frequency-Based Covert Channel With Intel Turbo Boost. doi:10.48550/arXiv.2007.07046 arXiv:2007.07046 [cs]

[36] Ben Keller, Martin Cochet, Brian Zimmer, Jaehwa Kwak, Alberto Puggelli, Yunsup Lee, Milovan Blagojevic, Stevo Bailey, Pi-Feng Chiu, Palmer Dabbelt, Colin Schmidt, Elad Alon, Krste Asanovic, and Borivoje Nikolic. 2017. A RISC-V Processor SoC With Integrated Power Management at Submicrosecond Timescales in 28 Nm FD-SOI. *IEEE Journal of Solid-State Circuits* 52, 7 (July 2017), 1863–1875. doi:10.1109/JSSC.2017.2690859

[37] Zijo Kenjar, Tommaso Frassetto, David Gens, Michael Franz, and Ahmad-Reza Sadeghi. 2020. {V0LTpwn}: Attacking X86 Processor Integrity from Software. In *29th USENIX Security Symposium (USENIX Security 20)*. USENIX, 1445–1461.

[38] S. Karen Khatamifard, Longfei Wang, Amitabh Das, Selcuk Kose, and Ulya R. Karpuzcu. 2019. POWERT Channels: A Novel Class of Covert CommunicationExploiting Power Management Vulnerabilities. In *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, Washington, DC, USA, 291–303. doi:10.1109/HPCA.2019.00045

[39] Andreas Kogler, Daniel Gruss, and Michael Schwarz. 2023. Minefield: A Software-only Protection for SGX Enclaves against DVFS Attacks. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, USA, 4147–4164.

[40] Andreas Kogler, Jonas Juffinger, Lukas Giner, Lukas Gerlach, Martin Schwarzl, Michael Schwarz, Daniel Gruss, and Stefan Mangard. 2023. {Collide+Power}: Leaking Inaccessible Data with Software-based Power Side Channels. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX, Anaheim, CA, USA, 7285–7302.

[41] Jonas Krautter, Dennis R. E. Gnad, and Mehdi B. Tahoori. 2018. FPGAhammer: Remote Voltage Fault Attacks on Shared FPGAs, Suitable for DFA on AES. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (Aug. 2018), 44–68. doi:10.13154/tches.v2018.i3.44-68

[42] Jonas Krautter, Dennis R. E. Gnad, and Mehdi B. Tahoori. 2019. Mitigating Electrical-level Attacks towards Secure Multi-Tenant FPGAs in the Cloud. *ACM Trans. Reconfigurable Technol. Syst.* 12, 3 (Aug. 2019), 12:1–12:26. doi:10.1145/3328222

[43] Tuan Minh La, Kaspar Matas, Nikola Grunchevski, Khoa Dang Pham, and Dirk Koch. 2020. FPGADefender: Malicious Self-oscillator Scanning for Xilinx UltraScale + FPGAs. *ACM Trans. Reconfigurable Technol. Syst.* 13, 3 (Sept. 2020), 15:1–15:31. doi:10.1145/3402937

[44] Johan Laurent, Vincent Beroulle, Christophe Deleuze, Florian Pebay-Peyroula, and Athanasios Papadimitriou. 2019. Cross-Layer Analysis of Software Fault Models and Countermeasures against Hardware Fault Attacks in a RISC-V Processor. *Microprocessors and Microsystems* 71 (Nov. 2019), 102862. doi:10.1016/j.micpro.2019.102862

[45] Dayeol Lee, David Kohlbrenner, Shweta Shinde, Krste Asanović, and Dawn Song. 2020. Keystone: An Open Framework for Architecting Trusted Execution Environments. In *Proceedings of the Fifteenth European Conference on Computer*

Systems. ACM, Heraklion Greece, 1–16. doi:10.1145/3342195.3387532

[46] Moritz Lipp, Andreas Kogler, David Oswald, Michael Schwarz, Catherine Easdon, Claudio Canella, and Daniel Gruss. 2021. PLATYPUS: Software-based Power Side-Channel Attacks on X86. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, San Francisco, CA, USA, 355–371. doi:10.1109/SP40001.2021.00063

