Optical Cryptanalysis: Recovering Cryptographic Keys from Power LED Light Fluctuations

Ben Nassi[1,2], Ofek Vayner[1], Etay Iluz[1], Dudi Nassi[1], Or Hai Cohen[1], Jan Jancar[3], Daniel Genkin[4], Eran Tromer[5], Boris Zadov[1], Yuval Elovici[1]

[1] Ben-Gurion University of the Negev, [2] Cornell Tech, [3] Masaryk University, [4] Georgia Tech, [5] Columbia University

{nassib, ofekvay, etayil, nassid, ora2, zadov}@post.bgu.ac.il, bn267@cornell.edu, elovici@bgu.ac.il, genkin@gatech.edu, 445358@mail.muni.cz, et2555@columbia.edu

ABSTRACT

Although power LEDs have been integrated in various devices that perform cryptographic operations for decades, the cryptanalysis risk they pose has not yet been investigated. In this paper, we present optical cryptanalysis, a new form of cryptanalytic side-channel attack, in which secret keys are extracted by using a photodiode to measure the light emitted by a device's power LED and analyzing subtle fluctuations in the light intensity during cryptographic operations. We analyze the optical leakage of power LEDs of various consumer devices and the factors that affect the optical SNR. We then demonstrate end-to-end optical cryptanalytic attacks against a range of consumer devices (smartphone, smartcard, and Raspberry Pi, along with their USB peripherals) and recover secret keys (RSA, ECDSA, SIKE) from prior and recent versions of popular cryptographic libraries (GnuPG, Libgcrypt, PQCrypto-SIDH) from a maximum distance of 25 meters.

TABLE I
THE TABLE MAPS THE DEVICES, CRYPTOGRAPHIC LIBRARIES, CRYPTANALYTIC ATTACKS, AND SECRET KEYS EXTRACTED IN THIS PAPER.

	Minerva Attack [11]				Acoustic Cryptanalysis [12]	Hertz-bleed [13]
	Raspberry Pi 3B+	Connected headphones*	Connected USB hub*	Smartcard reader**	Raspberry Pi 4	Samsung Galaxy S8
Libgcrypt 1.8.4	265-bit ECDSA key	265-bit ECDSA key	265-bit ECDSA key			
Unknown library***				265-bit ECDSA key		
GnuPG 1.4.13					4096-bit RSA Key	
PQCrypto-SIDH 3.4						378-bit SIKE key

*The power LED of the headphones and USB hub (that were connected to a Raspberry Pi3B+) was used to extract the keys.
**The power LED of the smartcard reader was used to extract the secret key from the smartcard.
**We do not know which cryptographic library is installed on the smartcard.

I. INTRODUCTION

Cryptanalytic side-channel attacks, which are aimed at recovering secret keys from devices, are considered a great threat to information confidentiality. Recent studies have demonstrated novel methods for performing such attacks against various devices by exploiting the correlation between the cryptographic operations executed by a device and its power consumption, electromagnetic radiation (EMR) leakage, acoustic leakage, cache access behavior, etc. [1–10]. These methods have increased understanding regarding the design of resilient devices and cryptographic algorithms, using hardware and software primitives secured cryptanalytic side-channel attacks.

While many papers have been published on cryptanalysis using the aforementioned leakage, little is known about cryptanalysis using optical leakage. This is surprising given that: (1) various components (e.g., integrated power LEDs, charger LEDs, monitors) that emit visible and non-visible light are integrated in and connected to devices that perform cryptographic operations, and (2) optical leakage can be observed from a distance, making the threat model more attractive to attackers than other threat models that require the attacker to obtain physical access to the target device (e.g., in order to attach probes) or compromise a target device to collect traces (e.g., in order to measure its cache behavior).

A few studies [14–16] performed cryptanalysis by capturing near-infrared photon emissions from switching transistors located on the back of exposed FPGAs during the execution of a proof of concept implementation of cryptographic algorithms. However, the suggested attacks are limited to devices with exposed chips/PCBs, and therefore they cannot be applied against commercial devices where chips are encapsulated in opaque materials (e.g., smartphones, card readers). Moreover, these attacks were not demonstrated on popular cryptographic libraries. To the best of our knowledge, an end-to-end cryptanalytic attack using optical leakage against a commercial consumer device running a popular cryptographic library has not yet been demonstrated.

In this paper, we present optical cryptanalysis, a new attack vector for the recovery of secret keys using cryptanalytic side-channel attacks. In optical cryptanalysis, a photodiode is used to measure fluctuations in a device's power LED intensity (during cryptographic computations), and the measurements (optical traces) are analyzed to recover the device's secret key. First, we show that optical traces obtained by a photodiode directed at a device's power LED can provide an accurate indication of the power consumption (within a wide spectrum) of the target device directly (using optical traces obtained from the device's power LED) and indirectly (using optical traces

obtained from a connected peripheral's power LED). Then, we analyze the influence of various factors on the optical traces (e.g., the distance between the photodiode and the power LED, ambient light, ripple of the power supply, and target cryptographic library and device under attack).

Finally, we demonstrate the end-to-end application of optical cryptanalysis to recover secret keys (4096-bit RSA key, 256-bit ECDSA key, and 378-bit SIKE key) from commercial consumer devices (smartcard, Raspberry Pi 3 B+ and 4, and Samsung Galaxy S8) that run commonly used cryptographic libraries (GnuPG, Libgcrypt, and PQCryptoSIDH) and connected peripherals (smartcard reader, USB hub, and USB headphones) by obtaining optical traces from the power LEDs of the devices/peripherals and exploiting the vulnerabilities presented in [11–13] (see Table I).

Contributions. (1) We raise awareness about a new type of TEMPEST attack that exploits optical leakage to perform cryptanalytic side-channel attacks using equipment that can be purchased online for a few thousand dollars and demonstrate its end-to-end application (see Table I). (2) In comparison to previously investigated cryptanalytic attack vectors that rely on electromagnetic-radiation (EMR) traces (e.g., [4]), acoustic traces (e.g., [12]), power traces (e.g., [1, 2]), and digital traces (e.g., [8, 9]), optical traces can be obtained unobtrusively (power traces require connecting the device to a scope), without compromising the target device with malware (digital traces can only be obtained by compromising the target device), while providing a higher bandwidth than acoustic traces (which are limited to a few hundred kHz using ultrasonic microphones). (3) We demonstrate key recovery from a distance of 25 meters which, to the best of our knowledge, is a greater distance than state-of-the-art cryptanalytic side-channel attacks that recovered keys from a maximum distance of 4-10 meters [4, 5, 12]).

Structure. In Section II, we review related work. In Section III, we present the threat model, and in Section IV, we present the default experimental setup. The influence of various factors on the optical SNR is analyzed in Section V. In Sections VI-VIII, we describe how we performed optical cryptanalysis to recover ECDSA (Section VI), RSA (Section VII), and SIKE (Section VIII) keys from various devices. We discuss countermeasures in Section IX and limitations in Section X. In Section XI, we discuss the findings of the study.

II. Related Work

Physical cryptanalytic side-channel attacks, which exploit the correlation between the cryptographic computations performed by a device and its physical emanations, have been demonstrated in many prior studies. These studies exploited variations in a device's power consumption to recover secret keys directly, by measuring a device's power consumption (e.g., [1, 2]), or indirectly, by measuring the power consumption's side-effects, including EMR leakage (e.g., [3–5]) and acoustic noise (e.g., [7, 12]).

Some research [14–16] performed cryptanalysis by capturing near-infrared photons emitted from switching transistors located on the back of FPGAs during the execution of a proof of concept implementation of cryptographic algorithms. However, the suggested attacks are ineffective against commercial devices, because their chips are encapsulated in light-blocking covers (e.g., in smartphones). Moreover, these attacks were not demonstrated on a commercial consumer device running a common cryptographic library.

The risks power LEDs pose to devices' information confidentiality have been discussed since 2002 [17]. However, prior research mainly focused on *covert channels* that were established using preinstalled malware that actively controlled the power LEDs of various devices (a keyboard [18], router [19], and hard drive [20]) and exploited them to exfiltrate data. Two recent studies presented optical side-channel attacks and recovered speech played by speakers [21, 22] using optical measurements obtained from the speakers' power LED. While some research investigating the risks that power LEDs pose to confidentiality has been performed, no prior studies examined the risk of cryptanalysis posed by power LEDs. This is surprising due to the fact that power LEDs have been integrated in a variety of devices that perform cryptographic operations for decades.

III. Threat Model

Optical cryptanalysis aims to recover secret keys (e.g., private and symmetric keys) from a device by obtaining optical traces, which are time-dependent measurements of the intensity of the device's power LED.

We assume that the target device is performing cryptographic operations and contains a power/status LED that is always on. The cryptographic operations can be initiated by: (1) the user of the device, e.g., by opening a TLS session to access an HTTPS website or by using a VPN, or (2) an attacker, e.g., by sending the device an encrypted message via an encrypted messaging application (WhatsApp, Telegram, Signal, encrypted email, etc.) in order to trigger automatic decryption or by sending the device messages aimed at triggering automatic digital signing. We consider an attacker that is a malicious entity interested in recovering a secret key from the target device in order to: (1) decrypt previous and future cryptograms delivered to the target device and intercepted by the attacker, or (2) sign on a message on behalf of a target device.

We consider two types of optical data acquisition models, which are based on the physical proximity the attacker has to the target device: (1) **Close data acquisition**. We assume that the attacker has physical access to the device (i.e., the attacker is located in the same room as the target device). In close data acquisition, the attacker places the photodiode near the power LED of the device (e.g., 2 cm away) and obtains optical traces. (2) **Remote data acquisition**. We assume that the attacker does not have physical access to the target device. In remote data acquisition, we assume that the attacker has an optical line of sight to the target device's power LED, and the optical traces are obtained by directing the photodiode at the device's power LED via a telescope (the two data acquisition models are visualized in Fig. 1).

