Video-Based Cryptanalysis: Extracting Cryptographic Keys from Video Footage of a Device's Power LED

Ben Nassi[1,2], Etay Iluz[2], Or Cohen[2], Ofek Vayner[2], Dudi Nassi[2], Boris Zadov[2], Yuval Elovici[2]

[1]Cornell Tech, [2]Ben-Gurion University of the Negev
bn267@cornell.edu, {nassib, etayil, ora2, ofekvay, nassid, zadov}@post.bgu.ac.il, elovici@bgu.ac.il
Website - https://www.nassiben.com/video-based-crypta

***Abstract*—In this paper, we present *video-based cryptanalysis*, a new method used to recover secret keys from a device by analyzing video footage of a device's power LED. We show that cryptographic computations performed by the CPU change the power consumption of the device which affect the brightness of the device's power LED. Based on this observation, we show how attackers can exploit commercial video cameras (e.g., an iPhone 13's camera or Internet-connected security camera) to recover secret keys from devices. This is done by obtaining video footage of a device's power LED (in which the frame is filled with the power LED) and exploiting the video camera's rolling shutter to increase the sampling rate by three orders of magnitude from the FPS rate (60 measurements per second) to the rolling shutter speed (60K measurements per second in the iPhone 13 Pro Max). The frames of the video footage of the device's power LED are analyzed in the RGB space, and the associated RGB values are used to recover the secret key by inducing the power consumption of the device from the RGB values. We demonstrate the application of *video-based cryptanalysis* by performing two side-channel cryptanalytic timing attacks and recover: (1) a 256-bit ECDSA key from a smart card by analyzing video footage of the power LED of a smart card reader via a hijacked Internet-connected security camera located 16 meters away from the smart card reader, and (2) a 378-bit SIKE key from a Samsung Galaxy S8 by analyzing video footage of the power LED of Logitech Z120 USB speakers that were connected to the same USB hub (that was used to charge the Galaxy S8) via an iPhone 13 Pro Max. Finally, we discuss countermeasures, limitations, and the future of *video-based cryptanalysis* in light of the expected improvements in video cameras' specifications.**

I. Introduction

Video cameras have become one of the most ubiquitous types of sensors used, and today they are integrated in a variety of devices/systems (e.g., smartphones, drones, autonomous vehicles) and deployed as standalone devices/systems (e.g., surveillance/security cameras, webcams) in smart cities, houses, offices, businesses, and other settings. The wide deployment of video cameras in devices and systems enabled the development of new automatic capabilities/functionality, however it has also created new privacy risks resulting from their deployment in residential neighborhoods, urban environments, etc. While many studies analyzed and discussed the risks posed by video cameras to individuals' privacy in the physical world (e.g., spying [1–4], sound/speech eavesdropping [5–7]), little is known about the risks posed by video cameras to information confidentiality in the digital world.

In this paper, we present *video-based cryptanalysis*, a new side-channel cryptanalytic attack that can be performed by attackers to recover a secret key from a target device by obtaining video footage of the target device's power LED using a commercial video camera (e.g., the video camera of a smartphone or an Internet-connected security camera). We show how attackers can exploit video footage of a device's power LED to recover a device's secret key. This is possible because the intensity/brightness of the device's power LED correlates with its power consumption, due to the fact that in many devices, the power LED is connected directly to the power line of the electrical circuit which lacks effective means (e.g., filters, voltage stabilizers) of decoupling the correlation.

We empirically analyze the sensitivity of video cameras and show that they can be used to conduct cryptanalysis because: (1) the limited eight-bit resolution (a discrete space of 256 values) of a single RGB channel of video footage of a device's power LED is sufficient for detecting differences in the device's power consumption which are caused by the cryptographic computations, and (2) the video camera's rolling shutter can be exploited to upsample the sampling rate of the intensity/brightness of the power LED in the video footage to the level needed to perform cryptanalysis, i.e., increasing the number of measurements (sampling rate) of the intensity/brightness of the power LED in video footage by three orders of magnitude from the FPS rate (which produces 60-120 measurements per second) to the rolling shutter rate (which produces 60K measurements per second in the iPhone 13 Pro Max), by zooming the video camera on the power LED of the target device so the view of the LED fills the entire frame of the video footage. By doing so, attackers can use a readily available video camera to perform cryptanalysis remotely instead of the professional dedicated sensors typically used (e.g., a scope, software-defined radio).