[47] Chen Liu, Abhishek Chakraborty, Nikhil Chawla, and Neer Roggel. 2022. Frequency Throttling Side-Channel Attack. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22)*. Association for Computing Machinery, New York, NY, USA, 1977–1991. doi:10.1145/3548606.3560682

[48] Dina G. Mahmoud, David Dervishi, Samah Hussein, Vincent Lenders, and Mirjana Stojilović. 2022. DFAulted: Analyzing and Exploiting CPU Software Faults Caused by FPGA-Driven Undervolting Attacks. *IEEE Access* 10 (2022), 134199–134216. doi:10.1109/ACCESS.2022.3231753

[49] Dina G. Mahmoud, Samah Hussein, Vincent Lenders, and Mirjana Stojilović. 2022. FPGA-to-CPU Undervolting Attacks. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, Antwerp, Belgium, 999–1004. doi:10.23919/DATE54114.2022.9774663

[50] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. 2008. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer Science & Business Media.

[51] Heiko Mantel, Johannes Schickel, Alexandra Weber, and Friedrich Weber. 2018. How Secure Is Green IT? The Case of Software-Based Energy Side Channels. In *Computer Security (Lecture Notes in Computer Science)*, Javier Lopez, Jianying Zhou, and Miguel Soriano (Eds.). Springer International Publishing, Cham, 218–239. doi:10.1007/978-3-319-99073-6_11

[52] Yan Michalevsky, Aaron Schulman, Gunaa Arumugam Veerapandian, Dan Boneh, and Gabi Nakibly. 2015. {PowerSpy}: Location Tracking Using Mobile Device Power Analysis. In *24th USENIX Security Symposium (USENIX Security 15)*. USENIX, Washington, D.C., 785–800.

[53] Sébastien Michelland, Christophe Deleuze, and Laure Gonnord. 2024. From Low-Level Fault Modeling (of a Pipeline Attack) to a Proven Hardening Scheme. In *Compiler Construction (CC'24)*. Association for Computing Machinery, Edinburgh (Scotland), United Kingdom, 174–185. doi:10.1145/3640537.3641570

[54] Philipp Miedl, Xiaoxi He, Matthias Meyer, Davide Basilio Bartolini, and Lothar Thiele. 2018. Frequency Scaling As a Security Threat on Multicore Systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 37, 11 (Nov. 2018), 2497–2508. doi:10.1109/TCAD.2018.2857038

[55] Nimish Mishra, Tridib Lochan Dutta, Shubhi Shukla, Anirban Chakraborty, and Debdeep Mukhopadhyay. 2024. Too Hot to Handle: Novel Thermal Side-Channel in Power Attack-Protected Intel Processors. In *2024 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, Tysons Corner, VA, USA, 378–382. doi:10.1109/HOST55342.2024.10545405

[56] Nimish Mishra, Rahul Arvind Mool, Anirban Chakraborty, and Debdeep Mukhopadhyay. 2024. Plug Your Volt: Protecting Intel Processors against Dynamic Voltage Frequency Scaling Based Fault Attacks. In *Proceedings of the 61st ACM/IEEE Design Automation Conference (DAC '24)*. Association for Computing Machinery, New York, NY, USA, 1–6. doi:10.1145/3649329.3658468

[57] Kit Murdock, David Oswald, Flavio D. Garcia, Jo Van Bulck, Daniel Gruss, and Frank Piessens. 2020. Plundervolt Source Code. https://github.com/0x0atang/clkscrew.

[58] Kit Murdock, David Oswald, Flavio D. Garcia, Jo Van Bulck, Daniel Gruss, and Frank Piessens. 2020. Plundervolt: Software-based Fault Injection Attacks against Intel SGX. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, San Francisco, CA, USA, 1466–1482. doi:10.1109/SP40000.2020.00057

[59] Hassan Nassar, Hanna AlZughbi, Dennis R. E. Gnad, Lars Bauer, Mehdi B. Tahoori, and Jörg Henkel. 2021. LoopBreaker: Disabling Interconnects to Mitigate Voltage-Based Attacks in Multi-Tenant FPGAs. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, Munich, Germany, 1–9. doi:10.1109/ICCAD51958.2021.9643485