We consider two types of attacks: (1) a **direct attack**, where the optical traces are obtained from the power LED

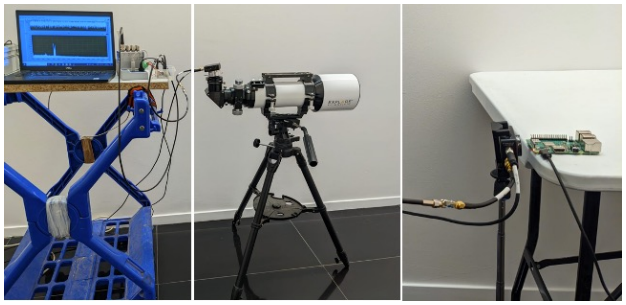

Fig. 1. Left: A LabView script run on a laptop that is connected to an ADC that is connected to an amplifier. Middle: Remote data acquisition in which the amplifier is connected to a photodiode that is mounted to a telescope. Right: Close data acquisition in which the amplifier is connected to a photodiode that is placed 2 cm from the device's power LED.

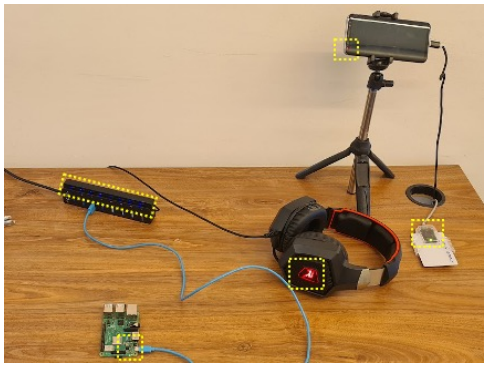

Fig. 2. Some of the devices (card reader, Raspberry Pi 3B+, Samsung Galaxy S8, RUNMUS K8 headset, Gold Touch 8 Ports USB3.0 Slim HUB) and their power LEDs (boxed in yellow).

of the target device, and (2) an **indirect attack**, where the optical traces are obtained from the power LED of a connected peripheral device (that does not perform the cryptographic computations), such as a power supply, card reader, connected USB hub, keyboard, or headphones.

The significance of the threat model with respect to related work is as follows: (1) **Non-invasive** - The attack can be applied remotely, from a distance. The attacker does not need to have physical access to the victim device to obtain the measurements (in contrast to attacks relying on power traces). (2) **Easy to purchase equipment** - The equipment needed to apply the attack can be purchased online for a few thousand dollars (within the budget of non-nation-state attackers). (3) **Indirect** - The attack can be applied indirectly by obtaining optical traces from devices that do not perform cryptographic operations (e.g., from the power LEDs of USB hub splitters, USB headphones). (4) **High bandwidth** - LEDs are highly responsive and can provide a high bandwidth of a few gigabits per second [23]. This is extremely risky in terms of cryptanalysis, because many devices have CPU clock rates under 1 GHz, including various Raspberry Pi models (Zero, 1, 2) and smartcards, and other devices have a CPU clock rate of 1-2 GHz, including newer Raspberry Pi models (3, 4) and various smartphones. This fact can be exploited by attackers to obtain optical traces at a sampling rate higher than or equal to the target device's CPU clock rate by using a high-end photodiode.

IV. Experimental Setup

In this section, we describe the experimental setup used to conduct the experiments described in Sections V-VIII.

Sampling Equipment. We used the Thorlabs PDA100A2 photodiode [24], which is an amplified switchable-gain light sensor that converts light (photons) to electrical current and thus to voltage. We connected the photodiode to a custom-built operational/analog amplifier (with a gain of 50 dB, a 1 kHz high-pass filter, and a bandwidth of 2 MHz). The same photodiode and amplifier were used in all of experiments described in Sections V-VIII. The internal gain of the photodiode was set to the highest level before saturation for each experiment. The amplified voltage was sampled using an NI-9223 ADC card (1M samples per second and 16 bits per sample) in the experiments described in Sections V-VII, while in the experiments described in Section VIII we used an NI PCI-6115 ADC card (5M samples per second and 12 bits per sample), because the frequencies in the spectrum affected by the attack are around 1.21 MHz. We connected the ADC to a laptop; the digital optical traces were visualized in real time on LabView and processed on MATLAB.

Default Setup. The photodiode was placed 2 cm away from the victim power LED. The optical traces were obtained from both dark and sunlit rooms (later, in Section V-E, we show that the spectral behavior is the same in both cases), i.e., we did not turn on the lights in the room. The optical measurements were obtained when there was a direct line of sight between the photodiode and the device's power LED and there were no physical objects between the two. Both for readability and to save space, we consider this setup the default for the experiments described in Sections V-VIII.

In some experiments, we changed the default experimental setup in order to examine the influence of various factors. In cases in which changes were made to the default experimental setup (e.g., a distance greater than 2 cm, a room in which fluorescent lights are on), the differences to the default setup are noted.

Devices. We analyzed several types of devices: an embedded device (card reader), microcontrollers (Raspberry Pi 3B+ and 4B), a smartphone (Samsung Galaxy S8), and a TV streamer (GOtv Streamer). We also analyzed the indirect leakage from connected peripherals (RUNMUS K8 gaming headset, Gold Touch 8 Ports USB3.0 Slim HUB, TP-Link UE300) that do not contain CPUs. Fig. 2 presents some of the devices.

The experimental setup, devices, and results for the experiments described in Sections VI-VIII are summarized in Table V to enable the reader to reproduce the experiments and understand the relationship between the equipment used, the SNR obtained, and the target devices and libraries.

V. Leakage from Power LEDs

In this section, we describe the analysis performed to better understand the potential of power LEDs in the context of cryptanalysis and the influence of various factors on the SNR of the optical traces.

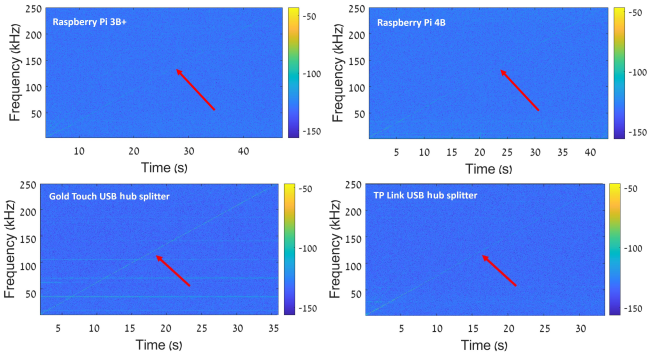

Fig. 3. Spectrograms of power LED optical traces when injecting a 0–500 kHz chirp into the device's power supply. Top: Raspberry Pi 3B+ (left) and 4 (right). Bottom: USB hub splitter models Gold Touch 8 Ports USB3.0 Slim HUB (left) and TP-Link UE300 (right).

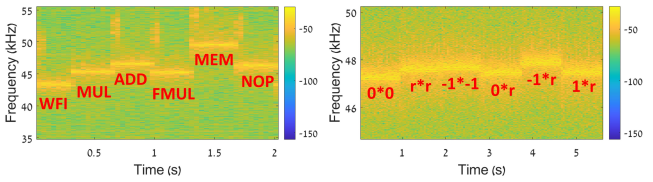

Fig. 4. Spectrograms extracted from optical traces of a Raspberry Pi during the execution of repetitions of: six different CPU operations (left) and big integer multiplication of operands with various operands (right).

A. Bandwidth of a Device's Power LED

First, we examine the response of power LEDs to changes in the device's supply voltage at various frequencies. The following devices were used in these experiments: two Raspberry Pi models (3B+ and 4B) and two USB hub splitters (Gold Touch 8 Ports USB3.0 Slim HUB and TP-Link UE300).

Experimental Setup. In each of the four experiments performed, we connected one of the devices to a function generator that was used to modulate a 0–500 kHz chirp (using a 20 mV peak-to-peak sinusoidal signal) over the 5V power supplied by the function generator to each device. Four optical traces were obtained using the photodiode, which was directed at the power LED of the device during the chirp.

Results. Fig. 3 presents the four spectrograms we extracted from the traces. As can be seen in Fig. 3, although intended to provide a binary indication regarding the state of the device (on/off), the intensity of the power LEDs of the tested devices provides information on the device's power supply voltage (with excellent frequency response, reaching 500 kHz).

One might question the cause of the correlation between the device's power supply and the optical leakage from the device's power LED. We note that various studies have already investigated this topic and found that the intensity of LEDs is greatly affected by the power supply level [25]. We note that in some electrical circuits, the integrated power LED is connected directly to the power line, and dedicated means aimed at decoupling the optical and power correlation are either not integrated or are integrated into the circuits but ineffective (this is illustrated in Fig. 13 in Section IX). As a result, the power LED provides an accurate indication of the power supply.

B. Influence of the CPU Activity

Next, we examine the influence of repeated operations executed by the CPU on the optical traces.

Experimental Setup. We wrote a program that executes repetitions of the following six ARM instructions: WFI (CPU sleep), MUL (integer multiplication), ADD (integer addition), FMUL (floating point multiplication), main memory access (forcing cache misses), and NOP (short-term idle). The repetitions of each CPU operation lasted around 300 ms. We installed the program and executed it on a Raspberry Pi 3B+ while obtaining optical traces.

Results. The spectrogram that we extracted from the optical trace is presented in Fig. 4. As can be seen, the repeated operations affect different frequencies (e.g., the repetition of the FMUL ARM instruction affects 46 kHz, and the repetition of the MEM ARM instruction affects 50 kHz). Based on this experiment, we concluded that repetitions of different operations create unique optical fingerprints in the spectrum of the optical trace. This is due to the fact that different CPU operations consume different amounts of power (see Fig. 14 in Appendix A). The unique and variable power consumption of a CPU operation, combined with the fact that the power LEDs of various devices are connected to the power line creates a unique optical fingerprint.

Next, we examine the influence of repetitions of operands executed by the CPU on the optical traces.

Experimental Setup. We executed a function that implements big integer multiplication (see *fproduct* implementation [26]) on a Raspberry Pi 3B+ while obtaining an optical trace. The code executed repetitions of the function as follows: 0×0 , $r \times r$, (where r is a random number), -1×-1, $0 \times r$, $-1 \times r$, and $1 \times r$ (where r is representative of half of a normalized Hamming weight, and -1 has a maximal Hamming weight).