First, we show that standard video cameras can detect changes in the power supply to a power LED at a higher frequency than their FPS (frames per second) rate (by exploiting their rolling shutter). Then, we discuss two potential threat models that attackers can use to apply *video-based cryptanalysis*, based on the type of the power LED of the

device: (1) for devices with standard on/off power LEDs, attackers can obtain video footage using their smartphone's video camera (i.e., attackers must have physical access to the target device), and (2) for devices with indicative power LEDs (in which the color of the power LED changes in response to a CPU operation), attackers can obtain the video footage using a nearby compromised Internet-connected video camera (i.e., attackers can apply the attack remotely over the Internet).

Next, we demonstrate *video-based cryptanalysis* and apply two cryptanalytic attacks published in the last few years in order to recover: (1) a 256-bit ECDSA key from a smart card (exploiting the vulnerability presented in [8, 9]). The ECDSA key is recovered by analyzing video footage obtained via an Internet-connected video camera located sixteen meters away from the smart card, (2) a 378-bit SIKE key from a Samsung Galaxy S8 (exploiting the vulnerability presented in [10]). The key is recovered by analyzing video footage obtained by an iPhone 13 Pro Max's video camera of a power LED of speakers that were connected to the same USB hub (which was used to charge the Samsung Galaxy S8). Finally, we raise concern regarding the possibility that a greater number of devices will be exposed to *video-based cryptanalysis* in light of the existing improvements in video camera specs (which include: an increased shutter speed, a wider RGB space, and improved zoom capabilities) and in light of the expected improvements in video cameras specs (based on Moore's Law).

Contributions. (1) We show that the combination of vulnerable cryptographic algorithms (i.e., that are vulnerable to cryptanalytic side-channel attacks) and vulnerable power LEDs (i.e., that their color/brightness leak information) can be exploited by attackers to recover secret keys in a weaker threat model with respect to SOTA works. *Video-based cryptanalysis* relies on video cameras, which are much more common and readily available than the equipment used to conduct cryptanalysis in prior works (e.g., a scope, probe, software-defined radio) and allow attackers to apply the attack in a non-intrusive manner using video footage. (2) Attack vector - We demonstrate two non-intrusive attack vectors (with physical proximity and over the Internet) to apply *video-based cryptanalysis* that can be exploited to perform existing and new cryptanalytic side-channel attacks, depending on the type of device's power LED. (3) Exposure - We show that at least six commercial smart card readers (that we bought on Amazon) and a smartphone leak information that can be exploited to apply *video-based cryptanalysis* directly from their power LEDs or indirectly via the power LEDs of connected peripherals (speakers, USB hubs).

Structure. In Section II, we review related work. The threat model is presented in Section III. In Section IV, we analyze the bandwidth captured by video cameras of power LEDs. In Sections V-VI, we demonstrate the application of video-based cryptanalysis and recover ECDSA and SIKE keys from various devices. In Section VII, we describe countermeasures, and in Section VIII, we discuss limitations. Finally, in Section IX, we discuss our findings and present the responsible disclosure we performed.

II. Related Work

Cryptanalysis. 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 studies. Those studies exploited the variation in a device's power consumption to recover secret keys by measuring a device's power consumption (e.g., [11, 12]) or by measuring other side effects, including EMR leakage (e.g., [13–19]) and acoustic noise (e.g., [20]).

In some studies [21–23] cryptanalysis was performed by capturing near-infrared photons emitted from switching transistors located on the back of field programmable gate arrays (FPGAs) during the execution of a proof-of-concept implementation of cryptographic algorithms. However, the suggested attacks are ineffective against commercial devices, because their electrical circuits 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.