[60] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. 2006. Threshold Implementations Against Side-Channel Attacks and Glitches. In *Information and Communications Security*, David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Dough Tygar, Moshe Y. Vardi, Gerhard Weikum, Peng Ning, Sihan Qing, and Ninghui Li (Eds.). Vol. 4307. Springer Berlin Heidelberg, Berlin, Heidelberg, 529–545. doi:10.1007/11935308_38

[61] Safouane Noubir, Maria Méndez Real, and Sebastien Pillement. 2020. Towards Malicious Exploitation of Energy Management Mechanisms. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, Grenoble, France, 1043–1048. doi:10.23919/DATE48585.2020.9116420

[62] Justin D Osborn and David C Challener. 2013. Trusted Platform Module Evolution. *JOHNS HOPKINS APL TECHNICAL DIGEST* 32, 2 (2013), 536–543.

[63] Sandro Pinto and Nuno Santos. 2019. Demystifying Arm TrustZone: A Comprehensive Survey. *Comput. Surveys* 51, 6 (Nov. 2019), 1–36. doi:10.1145/3291047

[64] Rajesh Poornachandran and Ned M. Smith. 2015. Enhanced Security of Power Management Communications and Protection from Side Channel Attacks.

[65] George Provelengios, Daniel Holcomb, and Russell Tessier. 2021. Mitigating Voltage Attacks in Multi-Tenant FPGAs. *ACM Trans. Reconfigurable Technol. Syst.* 14, 2 (July 2021), 9:1–9:24. doi:10.1145/3451236

[66] Yi Qin and Chuan Yue. 2018. Website Fingerprinting by Power Estimation Based Side-Channel Attacks on Android 7. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, New York, NY, USA, 1030–1039. doi:10.1109/TrustCom/BigDataSE.2018.00145

[67] Pengfei Qiu, Dongsheng Wang, Yongqiang Lyu, and Gang Qu. 2019. VoltJockey: Breaching TrustZone by Software-Controlled Voltage Manipulation over Multi-core Frequencies. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. ACM, London United Kingdom, 195–209. doi:10.1145/3319535.3354201

[68] Pengfei Qiu, Dongsheng Wang, Yongqiang Lyu, and Gang Qu. 2019. VoltJockey: Breaking SGX by Software-Controlled Voltage-Induced Hardware Faults. In *2019 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*. IEEE, Xi'an, China, 1–6. doi:10.1109/AsianHOST47458.2019.9006701

[69] Pengfei Qiu, Dongsheng Wang, Yongqiang Lyu, and Gang Qu. 2022. DVFSspy: Using Dynamic Voltage and Frequency Scaling as a Covert Channel for Multiple Procedures. In *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, Taipei, Taiwan, 654–659. doi:10.1109/ASP-DAC52403.2022.9712588

[70] Anja Rabich. 2020. *Software-based Undervolting Faults in AMD Zen Processors*. Ph. D. Dissertation. University of Lübeck.

[71] Barry Rountree, Dong H. Ahn, Bronis R. de Supinski, David K. Lowenthal, and Martin Schulz. 2012. Beyond DVFS: A First Look at Performance under a Hardware-Enforced Power Bound. In *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*. IEEE, Shanghai, China, 947–953. doi:10.1109/IPDPSW.2012.116

[72] Majid Sabbagh, Yunsi Fei, and David Kaeli. 2020. A Novel GPU Overdrive Fault Attack. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, San Francisco, CA, USA, 1–6. doi:10.1109/DAC18072.2020.9218690

[73] Mohamed Sabt, Mohammed Achemlal, and Abdelmadjid Bouabdallah. 2015. Trusted Execution Environment: What It Is, and What It Is Not. In *2015 IEEE Trustcom/BigDataSE/ISPA*. IEEE, Helsinki, Finland, 57–64. doi:10.1109/Trustcom.2015.357

[74] Gururaj Saileshwar, Christopher W. Fletcher, and Moinuddin Qureshi. 2021. Streamline: A Fast, Flushless Cache Covert-Channel Attack by Enabling Asynchronous Collusion. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '21)*. Association for Computing Machinery, New York, NY, USA, 1077–1090. doi:10.1145/3445814.3446742