Results. As can be seen in Fig. 4 which presents the spectrogram extracted from the optical trace, the repetitions of the execution of the big integer multiplication of different operands create unique optical fingerprints in the spectrum of the optical trace (e.g., there is a difference of 0.6 kHz between 0×0 and $-1 \times rand$).

C. Influence of the Device's Power LED

Next, we compare the optical SNR obtained from the power LED of various devices. The experiments were performed by installing a program that we wrote which alternates between 300 ms repetitions of integer multiplications (MUL) and 300 ms sleep operations (WFI). We will refer to this code as the *Prober* code in this section. The *Prober* code was installed on a Raspberry Pi 3B+, Samsung Galaxy S8, and GOtv Streamer.

Experimental Setup. We executed the *Prober* code on the three devices while obtaining one optical trace directly from the power LED of each of the three devices. In addition, three additional optical traces were obtained from the power LED of a USB hub (Gold Touch 8 Ports USB3.0 Slim HUB) that was connected to each device (separately) while the devices executed the *Prober* code. Finally, two additional optical traces were obtained from the power LED of USB headphones (RUNMUS K8 headset) that were connected to Raspberry

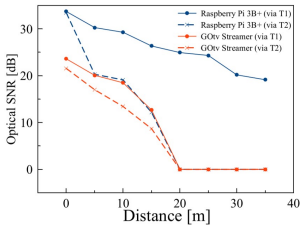

Fig. 5. Optical SNR vs. distance.

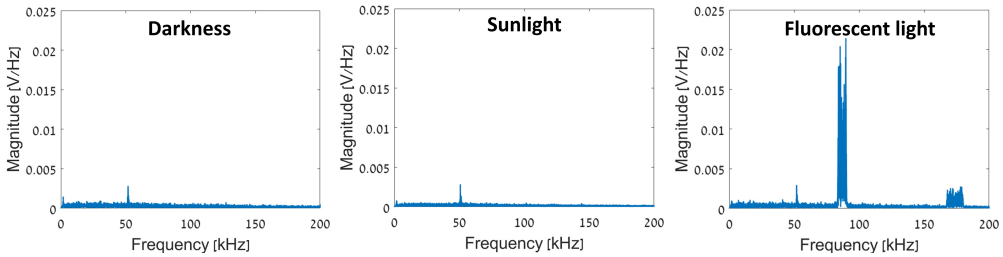

Fig. 6. The influence of the ambient light conditions: darkness (left), sunlight (middle), and fluorescent light (right).

TABLE II
THE OPTICAL SNR OBTAINED FROM THE VARIOUS DEVICES DIRECTLY (FROM THEIR POWER LED) AND INDIRECTLY (FROM THE POWER LED OF CONNECTED DEVICES).

	Direct/Indirect Data Acquisition		
	Directly	Indirectly	
	From the Device's Power LED	From the USB Hub's Power LED	From the USB Headphones' Power LED
Raspberry Pi 3B+	34.8 dB	38.0 dB	30.0 dB
Samsung Galaxy S8	20.1 dB	25.5 dB	-
GOtv Streamer	30.3 dB	33.97 dB	29.1 dB

Pi 3B+ and GOtv Streamer. The Samsung Galaxy does not contain a female USB port, so we were unable to connect the USB headphones to the smartphone.

Results. The spectrograms of the eight optical traces (and the affected frequencies) can be seen in Fig. 15 in Appendix A. We calculated the optical SNR obtained from the eight optical traces by dividing the magnitude around the frequency affected when the multiplications were executed (signal) by the magnitude around the same frequency during sleep (noise); the results are presented in Table II. Based on this experiment, we concluded that: (1) The power LED of the device under attack affects the optical SNR, since there is a difference of ∼14.7 dB in the SNR of the devices in the direct attack and ∼12.5 dB in the SNR of the devices in the indirect attack. (2) The optical leakage is present in the optical traces obtained both directly from the three devices examined and indirectly from a connected peripheral (that does not perform cryptographic operations). (3) The power LED of a connected peripheral may amplify or reduce the optical SNR (depending on the peripheral); in our case, the USB hub increases the optical SNR by ∼3.2-5.4 dB, while the USB headphones decrease the SNR by ∼0.4-4.8 dB (compared to the SNR obtained directly from the target device).

D. Influence of Distance and Telescopes

Next, we examine how the optical SNR is affected by the distance between the target device's power LED and the photodiode.

Experimental Setup. We note that the light emitted from the power LEDs of the devices examined is too weak to be captured from a distance using remote data acquisition by mounting the photodiode to the telescope. However, as was shown in the previous experiment, the connected USB hub (Gold Touch 8 Ports USB3.0 Slim HUB) amplifies the optical SNR, so we connected each of the two devices used in this experiment (the GOtv Streamer and Raspberry Pi 3B+) separately to the USB hub. We executed the *Prober* code on the devices and obtained optical traces from the power LED of the USB hub from various distances (75 cm-35 meters) using two telescopes: Sky-Watcher Flextube 350P with a lens diameter of 35 cm (T1) and Explore Scientific ED102 with a lens diameter of 10.2 cm (T2).

Results. We calculated the SNR from the optical traces; the results are presented in Fig. 5. In this experiment, we concluded that: (1) With a connected peripheral, the optical leakage from a device can be captured from a distance. In our case, the optical leakage can be captured from the power LED of the USB hub from a distance of 15-35 meters (depending on the connected device). (2) A telescope with a greater lens diameter yields a higher SNR due to the fact that it can capture more light from the device's power LED.

E. Influence of Ambient Light

Next, we examine how the optical SNR is affected by ambient light in two types of optical data acquisition: (1) close data acquisition in which the photodiode is placed 2 cm away from the power LED, and (2) remote data acquisition in which the photodiode is mounted to a telescope.

Experimental Setup. We connected a USB hub (Gold Touch 8 Ports USB3.0 Slim HUB) to a Raspberry Pi 3B+. Note that we obtained optical traces from the USB hub and not from the Raspberry Pi, because the Raspberry Pi's power LED is too weak to be captured from a distance greater than 5 meters, whereas the USB hub provides a higher optical SNR. We executed the *Prober* code on the Raspberry Pi and obtained optical traces from the power LED of the connected USB hub in three environmental settings: a dark room (0 lux), a room lit by fluorescent tubes (365 lux), and a sunlit room (2500 lux). In each of the three settings, two optical traces were collected, by directing the photodiode at the power LED from 2 cm away and by mounting the photodiode to a telescope (Explore Scientific ED102) located 10 meters away from the USB hub.

Results. The FFT graphs of the optical traces obtained in the three ambient light conditions (darkness, sunlight, and fluorescent light) via a telescope placed 10 meters away from the power LEDs are presented in Fig. 6. As can be seen from the FFT graphs, the spectral behavior of the traces obtained in darkness and sunlight is similar and contains one significant peak around 51 kHz (the result of the optical leakage, which is

TABLE III
THE INFLUENCE OF AMBIENT LIGHT AND DATA ACQUISITION ON THE OPTICAL SNR.

	Ambient Light		
	Darkness	Room Lightning (Fluorescent)	Sunlight
Data Acquisition	0 Lux	365 Lux	2500 Lux
Close (2 cm)	37.1 dB	37.6 dB	37.2 dB
Remote via a telescope (10 meters)	21.67 dB	22.19 dB	21.69 dB

associated with the activity triggered by the *Prober* code). The spectral behavior of the optical trace obtained in fluorescent light differs from the other two traces and contains additional noise (unrelated to the CPU activity) around 87 kHz and 174 kHz, which is associated with the subtle changes in the intensity of the light emitted from the fluorescent tubes. In this experiment, we concluded that artificial ambient light (e.g., produced by fluorescent tubes) that affects the spectral behavior of the optical trace may influence an attacker's ability to perform cryptanalysis. This can happen in cases in which the activity triggered by the cryptographic computations affects the same frequencies as those affected by the optical noise added to the spectrum by the artificial ambient light.

Next, we analyzed the optical traces and calculated the SNR around 51 kHz (the frequency that was affected by the activity triggered by the *Prober* code). Table III presents the results. Here, we concluded that when the frequencies in the optical spectrum that are affected by the CPU activity (in our case, 51 kHz) do not intersect with the noise added to the optical spectrum by the artificial light produced by light bulbs (in our case 87 kHz and 174 kHz), the level of the ambient light does not affect the optical SNR in close data acquisition (in this case, there is a change of up to 0.5 dB in the SNR, which is a reasonable sampling error) or remote data acquisition (there is a change of up to 0.48 dB in the SNR, which again is a reasonable sampling error).

F. Influence of Glass Placed Between the Photodiode and the Power LED

Next, we examine how the optical SNR is affected by the presence of glass (i.e., a window) placed between the photodiode and a device's power LED.

Experimental Setup. We connected the Raspberry Pi 3B+ to the USB hub and mounted the photodiode to a telescope (Explore Scientific ED102) and placed the telescope in three locations (30 cm, 3.5 meters, and 7 meters away from the USB hub). We compare the SNR obtained in three settings: (1) the baseline setting where there is no window between the photodiode and the power LED, (2) when there is a window consisting of single-layer transparent glass between the photodiode and the power LED, and (3) when there is a window consisting of double-layer transparent glass between the photodiode and the power LED. We executed the *Prober* code on the Raspberry Pi while obtaining optical traces from the power LED of the connected USB hub from the three distances in the three settings via a telescope.

Results. We calculated the SNR around the affected frequencies of the optical traces; the results are presented in Fig. 7). For the distances examined, the SNR in the setting with a window consisting of single-layer transparent glass decreases by ∼1.5-5.5 dB (compared to the baseline setting), while the SNR in the setting with a window consisting of double-layer transparent glass decreases by ∼4-9 dB (compared to the baseline setting). Based on this experiment, we concluded that the presence of a window consisting of double-layer transparent glass between the photodiode and the power LED (when obtaining optical traces remotely) decreases the optical SNR of the optical traces by one order of magnitude due to the double scattering of the light that is caused by the double-layer glass.