Power LEDs. Discussion regarding the risks posed to information confidentiality stemming from the correlation between the intensity of a power LED to the power consumed by the device [24] began over 20 years ago [25]. However, prior research demonstrating methods capable of exploiting a device's power LED for data exfiltration relied on preinstalled malware [26–28] that actively triggered and controlled a device's LED (e.g., a keyboard [26], router [27], hard drive [28]) in order to establish optical *covert channels*. A more recent study presented a side-channel attack to recover speech from virtual meetings by exploiting the power LED of speakers used in the meeting [29, 30].

Rolling Shutters. A few studies demonstrated sound recovery from video footage by exploiting a video camera's rolling shutter and analyzing the movements of objects in response to nearby sound [5–7].

III. Threat Model & Upsamplng the Sampling Rate

A. Threat Model

In *video-based cryptanalysis*, the attacker recovers secret keys from a target device using video footage of the power LED of the target device (*i.e., a direct attack*) or of the power LED of a connected peripheral (*i.e., an indirect attack*) whose power consumption is also affected by the power consumption of the target device. The attacker exploits the correlation between the intensity/brightness of a device's power LED and the device's power consumption (which is affected by the cryptographic operations performed); this correlation stems from the fact that in many devices, the power LED is connected directly to the power line of the device's electrical circuit which lacks effective means (e.g., filters, voltage stabilizers) of decoupling the correlation. This correlation, which can be detected by analyzing the RGB values of the device's power LED in video footage, is used by the attacker to perform cryptanalysis. In order to achieve a sampling rate that can be used for cryptanalysis, the attacker uses the video camera's rolling shutter to upsample the sampling rate by filling the

entire frame with the LED (a detailed explanation of this is provided later in this section).

Target Device. We assume that a target device is performing cryptographic operations. 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 messages aimed at triggering automatic digital signing. We assume that the target device contains a power LED or is connected to another device/peripheral that contains a power LED (e.g., speakers, USB hub); the power LED of the target device or the connected peripheral is one of type types of LEDs: (1) **A standard on/off power LED (type 1)** - This is the most common type of power LED integrated in devices. In this case, the color of the LED does not change, and it emits light only when the device is turned on. The brightness of the LED changes very slightly in response to the level of power consumption, however these changes are imperceptible to the human eye. (2) **An indicative power LED (type 2)**: This type of power LED is very common in smart card readers, and its color changes in response to triggered cryptographic operations.

Attacker. 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 that were delivered to the target device and intercepted by the attacker, or (khz) sign on a message on behalf of a target device. We assume that the attacker can obtain video footage of the power LED.

Video Acquisition. We consider two types of video footage acquisition models, which are based on the type of power LED. (1) **Close video acquisition** - In this acquisition model, the attacker uses their smartphone's video camera to obtain the video footage. In this case, we assume that the power LED (type 1 or 2, which are described above) of a device or connected peripheral leaks information from its power LED (which correlates with the cryptographic operations). We also assume that the attacker has access to the room in which the target device is located and can use their smartphone to obtain video footage of the power LED of the victim device (or the connected peripheral) while the target device is performing cryptographic operations. (2) **Over the Internet video acquisition** - In this acquisition model, the attacker obtains the video footage using a hijacked Internet-connected security camera. In this case, we assume that the attacker is able to compromise a 360°Internet-connected video camera with an optical zoom which is located near the target device (up to 16 meters away). We assume that the attacker can control the video camera using its API, use it to zoom and record video footage of the power LED of the target device (or the connected peripheral), and exfiltrate the footage over the Internet to the attacker's possession. In this video acquisition model, we also assume that the device consists of an indicative power LED (type 2) and that the differences in the color of the device's power LED triggered by cryptographic operations can be detected from a distance.