[75] Sandeep Saxena, Goutam Sanyal, and Manu. 2018. Cache Based Side Channel Attack: A Survey. In *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*. IEEE, Greater Noida, India, 278–284. doi:10.1109/ICACCCN.2018.8748811

[76] Robert Schöne, Thomas Ilsche, Mario Bielert, Markus Velten, Markus Schmidl, and Daniel Hackenberg. 2021. Energy Efficiency Aspects of the AMD Zen 2 Architecture. In *2021 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, Los Alamitos, CA, USA, 562–571. doi:10.1109/Cluster48925.2021.00087 arXiv:2108.00808 [cs]

[77] David Schrammel, Moritz Waser, Lukas Lamster, Martin Unterguggenberger, and Stefan Mangard. 2023. SPEAR-V: Secure and Practical Enclave Architecture for RISC-V. In *Proceedings of the ACM Asia Conference on Computer and Communications Security*. ACM, Melbourne VIC Australia, 457–468. doi:10.1145/3579856.3595784

[78] Markus Schweikhardt and Matthias Hahn. 2022. DFS for Mixed Criticality Real Time Scenarios on 11th Generation Intel Core Processors.

[79] Johanna Sepúlveda, Mathieu Gross, Andreas Zankl, and Georg Sigl. 2021. Beyond Cache Attacks: Exploiting the Bus-based Communication Structure for Powerful On-Chip Microarchitectural Attacks. *ACM Trans. Embed. Comput. Syst.* 20, 2 (March 2021), 17:1–17:23. doi:10.1145/3433653

[80] Alireza Shameli-Sendi. 2021. Understanding Linux Kernel Vulnerabilities. *Journal of Computer Virology and Hacking Techniques* 17, 4 (Dec. 2021), 265–278. doi:10.1007/s11416-021-00379-x

[81] Shengqi Yang, W. Wolf, N. Vijaykrishnan, D.N. Serpanos, and Yuan Xie. 2005. Power Attack Resistant Cryptosystem Design: A Dynamic Voltage and Frequency Switching Approach. In *Design, Automation and Test in Europe*. IEEE, Munich, Germany, 64–69. doi:10.1109/DATE.2005.241

[82] Amit Mazumder Shuvo, Tao Zhang, Farimah Farahmandi, and Mark Tehranipoor. 2023. A Comprehensive Survey on Non-Invasive Fault Injection Attacks.

[83] Rihui Sun, Pengfei Qiu, Yongqiang Lyu, Jian Dong, Haixia Wang, Dongsheng Wang, and Gang Qu. 2023. Lightning: Leveraging DVFS-induced Transient Fault Injection to Attack Deep Learning Accelerator of GPUs. *ACM Trans. Des. Autom. Electron. Syst.* 29, 1 (Nov. 2023), 14:1–14:22. doi:10.1145/3617893

[84] Hritvik Taneja, Jason Kim, Jie Jeff Xu, Stephan van Schaik, Daniel Genkin, and Yuval Yarom. 2023. Hot Pixels: Frequency, Power, and Temperature Attacks on GPUs and ARM SoCs. doi:10.48550/arXiv.2305.12784 arXiv:2305.12784 [cs]

[85] Adrian Tang, Simha Sethumadhavan, and Salvatore Stolfo. 2017. CLKscrew Code and Scripts.

[86] Adrian Tang, Simha Sethumadhavan, and Salvatore Stolfo. 2017. {CLKSCREW}: Exposing the Perils of {Security-Oblivious} Energy Management. In *26th USENIX Security Symposium (USENIX Security 17)*. USENIX, Vancouver, BC, Canada, 1057–1074.