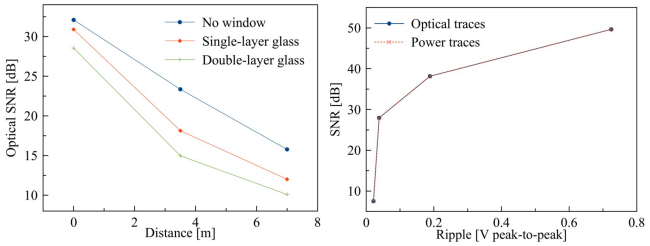

Fig. 7. The influence of a window placed between the photodiode and power LED on the optical SNR (left) and the influence of the ripple of the power supply on the SNR of the optical and power traces (right).

G. Influence of the Power Supply's Ripple

Next, we examine how the optical SNR is affected by the ripple (peak-to-peak variations) of the power provided by the power supply.

Experimental Setup. We conducted four experiments, each using a different type of power supply: the original power supply of the Raspberry Pi: DSA-13PFC-05 (5.1V, 2.5A); two additional types of 5V power supplies: HNT-S520 (5V,2A) and TC09iG (5V,1A); and a professional dual DC power supply: DHR-3652 (40V,3A). The power supplies are presented in Fig. 16 in Appendix A. In each experiment, we connected a different power supply to the same Raspberry Pi 3B+ and executed the *Prober* code. We obtained power traces from the power supply by connecting pin 2 (the input voltage) of the Raspberry Pi to the ADC (to avoid causing an affect on the Raspberry Pi's power consumption). In parallel, we obtained optical traces from the photodiode that was directed at the Raspberry Pi's power LED.

Results. The four power traces were used to compute the ripple of the associated power supplies by calculating the peak-to-peak voltage. For the four calculated peak-to-peak values, we computed the associated optical SNR from the associated optical trace; the results are presented in Fig. 7). As can be seen from the results, the SNR calculated from the power traces is similar to the SNR calculated from the optical traces. We concluded that the ripple of the power supply greatly affects the optical SNR, as higher peak-to-peak variation in the voltage of the power supply yields a higher SNR in the power traces, which in turn results in a higher SNR in the optical traces.

TABLE IV
COMPARISON OF THE OPTICAL SNR OBTAINED FROM VARIOUS DEVICES RUNNING THE VULNERABLE CRYPTOGRAPHIC LIBRARIES TARGETED BY [11–13].

	Devices	
	Raspberry Pi 3B+	Samsung Galaxy S8
Libgcrypt 1.8.4	30.7	24.8
PQCrypto-SIDH 3.4	32.5	26.6
GnuPG 1.4.13	23.5	-

H. Influence of the Cryptographic Library

Next, we examine how the optical SNR is affected by the cryptographic library installed on the target device.

Experimental Setup. We compare the optical SNR obtained from the cryptographic computations performed by three cryptographic libraries installed on a Samsung Galaxy S8 and Raspberry Pi 3B+: (1) Libgcrypt, (2) GnuPG, and (3) PQCrypto-SIDH. We obtained optical traces by directing the photodiode toward the two devices' power LEDs while replicating three cryptanalytic side-channel attacks, aiming to recover a 256-bit ECDSA key from Libgcrypt 1.8.4 (replicating the attack in [11]), a 378-bit SIKE key from PQCrypto-SIDH 3.4 (replicating the attack in [13]), and a 4096-bit RSA key from GnuPG 1.4.13 (replicating the attack in [12]).

Results. We calculated the optical SNR for the six optical traces; the results are presented in Table IV. As can be seen in the results presented in Table IV, the target library under attack greatly affects the optical SNR, due to the fact that: (1) there is a difference in the SNR obtained from the Raspberry Pi, depending on the library used (there is up to ∼7.2 dB variation in the SNR), and (2) in the case of the Samsung Galaxy and GnuPG, the optical traces do not contain any leakage associated with the attack (i.e., the attack in [12] cannot be applied optically against GnuPG 1.4.13); we note, however, that leakage associated with the attack does not appear in the power trace we collected for this experiment, and therefore this is not a limitation of the optical channel.

VI. RECOVERING ECDSA KEYS

In this section, we recover a 256-bit ECDSA private key from various devices.

As observed in the original papers on the Minerva attack [11] and TPM-FAIL [27], many cryptographic libraries optimize the ECDSA signing computation time by running a variable number of loop iterations (which is determined by the number of leading zeros in the nonce) instead of a fixed number of iterations. As a result, the signing times of a set of ECDSA signatures can be used to extract the target's private key by using lattice techniques (the signatures whose nonces have many leading zeros are used to construct a hidden number problem, which is reduced to a shortest vector problem and solved using lattice reduction; see [11] for details).

A. Identifying ECDSA Operations

First, we show that optical traces obtained from the power LED of various devices can be used to distinguish between different ECDSA signatures using spectral analysis. We targeted

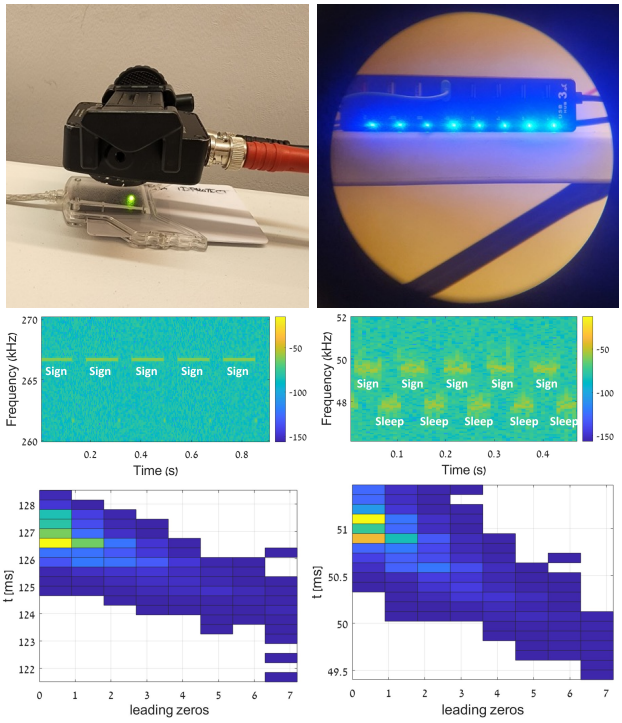

Fig. 8. Recovering the ECDSA keys. Experimental setup (top): A photodiode is directed at the power LED of a card reader (from 2 cm away) and a view of the Gold Touch 8 Ports USB3.0 Slim HUB (which is connected to the Raspberry Pi) through the telescope placed 25 meters away. Optical leakage (middle): The associated spectrograms extracted from optical traces during ECDSA sign operations followed by sleep. Results (bottom): The associated heat maps of the estimated execution times of ECDSA signatures as a function of the number of leading zero bits in the nonce.

two devices: (1) a Raspberry Pi 3B+ (with the Libgcrypt 1.8.4 library installed), powered by a USB hub splitter (Gold Touch 8 Ports USB3.0 Slim HUB), and (2) an Athena IDProtect smartcard inserted into a card reader which was connected to a laptop via a USB cable. We note that the ECDSA implementation that we targeted in Libgcrypt 1.8.4 is the secp256r1 curve (with random ECDSA nonces and hash function SHA-256). We do not know which cryptographic library is installed on the smartcard.

The experimental setups for the two experiments described in this section, which were performed in order to obtain optical traces from the power LED of each device, are summarized in Table V and can be seen in Fig. 8 (top). We note that the primary difference between the two experimental setups was the distance: We obtained the optical traces from the power LED of the USB hub from a distance of 25 meters (by mounting the photodiode to a telescope), and we obtained the optical traces from the power LED of the smartcard reader from 2 cm away.

We conducted two experiments, one for each device. In each experiment we obtained optical traces during five sign operations that were performed on random messages and were separated by sleep operations of 40 ms.

Results. Fig. 8 (middle) presents the spectrograms extracted from the two optical traces. The optical leakage associated with the sign operations appears at around 49.6 kHz for the Raspberry Pi and around 266.6 kHz for the smartcard

TABLE V
EXPERIMENTAL SETUP FOR THE EXPERIMENTS DESCRIBED IN SECTIONS VI-VIII

	Recovered Key	**256-bit ECDSA Key**				**4096-bit RSA Key**	**378-bit SIKE Key**
	Section	Appendix		Section VI		Section VII	Section VIII
Exp. Setup - Victim's Side	Library	Libgcrypt 1.8.4			Unknown	GnuPG 1.4.13	PQCryptoSIDH 3.4
	Victim Device (that contains the key)	Raspberry Pi 3B+	Raspberry Pi 3B+	Raspberry Pi 3B+	Athena ID Protect smartcard	Raspberry Pi 4B	Samsung Galaxy S8
	Victim Power LED		A connected Gold Touch 8 Ports USB3.0 Slim HUB	A connected RUNMUS K8 gaming headset	card reader		
Exp. Setup - Attacker's Side	ADC (model)	NI-9223	NI-9223	NI-9223	NI-9223	NI-9223	NI PCI-6115
	Photodiode (model)	PDA100A2					
	Photodiode Internal Gain	40 dB					10 dB
	Amplifier External Gain	Operational/Analog Amplifier 50 dB					
	Sampling Rate	1 MHz	1 MHz	1 MHz	1 MHz	200 kHz	5 MHz
	Distance	2 cm	25 meters	2 cm	2 cm	2 cm	2 cm
	Telescope	-	Sky-Watcher Flextube 350P	-	-	-	-
Properties of the Attack	Cryptanalytic Attack	Minerva				Acoustic Cryptanalysis	Hertzbleed
	Number of Traces Collected	7,000 (used: 5,325 filtered: 1,675)	22,000 (used: 6,051 filtered: 15,949)	11,000 (used: 5,743 filtered: 5,257)	7,000 (used: 6,862 filtered:138)	1,024 (20 additional traces for error detection)	378 (each consists of 400 SIKE operations)
	Affected Frequencies	Sign: 99.5 kHz Sleep: 87.5 kHz	Sign: 49.6 kHz Sleep: 47.8 kHz	Sign: 49.6 kHz Sleep: 47.8 kHz	Sign: 266.6 kHz	22-26.5 kHz	Decapsulate: 1.21 MHz Sleep: 1.17 MHz
	Optical SNR [dB]	24 dB	24.9 dB	25.6 dB	30 dB	26.4 dB	28.5 dB

reader. Interestingly, while the optical leakage associated with the sleep operations affected the 47.8 kHz frequency in the optical trace obtained from the USB hub, the optical leakage associated with the sleep operations did not affect any specific frequency in the optical trace obtained from the smartcard reader. In both cases, the sign operations can be distinguished using spectral analysis.