Compromising an Internet-connected video camera is a fair assumption considering that various studies that analyzed

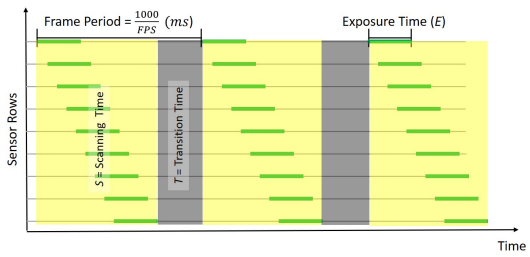

Fig. 1. A rolling shutter of a video camera. In every frame period the rolling shutter scans an object vertically and exposes the shutter for a short time determined by E. The time it takes to scan a single frame is denoted by S. Between two consecutive frames there is a transition period, during which the object is not captured by any frame, which is denoted by T.

IP cameras' security concluded that they are poorly secured against cyber-attacks and allow attackers to remotely hijack and control them over the Internet [31–33]. The poor security level of IP cameras has been exploited in the wild by the famous Mirai and BASHLITE botnets that targeted many Internet-connected video cameras as hosts for their bots [34, 35].

Significance. We note that the threat model is non-invasive (in contrast to power traces which require connecting a scope to the device in order to obtain power measurements), relies on common/ubiquitous equipment (as opposed to other methods that rely on software-defined radios, photodiodes, scopes, probes, etc.), can be applied over the Internet, and in some cases, may endanger devices that do not even contain a power LED via the power LED of connected peripherals (e.g., speakers, USB hub splitters, chargers, and headphones).

B. Increasing a Video Camera's Sampling Rate Using a Rolling Shutter

We note that the FPS rate supported by the vast majority of commercial smartphones and security/IP video cameras is limited to 60-120 FPS which is insufficient for performing cryptanalysis. In order to increase the number of measurements per second (sampling rate) to a level sufficient for cryptanalysis, the attacker can exploit the video camera's rolling shutter.

The rolling shutter is an image-capturing method in which a frame of a video (in video footage) is captured by scanning the scene vertically/horizontally. When this method is used, a frame/picture is not actually composed of a single snapshot of a scene taken at a specific point in time but rather is composed of multiple snapshots taken of vertical/horizontal pieces of the scene at different times. Fig. 1 visualizes this process: With a vertical rolling shutter, a sensor's pixels are exposed and read out row-by-row sequentially at different times from top to bottom (or left to right) according to a configurable shutter speed (E) which determines the amount of time that the sensor is exposed to light. Because each row (or a group of adjacent rows) in a sensor with a rolling shutter is captured at a different time, attackers can increase the sampling rate from the camera's FPS rate (60/120 FPS) to the rate at which rows are recorded, a rate which is based on the shutter speed.

In *video-based cryptanalysis*, attackers exploit the rolling shutter to upsample the number of measurements (sampling

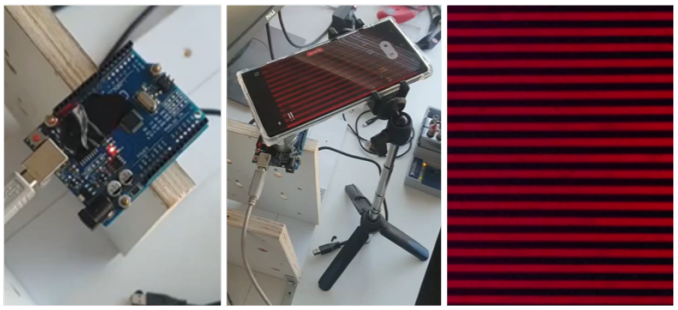

Fig. 2. Upsamling the FPS rate of the video camera to the shutter rate: An Arduino's LED flickering at 4 kHz (left) is recorded by a Samsung Galaxy S22 Ultra using a lens that increases the size of the LED so that it fills the entire screen (middle). A frame of the video recorded by the smartphone that captures the 4 kHz flickering (right).