[87] Zikang Tao, Rihui Sun, and Jian Dong. 2023. Software Countermeasures against DVFS Fault Attack for AES. In *2023 10th International Conference on Dependable Systems and Their Applications (DSA)*. IEEE, Tokyo, Japan, 575–582. doi:10.1109/DSA59317.2023.00080

[88] Yingchen Wang, Riccardo Paccagnella, Elizabeth Tang He, Hovav Shacham, Christopher W. Fletcher, and David Kohlbrenner. 2023. Hertzbleed: Turning Power Side-Channel Attacks Into Remote Timing Attacks on X86. *IEEE Micro* 43, 4 (July 2023), 19–27. doi:10.1109/MM.2023.3274619

[89] Yingchen Wang, Riccardo Paccagnella, Alan Wandke, Zhao Gang, Grant Garrett-Grossman, Christopher W. Fletcher, David Kohlbrenner, and Hovav Shacham. 2023. DVFS Frequently Leaks Secrets: Hertzbleed Attacks Beyond SIKE, Cryptography, and CPU-Only Data. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, San Francisco, CA, USA, 2306–2320. doi:10.1109/SP46215.2023.10179326

[90] Samuel Weiser, Mario Werner, Ferdinand Brasser, Maja Malenko, Stefan Mangard, and Ahmad-Reza Sadeghi. 2019. TIMBER-V: Tag-Isolated Memory Bringing Fine-grained Enclaves to RISC-V. In *Proceedings 2019 Network and Distributed System Security Symposium*. Internet Society, San Diego, CA, 15. doi:10.14722/ndss.2019.23068

[91] Mario Werner, Robert Schilling, Thomas Unterluggauer, and Stefan Mangard. 2019. Protecting RISC-V Processors against Physical Attacks. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, Florence, Italy, 1136–1141. doi:10.23919/DATE.2019.8714811

[92] Junge Xu, Bohan Xuan, Anlin Liu, Mo Sun, Fan Zhang, Zeke Wang, and Kui Ren. 2022. Terminator on SkyNet: A Practical DVFS Attack on DNN Hardware IP for UAV Object Detection. In *Proceedings of the 59th ACM/IEEE Design Automation Conference (DAC '22)*. Association for Computing Machinery, New York, NY, USA, 685–690. doi:10.1145/3489517.3530516

[93] Junge Xu, Fan Zhang, Wenguang Jin, Kun Yang, Zeke Wang, Weixiong Jiang, and Yajun Ha. 2025. A Deep Investigation on Stealthy DVFS Fault Injection Attacks at DNN Hardware Accelerators. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 44, 1 (Jan. 2025), 39–51. doi:10.1109/TCAD.2024.3426364

[94] Lin Yan, Yao Guo, Xiangqun Chen, and Hong Mei. 2015. A Study on Power Side Channels on Mobile Devices. doi:10.48550/arXiv.1512.07972 arXiv:1512.07972 [cs]

[95] Bilgiday Yuce, Nahid F. Ghalaty, Chinmay Deshpande, Conor Patrick, Leyla Nazhandali, and Patrick Schaumont. 2016. FAME: Fault-attack Aware Microprocessor Extensions for Hardware Fault Detection and Software Fault Response. In *Proceedings of the Hardware and Architectural Support for Security and Privacy 2016*. ACM, Seoul Republic of Korea, 1–8. doi:10.1145/2948618.2948626

[96] Mengchao Yue, William H. Robinson, Lanier Watkins, and Cherita Corbett. 2014. Constructing Timing-Based Covert Channels in Mobile Networks by Adjusting CPU Frequency. In *Proceedings of the Third Workshop on Hardware and Architectural Support for Security and Privacy (HASP '14)*. Association for Computing Machinery, New York, NY, USA, 1–8. doi:10.1145/2611765.2611768

[97] Sheng Zhang, Adrian Tang, Zhewei Jiang, Simha Sethumadhavan, and Mingoo Seok. 2018. Blacklist Core: Machine-Learning Based Dynamic Operating-Performance-Point Blacklisting for Mitigating Power-Management Security Attacks. In *Proceedings of the International Symposium on Low Power Electronics and Design*. ACM, Seattle WA USA, 1–6. doi:10.1145/3218603.3218624

[98] Mark Zhao and G. Edward Suh. 2018. FPGA-Based Remote Power Side-Channel Attacks. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, San Francisco, CA, USA, 229–244. doi:10.1109/SP.2018.00049