B. ECDSA Key Recovery

We now demonstrate the recovery of a 256-bit ECDSA key by applying the Minerva attack against the two devices. We wrote code that triggers ECDSA sign operations followed by a 40 ms sleep operation. In the first experiment, we obtained optical traces of 22,000 different ECDSA sign operations performed by the Raspberry Pi from a distance of 25 meters from the power LED of the USB hub. In the second experiment, we obtained optical traces of 7,000 different ECDSA sign operations performed by the smartcard from a distance of 2 cm from the power LED of the smartcard reader.

Signal Processing. First, we describe the technique we used to process the optical traces obtained from the USB hub; later in this section we describe the modifications we made to the technique for its use on the optical traces obtained from the smartcard reader. We performed short-time Fourier transform (STFT) with a window of 7.5 ms and an overlap of 50% on each trace. We divided the optical trace, which consisted of 22,000 sign operations, into short traces (each associated with one sign operation), by identifying the sections in which the magnitude of the frequency bin associated with the sleep operation ($\sim$47.8 kHz) is greater than the magnitude of the bin associated with the sign operation ($\sim$49.6 kHz) for at least 30 ms (the sleep operations were 40 ms long). Each of the 22,000 short traces consisted of a sign operation followed by 5-10 ms of sleep, since we added an extra window at the beginning and end of each trace.

Next, for each trace we performed the following steps to estimate the execution time: (1) We applied STFT with a window of 0.75 ms and an overlap of 90% on the traces. (2) We located the sign operation in the trace, by detecting the sections in the trace in which the magnitude of the bin associated with the sign operation ($\sim$49.6 kHz) is greater than the magnitude of the bin associated with the sleep operation ($\sim$47.8 kHz). (3) Because of errors, more than one section can satisfy the abovementioned condition (i.e., two or more different sections that are separated by some windows are located in the same trace). In order to correct such errors, we used an error tolerance threshold of two windows (i.e., if two consecutive sections were separated by only two windows, we merged them into a single section). (4) Traces that yielded more than one sign section that satisfied the abovementioned condition were filtered out after the second step. (5) The estimated execution time of the sign operation was calculated based on traces with a single section associated with an ECDSA operation. After filtering, 6,051 signatures remained, along with the estimates of their ECDSA execution time from the optical traces obtained from the USB hub.

We used the same technique described above to process the optical traces obtained from the 7,000 sign operations from the smartcard reader's power LED, with one minor modification. Since the sleep operations of the optical traces were not mapped to a specific frequency in the spectrum (as opposed to the case of the USB hub), we detected the sleep operations based on the magnitude associated with the frequency of the sign operations (266.6 kHz); a low magnitude is indicative of the sleep operations, and a high magnitude is indicative of the sign operations. After filtering, 6,862 signatures remained, along with the estimates of their ECDSA execution time from the optical traces.

Results. The heat maps in Fig. 8 (bottom) show the estimated execution time of the ECDSA signatures (based on the optical traces) vs. the number of leading zeros in the nonces of

the signatures for each device. As can be seen, the signatures that were estimated to have the shortest ECDSA operation execution time (based on the optical traces) have nonces with many leading zeroes (as expected).

We used Minerva's cryptanalysis script [28] to perform lattice-based key extraction on each device by creating two datasets (one for the Raspberry Pi and one for the smartcard). Each dataset consists of the associated ECDSA public key, messages, signatures, and corresponding ECDSA execution times, as estimated from the optical traces obtained in the associated experiment. We executed Minerva's cryptanalysis script twice on a laptop computer (once for each dataset associated with a device), and recovered the two 256-bit ECDSA secret keys (in approximately two minutes per execution).

In Section XIII (the appendix), we demonstrate the direct application of the Minerva attack, targeting a Raspberry Pi 3B+, by obtaining optical traces from its power LED, and an indirect application of the attack, by obtaining optical traces from the power LED of USB headphones that were connected to the Raspberry Pi 3B+.

VII. Recovering RSA Keys

In this section, we describe the recovery of a 4096-bit RSA signing key from a Raspberry Pi.

In [7, 12], the authors demonstrated a 4096-bit RSA key extraction attack from GnuPG 1.4.13, by decrypting a series of adaptively chosen ciphertexts, where each cipher is used to recover a single bit of the RSA secret prime q (or p), starting with the most significant bit (MSB). For each bit q_i of q, the attacker crafts a cipher that will reveal the i-th bit (q_i) when decrypted by the target. The acoustic fingerprint was then used by the attacker to determine the value of the bit by using spectral analysis (see [7, 12] for more details).

A. Identifying RSA Operations

First, we show that optical traces obtained from the power LED of a Raspberry Pi 4B (with the GnuPG 1.4.13 library installed) can be used to distinguish between different RSA decryptions by using spectral analysis. We drew a private and public key (n = 4096) and executed code that triggers three RSA decryptions with three different ciphers followed by sleep operations of 300 ms and obtained an optical trace from the power LED of the Raspberry Pi 4B during the code's execution. Table V summarizes the experimental setup used to obtain the optical trace/s in the experiments described in this section; the setup can also be seen in Fig. 9.

Results. Fig. 9 presents the spectrogram extracted from the optical trace. The optical leakage associated with the decryptions of the three ciphers appears around 23 kHz and is separated from the optical leakage associated with the sleep operations which appears around 47 kHz. Clearly, the three RSA decryptions can be distinguished from sleep operations by using spectral analysis of the optical trace.

B. Distinguishing Bits of the Secret Prime

We now show that the two cases of $q_i = 0$ and $q_i = 1$ yield different optical fingerprints which can be used to distinguish between 0/1 bits of q. An index of the prime number q containing a zero bit and another index containing one bit were selected. We created a shell script which executes decryption operations on the two ciphers needed to attack the two indexes of the prime number q (the ciphers were created according to the details provided in [12]). The two optical traces from the Raspberry Pi's power LED were obtained while we executed the code and triggered decryptions.

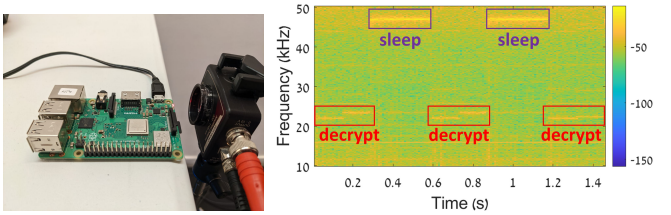

Fig. 9. Recovering RSA keys. Left: A photodiode directed at the power LED of a Raspberry Pi 4B. Right: A spectrogram with three RSA decryption operations.

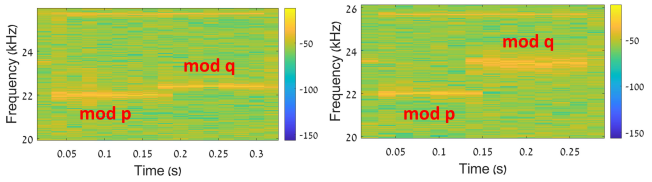

Fig. 10. Spectrograms extracted from optical traces obtained from the power LED of a Raspberry Pi 4B when the zero bit ($q_i = 0$) is under attack (left) and when the one bit ($q_i = 1$) is under attack (right).

Results. Fig. 10 presents the spectrograms extracted from the optical traces obtained. Clearly, the optical fingerprints when the attacked bit is $q_i = 0$ vs. $q_i = 1$ can be distinguished based on their spectral behavior.

C. RSA Key Recovery

We now recover a 4096-bit RSA key by recovering the bits of q using an adaptively chosen cipher attack.

Experimental Protocol and Signal Analysis. The recovery of bits of prime number q was performed based on q_i (the index under attack):

q_{2047} For the MSB, we created the profiles for a zero bit and one bit under attack. The profiles for $q_i = 0$ and $q_i = 1$ were created by obtaining the optical traces of the decryptions of the ciphers $0xfff..fff$ and $0x7ff...fff$, respectively (the same ciphers used in the original paper [12]). The value of q_{2047} was determined to be one, since we know that the MSB of prime number q is set to one in order to ensure a high prime number

q_{2046} - q_{2027} The ciphers were created by placing the $i-1$ MSB bits recovered in the MSB indexes of the cipher C_i, setting the ith bit of C_i to zero, and setting the remaining LSB (least significant bit) indexes of C_i to one. For each q_i, we triggered the decryption of C_i and obtained the optical trace. We extracted the signal that was associated with the decryption operation from the trace and appeared between the two sleep operations, by identifying the section between the optical leakage associated with the sleep operations (47

kHz). We divided this signal into two halves. The analysis was performed on the second half of the signal, which is associated with the modulo q exponentiation. We extracted $argmax_{22-26.5KHz}(FFT_{C_i})$, which is the frequency with the highest magnitude in the second half of the signal in the spectrum of 22-26.5 kHz (the spectrum associated with leakage of q). We also extracted $argmax_{22-26.5KHz}(FFT_{P_0})$ and $argmax_{22-26.5KHz}(FFT_{P_1})$, the frequencies with the highest magnitude in the zero (P_0) and one profile (P_1) associated with the last index that was classified respectively as zero/one. We determined the value of q_i by finding the profile (among P_0 and P_1) closest to $argmax_{22-26.5KHz}(FFT_{C_i})$ using the Euclidean distance.

q_{2026} - q_{1911} We created the ciphers and extracted the signals associated with the modulo q exponentiation using the process described above for the case of q_{2046} - q_{2027}. We calculated $argmax_{22-26.5KHz}(FFT_{P_0})$ and $argmax_{22-26.5KHz}(FFT_{P_1})$ by averaging the $argmax$ values of the signals of the last five bits associated with the corresponding zero/one bit. The value of q_i was determined as described above for the case of q_{2046} - q_{2027}. Note that we stopped this operation after determining the value of q_{1911}, because the Euclidean distance between the two profiles in index 1911 decreased to 250 Hz which we consider a low margin for the decision.