rate) obtained from the power LED to a higher rate. This is done by setting the rolling shutter of the video camera to its highest speed and zooming the video camera in on the LED, ensuring that the view of the LED fills the entire frame of the video footage. By doing this, the attacker ensures that all of the time it takes to scan a frame (which is denoted as S in Fig. 1) is dedicated to obtaining RGB samples of the power LED. This allows the attacker to upsample the sampling rate by a few orders of magnitude from the FPS rate (60-120 measurements per second) to the approximate shutter rate of the video camera (60K measurements per second in an iPhone 13 Pro Max). We note that the measurements are not ideal, because they are not uniformly sampled across time. As can be seen in Fig. 1, there are transition periods (demoted by T) between frames that are not sampled by the video camera and do not appear in any frame. We consider the rolling shutter sampling as semi-uniform: The sampling is uniform within a frame but is not uniform across the video due to the transitions between frames.

C. Determining the Transition and Scanning Times

We now explain how attackers can empirically determine the transition and scanning time. We note that in some cases, the exact transition time and scanning time may be required to perform some types of cryptanalytic attacks (as demonstrated later in Section V).

Experimental Setup. We programmed an Arduino Uno to modulate a 4 kHz flicker using the Arduino's integrated red LED (using on/off modulation) by turning the power LED on and off every 250 microseconds. We placed a Samsung Galaxy S22 Ultra on the power LED and used a lens so that the entire frame of the video footage would be filled with the view of the LED. We used the smartphone's native camera application and set the video camera's FPS rate at 60 and set the shutter speed at $\frac{1}{12{,}000}$.

Calculating the Scanning and Transition Times. The results of this experiment are presented in Fig. 2. As can be seen, the frame consists of red lines (which indicate that the LED was on) and black lines (which indicate that the LED was off) which result from the flickering LED. This experiment shows that the number of measurements obtained of a power LED by a video camera can be increased by filling the frame with the LED and exploiting the rolling shutter speed. The scanning time can be calculated by multiplying the time that the flicker was on (250 microseconds in our experiment) by the number of transitions between the on/off states in the frame (39 transitions in Fig. 2). In our case, the scanning time is $S = 9.75$ ms. Since $T = \frac{1000}{FPS} - S$, the transition time of the video camera used in this experiment is $T = \frac{1000}{60} - 9.75 = 6.91$ ms.

Note that this process can only be performed by attackers with physical access to a video camera: for example, when an attacker uses their smartphone to perform the attack, the attacker can simply perform the steps describe above to determine T and S. In cases in which the attack is performed by an attacker over the Internet using a remote video camera, the attacker would need to purchase the same camera used to perform the attack in order to empirically determine T and S (unless such information appears in the video camera's specs).

IV. Analysis

In this section, we analyze the factors that affect **video-based cryptanalysis**: the bandwidth of the video camera, the target cryptographic library, the distance between the video camera and a device's power LED, and the ambient light. Throughout this section, we used two functions to create a signal from a given channel (red, green, or blue) from the video footage: $Average - Rows$ and $Average - Frames$ (see Algorithm 1). $Average - Rows$ function creates a signal (time series) from the rows of a video's frames by averaging the RGB values in each row in a frame to produce a single value for the signal. $Average - Frames$ function creates a signal (time series) from the frames of a video by averaging all RGB values in a frame to produce a single value for the signal. We note that the $Average - Rows$ function is mostly used for video footage that was obtained from a type 2 power LED, while the $Average - Frames$ function is mostly used for video footage that was obtained from a type 1 power LED.

The reason that we used a different function for each case is due to the noise added to a frame. In a single frame, noise is present in individual pixels. When the signal is weak, the noise can overpower the signal. To mitigate this, averaging all pixel values in the frame (i.e., using $Average - Frames$ function) reduces noise and improves the signal-to-noise ratio (SNR). However, when the signal is strong, averaging only the rows (i.e., using $Average - Rows$ function) can preserve fine details and capture rapid changes, providing better time resolution. The choice depends on the signal strength and the desired balance between noise reduction and temporal accuracy.

A. The Captured Bandwidth

First, we examine the bandwidth captured by various video cameras in response to changes in the intensity of a device's power LED.

Experimental Setup. We connected a USB hub to a function generator which was used to modulate a 200-25,200 Hz frequency scan using 26 sine waves at intervals of 1000 Hz

Algorithm 1 Creating Time Series from a Video

Inputs: vid = (f_1, ..., f_n) // a series of frames
chan // a value (0-2) for the RGB channel
Output: signal = a time series of rows' average values
procedure AVERAGE-ROWS(vid, chan)
return Avg-Rows-And-Frames (vid, chan, 'rows')
procedure AVERAGE-FRAMES(vid, chan)
return Avg-Rows-And-Frames (vid, chan, 'cols')
procedure AVG-ROWS-AND-FRAMES(vid, chan, fl)
signal1 = {}, index1 = 0, signal2 = {}, index2 = 0
for (frame in video) **do**
nRows = length(frame), sum2 = 0
for (r = 0; r < nRows; r++) **do**
nCols = length(frame[r]), sum1 = 0
for (c = 0; c < nCols; r++) **do**
sum1 += frame[r][c][chan]
sum2 += frame[r][c][chan]
signal1 [index1] = sum1/nCols
index1++
signal2 [index2] = sum2/(nCols * nRows)
index2++
if (fl == 'rows') **then**
return signal1
return signal2

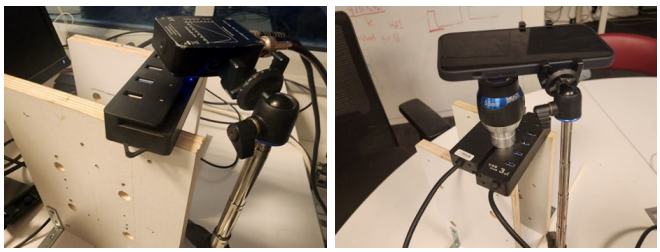

Fig. 3. Recovering a frequency scan using a photodiode (left) and a smartphone (right), by capturing video footage of the power LED of a USB hub.

(starting from 200, 1,200, 2,200, ..., 25,200 Hz), each of which was modulated at a different time. Each sine wave was modulated for 500 ms over the power supplied to the USB Hub using an amplitude of 2 V.

We conducted two experiments with two smartphones: an iPhone 13 Pro Max (resolution: 1920x1080, FPS: 120, rolling shutter speed: $\frac{1}{61400}$) and a Samsung Galaxy S22 Ultra (resolution: 1920x1080, FPS: 120, rolling shutter speed: $\frac{1}{12000}$). Both smartphones were configured to their highest rolling shutter speed. Each of the smartphones was placed on the USB hub's power LED, and their video cameras were used to film the LED, using a lens (see Fig. 3). We also conducted a third experiment and obtained an optical trace, using a photodiode (Thorlabs PDA100A2) that was connected to an NI-9223 ADC card. The photodiode was placed 2 cm away from the power LED and sampled at a sampling rate of 100 KHz. The optical trace was used for control purposes to validate the changes in the USB hub's power LED with a high-end optical sensor. The experimental setup is presented in Fig. 3. In addition, we

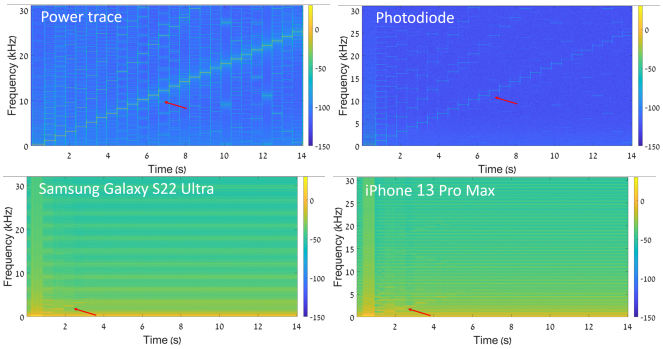

Fig. 4. A frequency scan modulated over the power supplied to a USB hub. Pictured are the spectrograms extracted from the power trace, optical trace (using a photodiode), and the blue channel of two smartphones' video cameras

obtained a power trace by inserting an adapter between the function generator and the USB. The power trace was obtained using a Digilent Analog Discovery 2 (scope).