$q_{1910} - q_{1789}$ The ciphers were created by placing the $i-1$ MSB bits recovered in the MSB indexes of the cipher C_i, setting the ith bit of C_i to one and setting the remaining LSB indexes of C_i to zero. We extracted the signals associated with the modulo q exponentiation as described above for the case of q_{2046} - q_{2027}. Note that in this case, we created the zero/one profiles by decrypting the bits in indexes $1911-1931$ with 20 new ciphers that we created as described in this case (the bits of those indexes were already extracted, as described above for the case of q_{2026} - q_{1911}). We determined the value of q_i as described for q_{2046} - q_{2027}. Note that we stopped this operation after determining the value of q_{1789}, because the Euclidean distance between the two profiles (P_0 and P_1) in index 1789 decreased to 250 Hz which we consider a low margin for the decision.

$q_{1788} - q_{1024}$ We created the ciphers and extracted the signals associated with the modulo q exponentiation as described above for the case of q_{2046} - q_{2027}. Note that we created the zero/one profiles (P_0 and P_1) by decrypting the bits in indexes $1809 - 1789$ with 20 new ciphers that we created as described for q_{2046} - q_{2027} (the bits of those indexes were already extracted as described above for the case of $q_{1910} - q_{1789}$). We calculated $argmax_{22-26.5KHz}(FFT_{P_0})$ and $argmax_{22-26.5KHz}(FFT_{P_1})$ by averaging the $argmax$ values of the signals of the last five bits associated with the corresponding zero/one bits. We determined the value of q_i as described above for the case of q_{2046} - q_{2027}.

Results. We recovered the 1,024 most significant bits of the secret q (with just one error that we detected and corrected using a dedicated error detection algorithm) which is sufficient for recovering the entire RSA private key via Coppersmith's attack [29, 30].

VIII. Recovering SIKE Keys

In this section, we recover a 378-bit Supersingular Isogeny Key Encapsulation (SIKE) key from a Samsung Galaxy S8.

As seen in the Hertzbleed attack [13], the SIKE implementation in PQCrypto-SIDH leaks information regarding the bits of the SIKE key due to an Intel mechanism, dynamic voltage and frequency scaling (DVFS), which under certain circumstances can be exploited by an attacker to induce variations in the CPU frequency by overloading the CPU with computations. This results in differences in the execution times associated with the data processed; these differences can be amplified to a distinguishable level (at a granularity of milliseconds) by overloading the CPU using a large number of operations executed in parallel (see [13] for details).

The Hertzbleed key extraction attack targets the static secret key, an integer m with bit expansion $m = (m_{l-1}, ..., m_0)_2$, where $l = 378$ (for SIKE-751). During the decapsulation operation, the code computes $P+[m]Q$ for elliptic curve points P and Q included in the ciphertext, using the Montgomery three-point ladder. Based on m_0,...,m_{i-1} (the i-th LSBs of m), an attacker can construct points P and Q such that if $m_i \neq m_{i-1}$, then the $(i+1)$st round of the Montgomery three-point ladder produces an anomalous zero value. Once that anomalous zero value appears, the decapsulation algorithm gets stuck, and every intermediate value produced for the remainder of the ladder is zero. If $m_i = m_{i-1}$, or if the attacker was wrong about the i-th LSB of m when constructing the challenge ciphertext, then the $(i+1)$ round generates a non-zero value. Heuristically, the remainder of the computation proceeds without producing an anomalous zero value (except with negligible probability).

When $m_i \neq m_{i-1}$ and the decapsulation algorithm gets stuck, repeatedly producing and operating on zero values, the processor consumes less power and runs at a higher steady-state frequency (and therefore decapsulation takes a shorter amount of time). Hertzbleed exploits this observation and amplifies the effect of the time difference to recover bits by triggering a large fixed number of encapsulation operations for the secret key's bit under attack and determining whether $m_i = m_{i-1}$ or not, based on a timing threshold.

A. Identifying SIKE Operations

First, we show that the optical leakage from the power LED of a Samsung Galaxy S8 (with the PQCrypto-SIDH 3.4 [31] library installed) can be used to distinguish between different SIKE decapsulation operations by analyzing optical traces in the frequency domain. The experimental setup is summarized in Table V and can be seen in Fig. 11. We ran 800 SIKE operations, which were divided into eight iterations; in each iteration, 100 SIKE operations were executed with the same private key on 100 threads spawned concurrently (as done in the case of Hertzbleed [13]). Every four iterations were followed by a one second sleep operation.

Results. Fig. 11 presents the spectrogram extracted from the optical trace. The optical leakage associated with the 800 SIKE operations executed over eight iterations appears around 1.21 MHz and is separated from optical leakage associated

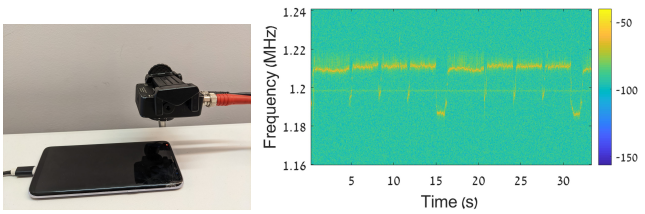

Fig. 11. Recovering SIKE keys. Left: Experimental setup. A photodiode is directed at a Samsung Galaxy S8's power LED. Right: Spectrogram extracted from an optical trace during eight consecutive iterations (each with 100 SIKE operations).

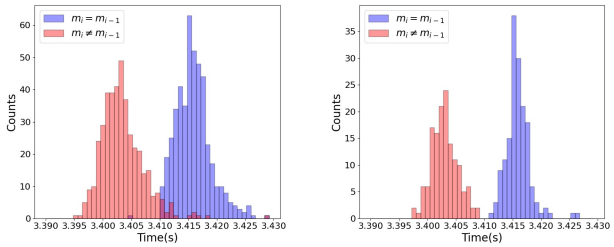

Fig. 12. Left: A histogram of the execution time of each iteration calculated based on 912 iterations (each iteration consists of 100 encapsulation operations). Right: A histogram of the 304 median values of the same execution times (the median value of the three iterations is presented).

with the sleep operations which appears around 1.187 MHz. Clearly, the SIKE iterations can be distinguished from sleep operations.

B. Distingushing $m_i \neq m_{i-1}$ and $m_i = m_{i-1}$

We now examine whether the behavior (the time difference) on the x86 architecture reported in the original paper on Hertzbleed [13] is also seen on the ARM architecture of the Samsung Galaxy S8. We downloaded the code published in the Hertzbleed repository [32], installed the code on the Samsung Galaxy S8, and used it to examine whether the time difference is observable on the smartphone. We analyzed the Samsung Galaxy S8's CPU's execution time for each decapsulation operation. In our experiments, we used the same eight SIKE-751 keys used by the authors of the Hertzbleed paper. For each key $m = (m_{l-1}, ..., m_0)_2$, we uniformly targeted 38 bit positions: 7, 17, 27,...,377. For each of the bit positions, we executed a series of 400 SIKE operations, divided into four iterations, where in each iteration 100 SIKE operations were executed on 100 threads spawned concurrently. Overall, we executed 121,600 SIKE operations that consisted of 1,216 iterations (each of which consists of 100 SIKE operations).

Results. For each bit, we only used the last three iterations (which consist of 300 SIKE decapsulation operations) and disregarded the first iteration (which consists of 100 SIKE decapsulation operations), since we found that the first iteration is unstable and is mainly used to overload the CPU in order to trigger stable execution differences associated with the data processed in the next three iterations. As a result, 25% of the measurements were filtered out; then, the execution time for each of the 912 iterations was calculated. The distribution of the 912 iterations' execution times calculated from the CPU measurements is presented in Fig. 12. The execution times in red represent cases of a switch ($m_i \neq m_{i-1}$), with a mean = 3.405 and STD = 0.0111, and the execution times in blue represent cases of a non-switch ($m_i = m_{i-1}$), with a mean = 3.417 and STD = 0.0106.

Clearly, the behavior (the time difference) reported in the Hertzbleed paper [13] on the x86 architecture is also observable on the ARM architecture at the granularity of a series of 100 consecutive operations, with a threshold of 3.41 seconds that differentiates the switch cases from the non-switch cases. However, as can also be seen in Fig. 12, a negligible number of 35 SIKE iterations (which corresponds to 3.8% of the iterations) crossed the threshold (3.14 seconds), meaning that an algorithm used to distinguish between the two cases based on this threshold will misclassify some bits. In order to handle the expected errors, we calculated the median execution iteration time of the three iterations associated with each bit. The histogram created from the 304 median values of the iterations is presented in Fig. 12 and differentiates the switch cases from the non-switch cases without any errors.

C. SIKE Key Recovery

Finally, we demonstrate the recovery of a 378-bit private key from the SIKE-751 implementation, using optical traces obtained from a Samsung Galaxy S8's power LED, in a series of adaptively chosen ciphertext attacks. The experimental setup is summarized in Table V and can be seen in Fig. 11.

Experimental Protocol. For each index i of the private key we wanted to recover, we created a dedicated input M_i (as described in the paper presenting Hertzbleed [13] using the $i-1$ bits already recovered). We used M_i to trigger 400 SIKE operations, which were divided into four iterations, where in each iteration 100 consecutive SIKE operations were triggered with M_i and executed using 100 threads. This process was repeated iteratively for all 377 indexes; first we calculated the value of the ith bit, and then we created M_{i+1}.

Processing the Signal. We divided the optical trace that consists of the four iterations (which were used to recover the i-th bit) into four traces based on the leakage, which is associated with the transitions from a higher to lower frequency between each iteration (this appears around 1.21 MHz, as seen in Fig. 11). Based on the observation we made earlier, we only used the last three traces associated with the last three iterations (consisting of 300 SIKE decapsulation operations) and disregarded the first trace associated with the first iteration (consisting of 100 SIKE decapsulation operations). For each of the three traces, we applied STFT and estimated the execution time of the associated iteration by analyzing the STFT bin associated with the SIKE optical leakage around 1.21 MHz (see Fig. 11).

Results. First, we note that we guessed that the value of the first index of the key (where $j = 0$) would be zero. According to the Hertzbleed paper, an incorrect guess/prediction of the value of the key in any index n (where $0 \leq n \leq 377$) will create $377 - n$ consecutive non-switch cases (i.e., no anomalous zero will appear from this point on). In our case,

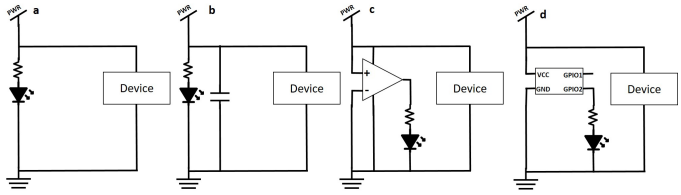

Fig. 13. Circuits that leak information via their power LED (a). Countermeasures using a capacitor (b), an additional OPAMP (c), and an existing OPAMP (d).

we verified that our guess for the first index was correct by using the next bit index (where $j = 1$) which was predicted as a switch case. For each index j (where $1 \leq j \leq 377$), we calculated the median value from the three estimated execution times. A threshold of 3.41 seconds was used to distinguish between the switch and non-switch cases (determined based on the median values). We note that the median values estimated for two of the predicted indexes (indexes 274 and 276) were within the range of 3.409-3.411. Since we considered those median values as vulnerable to errors (due to their proximity to the threshold), we resampled those indexes again by repeating the process described above for the corresponding bits, i.e., we triggered an additional 400 SIKE operations for each bit (as described above) and used the new measurements instead of the previous measurements. The 378 bits of the key were recovered without any errors.

IX. Countermeasures

Manufacturer-Side Methods. In many devices, the power LED is connected directly to the power line of the PCB (see Fig. 13a). As a result, the device's power LED is affected by the power consumption fluctuations that occur when cryptographic operations are performed. To counter this phenomenon, a few approaches should be considered by hardware manufacturers: (1) Using a capacitor: A capacitor can be integrated parallel to the power LED indicator; in this case, the capacitor would behave as a low-pass filter (see Fig. 13b). This is an inexpensive solution for reducing the fluctuations. (2) Using an operational amplifier (OPAMP): This can be implemented by integrating an OPAMP between the power line and the power LED (see Fig. 13c) or by using an existing GPIO port of an integrated microcontroller as a power supply for the power LED (see Fig. 13d). In both cases, this will eliminate power line AC fluctuations by a factor of the OPAMP amplifier's common mode rejection ratio.

Consumer-Side Methods. The attack can also be prevented by placing black tape over a device's power LED. While this solution decreases a device's UX, it prevents attackers from obtaining traces from vulnerable devices.

X. Limitations

Line of Sight. Attackers must establish a line of sight to a power LED in order to obtain optical traces.

Variable SNR. Various factors may contribute to the deterioration of the SNR of the optical traces: the distance between the photodiode and the power LED (light deteriorates with distance), the power LED of the device under attack (see Table II), the target library under attack (see Table IV), and the type of power supply used (see Fig. 7). We note, however, that attackers can improve the SNR by increasing the sensitivity of the equipment used to obtain the optical traces by adding external amplifiers or using a more sensitive photodiode, a higher resolution ADC, or a telescope and zooming lens to capture more light (see Fig. 5).

Sampling Rate. While LEDs are highly responsive and can provide a high bandwidth [23], in our experiments we found that the photodiode chosen could limit the sampling rate. For example, the $300 photodiode we used (the Thorlabs PDA100A2) supports a maximum sampling rate of just 11 MHz. With this limited sampling rate, we were able to recover keys from devices with CPU rates of up to a few GHz (e.g., Samsung Galaxy S8). A more advanced photodiode is required to recover cryptographic keys from devices with higher CPU rates (e.g., servers and laptops).

Distance. In this study we recovered a secret key from a Raspberry Pi using optical traces obtained from a connected peripheral (i.e., a USB hub) from a distance of 25 meters. However, the optical leakage from the power LEDs of the devices examined in this paper is too weak to be captured from a distance greater than 5 meters. This limits the ability of attackers to recover a secret key from a distance in cases in which a peripheral is not connected to the target device.

XI. Discussion & Future Work

We note that the contribution of our paper relates to the attack vector and not to the discovery of a new cryptographic vulnerability. While demonstrated on known cryptanalytic side-channel attacks, the new attack vector can be used to facilitate new cryptanalytic side-channel attacks.

We disclosed our findings to the manufacturers of the devices used in our study via their bug bounty programs and contact us email addresses. Raspberry Pi and Samsung responded to our email and asked us for more details which we shared with them. We recommend that other hardware manufacturers empirically test whether their devices are vulnerable to optical cryptanalysis and if needed, redesign their electrical circuits (according to the suggestions provided in Section IX) in order to prevent attackers from performing optical cryptanalysis against their devices. We are, however, uncertain whether they will choose to do so, as some solutions may increase the manufacturer's overall cost, decreasing revenue or requiring the manufacturer to increase the product's price. While the cost of our countermeasures might seem negligible, the addition of a component to prevent the attack could cost a manufacturer millions of dollars, since such devices are often mass-produced. Given the cost-driven nature of consumers and the profit-driven nature of manufacturers, mitigations are not always applied. This fact may leave many devices vulnerable to optical cryptanalysis.

In this study, we used the Thorlabs PDA100A2 photodiode which supports a maximum sampling rate of 11 MHz. In future work, we suggest using a high-end photodiode that supports

a sampling rate of a few GHz to examine whether optical leakage can be used to (1) detect a single CPU operation, and (2) detect the Hamming weight of an operand based on a single CPU operation. Future work could also focus on examining the effectiveness of the countermeasures suggested in Section IX on the SNR of the optical traces.

ACKNOWLEDGMENTS

This work was partially supported by the Cyber Security Research Center at Ben-Gurion University of the Negev, the Jacobs Urban Tech Hub at Cornell Tech, the Technion's Viterbi Fellowship for Nurturing Future Faculty Members, the Air Force Office of Scientific Research (AFOSR) under award number FA9550-20-1-0425, the Defense Advanced Research Projects Agency (DARPA) under Award number HR00112390029, the National Science Foundation under grant CNS-1954712, as well as gifts from Cisco and Qualcomm.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government.

REFERENCES

[1] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Annual international cryptology conference*. Springer, 1999, pp. 388–397.

[2] P. Kocher, J. Jaffe, B. Jun, and P. Rohatgi, "Introduction to differential power analysis," *Journal of Cryptographic Engineering*, vol. 1, no. 1, pp. 5–27, 2011.

[3] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, "The em side—channel (s)," in *International workshop on cryptographic hardware and embedded systems*. Springer, 2002, pp. 29–45.

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

[5] D. R. Gnad, J. Krautter, and M. B. Tahoori, "Leaky noise: New side-channel attack vectors in mixed-signal iot devices," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 305–339, 2019.

[6] D. Genkin, L. Pachmanov, I. Pipman, and E. Tromer, "Ecdh key-extraction via low-bandwidth electromagnetic attacks on pcs," in *Cryptographers' Track at the RSA Conference*. Springer, 2016, pp. 219–235.

[7] D. Genkin, A. Shamir, and E. Tromer, "Rsa key extraction via low-bandwidth acoustic cryptanalysis," in *Annual Cryptology Conference*. Springer, 2014, pp. 444–461.

[8] Y. Tsunoo, "Crypt-analysis of block ciphers implemented on computers with cache," *Proc. ISITA2002, Oct.*, 2002.

[9] D. Gullasch, E. Bangerter, and S. Krenn, "Cache games–bringing access-based cache attacks on aes to practice," in *2011 IEEE Symposium on Security and Privacy*. IEEE, 2011, pp. 490–505.

[10] O. Aciiçmez, Ç. K. Koç, and J.-P. Seifert, "On the power of simple branch prediction analysis," in *Proceedings of the 2nd ACM symposium on Information, computer and communications security*, 2007, pp. 312–320.

[11] J. Jancar, V. Sedlacek, P. Svenda, and M. Sys, "Minerva: The curse of ECDSA nonces (systematic analysis of lattice attacks on noisy leakage of bit-length of ECDSA nonces)," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2020, no. 4, pp. 281–308, 2020.

[12] D. Genkin, A. Shamir, and E. Tromer, "Acoustic cryptanalysis," *Journal of Cryptology*, vol. 30, no. 2, pp. 392–443, 2017.

[13] Y. Wang, R. Paccagnella, E. T. He, H. Shacham, C. W. Fletcher, and D. Kohlbrenner, "Hertzbleed: Turning power {Side-Channel} attacks into remote timing attacks on x86," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 679–697.

[14] A. Schlösser, D. Nedospasov, J. Krämer, S. Orlic, and J.-P. Seifert, "Simple photonic emission analysis of AES," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2012, pp. 41–57.

[15] E. Carmon, J.-P. Seifert, and A. Wool, "Photonic side channel attacks against RSA," in *2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2017, pp. 74–78.

[16] A. Schlösser, D. Nedospasov, J. Krämer, S. Orlic, and J.-P. Seifert, "Simple photonic emission analysis of AES," *Journal of cryptographic engineering*, vol. 3, no. 1, pp. 3–15, 2013.

[17] J. Loughry and D. A. Umphress, "Information leakage from optical emanations," *ACM Transactions on Information and System Security (TISSEC)*, vol. 5, no. 3, pp. 262–289, 2002.

[18] M. Guri, B. Zadov, D. Bykhovsky, and Y. Elovici, "Ctrl-alt-led: Leaking data from air-gapped computers via keyboard leds," in *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 1. IEEE, 2019, pp. 801–810.

[19] M. Guri, B. Zadov, A. Daidakulov, and Y. Elovici, "xled: Covert data exfiltration from air-gapped networks via switch and router leds," in *2018 16th Annual Conference on Privacy, Security and Trust (PST)*. IEEE, 2018, pp. 1–12.

[20] M. Guri, B. Zadov, and Y. Elovici, "Led-it-go: Leaking (a lot of) data from air-gapped computers via the (small) hard drive led," in *International conference on detection of intrusions and malware, and vulnerability assessment*. Springer, 2017, pp. 161–184.

[21] B. Nassi, Y. Pirutin, T. C. Galor, Y. Elovici, and B. Zadov, "Glowworm attack: Optical tempest sound recovery via a device's power indicator led," Cryptology ePrint Archive, Report 2021/1064, 2021, https://ia.cr/2021/1064.

[22] B. Nassi, Y. Pirutin, J. Shams, R. Swissa, Y. Elovici, and B. Zadov, "Optical speech recovery from desktop speakers," *Computer*, vol. 55, no. 11, pp. 40–51, 2022.

[23] N. Chi, M. Shi, Y. Zhao, F. Wang, J. Shi, Y. Zhou, X. Lu, and L. Qiao, "Led-based high-speed visible light communications," in *Broadband Access Communication Technologies XII*, vol. 10559. SPIE, 2018, pp. 90–97.

[24] "Pda100a2," https://www.thorlabs.com/thorproduct.cfm?partnumber=PDA100A2.
[25] S. King, "Luminous intensity of an LED as a function of input power," *ISB J. Phys*, vol. 2, no. 2, 2008.
[26] "curve25519-donna.c," https://github.com/agl/curve25519-donna/blob/master/curve25519-donna.c.
[27] D. Moghimi, B. Sunar, T. Eisenbarth, and N. Heninger, "Tpm-fail: Tpm meets timing and lattice attacks," in *Proceedings of the 29th USENIX Security Symposium*, 2020.
[28] "Minerva github," https://github.com/crocs-muni/minerva/tree/master/poc/attack.
[29] D. Coppersmith, "Small solutions to polynomial equations, and low exponent rsa vulnerabilities," *J. Cryptol.*, vol. 10, no. 4, p. 233–260, sep 1997. [Online]. Available: https://doi.org/10.1007/s001459900030
[30] R. L. Rivest and A. Shamir, "Efficient factoring based on partial information," in *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer, 1985, pp. 31–34.
[31] "Pqcrypto-sidh," https://github.com/microsoft/PQCrypto-SIDH, 2019.
[32] "Hertzbleed github," https://github.com/FPSG-UIUC/hertzbleed.

XII. Appendix A: Additional Material

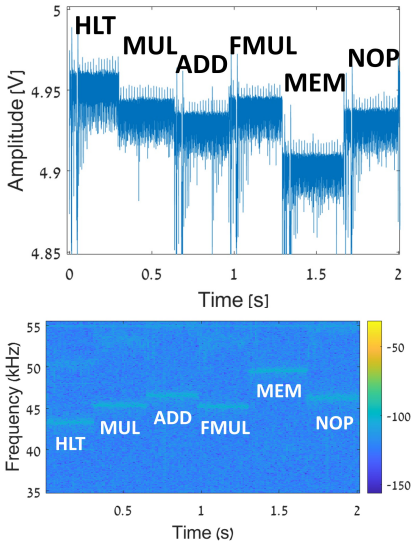

Fig. 14. A power trace obtained from a Raspberry Pi during the execution of six repeated CPU operations visualized in the time domain (top) and in the spectrum (bottom).

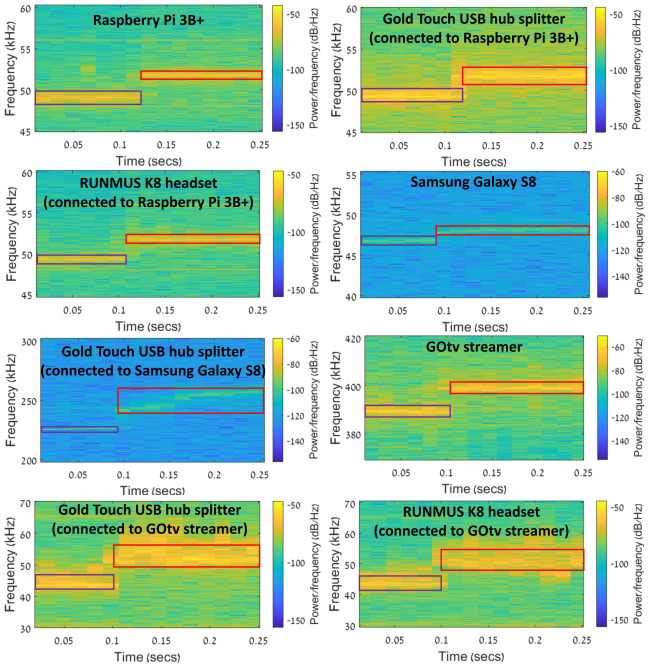

Fig. 15. Spectrograms obtained from various devices while running the *Prober* code that alternates between sleep operations (boxed in purple) and repeated MUL operations (boxed in red).

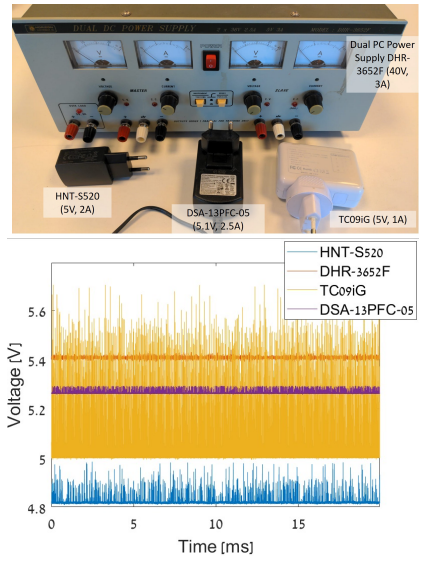

Fig. 16. Top: The four power supplies used. Bottom: The power traces obtained from them.

XIII. Appendix B: ECDSA Key Recovery from Raspberry Pi 3B+ and RUNMUS K8 Gaming Headset

In this section, we demonstrate the application of the Minerva attack [11] to recover a 256-bit ECDSA private key from the implementation of secp256r1 curve (with non-deterministic ECDSA nonces and hash function SHA-256) by obtaining optical measurements from the power LED of a Raspberry Pi 3B+. We also demonstrate this attack by obtaining optical measurements from the power LED of USB headphones (RUNMUS K8 gaming headset) that were connected to the

Raspberry Pi.

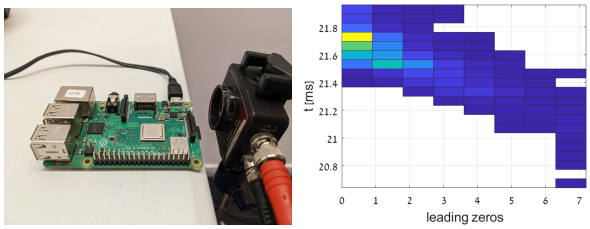

Fig. 17. Recovering the ECDSA key from a Raspberry Pi 3B+. Left: Experimental setup. Right: The estimated execution time of 5,325 ECDSA sign operations (calculated from the optical measurements obtained from the power LED of a Raspberry Pi 3B+) as a function of the number of leading zero bits in the nonce.

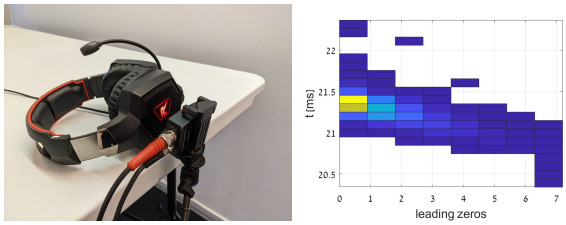

Fig. 18. Recovering the ECDSA key from a RUNMUS K8 gaming headset. Left: Experimental setup. Right: The estimated execution time of 5,743 ECDSA sign operations (calculated from the optical measurements obtained from the power LED of the USB headphones) as a function of the number of leading zero bits in the nonce.

Experimental Setup. The exact experimental setups for the experiments conducted in this section are described in Table V. The photodiode was directed at the power LED of the Raspberry Pi from a range of a few centimeters; the experimental setup can be seen in Fig. 17. We used the same code as in Section VI to obtain optical measurements from 7,000 different ECDSA sign operations performed by the Raspberry Pi 3B+. Then we connected USB headphones (RUNMUS K8 gaming headset) to the Raspberry Pi 3B+ and repeated the same experiment; this time we obtained 11,000 optical measurements from the power LED of the headphones (see Fig. 18).

Processing the Signal. We processed the optical measurements for the Raspberry Pi according to the process described in Section VI , which was applied to process the optical traces that were obtained from the power LED of the USB Hub. After we applied the same process for the optical measurements obtained from the power LED of the Raspberry Pi, we had 5,325 signals. The same process was applied to the optical measurements obtained from the USB headphones. At the end of this process, we had 5,743 signals.

Results. We produced the heat maps presented in Figs. 17 and 18, which show the estimated ECDSA operation time (based on the optical measurements) vs. the number of leading zeros in the signature's nonce for the optical signals obtained respectively from the Raspberry Pi and headphones. As can be seen, the signatures that were estimated to have the shortest ECDSA operation execution time (based on the optical traces) have nonces with many leading zeroes (as expected).

Given these findings, we used Minerva's cryptanalysis script to perform lattice-based key extraction on each device. We used the script provided in the official Minerva GitHub [28] which implements the lattice reduction attack mentioned above. First, we created two datasets (one for the Raspberry Pi and one for the headphones). Each dataset consists of the associated ECDSA public key, messages, signatures, and corresponding ECDSA execution times, as estimated from the optical traces obtained in the associated experiment. We executed Minerva's cryptanalysis script twice on a laptop computer (once for each dataset associated with a device), and recovered the two 256-bit ECDSA secret keys (in approximately two minutes per execution).