Results. First, we extracted a spectrogram from the optical trace obtained by the photodiode (see Fig. 4). As can be seen in Fig. 4, the spectrogram shows that the frequency scan can be captured using a high-end optical sensor.

Next, we used the $Extract - Rows$ function and extracted three signals for the blue, and extracted a spectrogram for each signal (see Fig. 4). As can be seen in Fig. 4, only the first six sine waves can be seen in the spectrogram extracted from the video footage obtained by the Samsung Galaxy S22 Ultra, while only the first nine sine waves can be seen in the spectrogram extracted from the video obtained by the iPhone 13 Pro Max. These results indicate that the effective bandwidth captured by the video cameras is lower than the potential of their shutter speed. Moreover, as can be seen in the figure, the sixth sine wave that was originally produced at 5.2 kHz (and was captured by the photodiode at the same frequency) appears at around 4 kHz in the spectrogram extracted from the Samsung Galaxy and around 3.7 kHz in the spectrogram extracted from the iPhone. This is due to the video cameras' non-uniform sampling of the LED, which stems from the fact that the video cameras do not capture the LED during transitions between frames.

Next, we examined the SNR in the tested spectrum, comparing the SNR of the two smartphones' video cameras, the photodiode, and the power trace generated by the function generator (see Fig. 5. Based on the results presented in Fig. 5, we concluded that: (1) the effective bandwidth captured by the video cameras (maximum 6-10 kHz) is lower than the bandwidth captured by the photodiode (25 kHz). Within the effective bandwidth, the SNR obtained by the signal extracted from the video cameras is significantly lower than the SNR of the optical trace obtained by the photodiode, (2) the bandwidth captured by the iPhone (maximum 10 kHz) is wider than the bandwidth captured by the Samsung Galaxy S22 Ultra (maximum 6 kHz).

TABLE I
COMPARISON OF THE OPTICAL SNR OBTAINED DIRECTLY AND INDIRECTLY FROM A RASPBERRY PI RUNNING THE CRYPTOGRAPHIC LIBRARIES TARGETED BY [8, 10, 36].

	Directly	Indirectly	
	Raspberry Pi 3b+	Connected USB Hub	Connected Speakers
Libgcrypt 1.8.4	15.2 dB	16.4 dB	13.2 dB
GnuPG 1.4.13	16.5 dB	17.6 dB	14.5 dB
PQCrypto-SIDH 3.4	18.1 dB	22.4 dB	17.4 dB

B. Influence of the Cryptographic Library and Connected Peripherals

Next, we examine how the SNR is affected by the cryptographic library installed on the target device when the video is acquired directly from its power LED and indirectly from connected peripherals.

Experimental Setup. We compared the SNR obtained from the cryptographic computations performed by three cryptographic libraries installed on a Raspberry Pi 3B+: (1) Libgcrypt 1.8.4 (during an ECDSA sign operation), (2) GnuPG 1.4.13 (during an RSA decrypt operation), and (3) PQCrypto-SIDH 3.4 (during a SIKE operation).

We conducted three experiments. In the first experiment, we obtained video footage of the power LED of the Raspberry Pi 3B+ for the three cryptographic libraries. In the second experiment, we obtained video footage of the power LED of a USB hub that we connected to the Raspberry Pi 3B+ for the three cryptographic libraries. In the third experiment, we obtained video footage of the power LED of Logitech Z120 USB speakers that were connected to the USB hub that was connected to the Raspberry Pi 3B+ for the three cryptographic libraries.

Results. We applied the $Extract - Frames$ function on each video and extracted nine signals. Then, we calculated the SNR for the signals. As can be seen in the results presented in Table I, the target library under attack greatly affects the optical SNR: the SNR obtained from a SIKE operation executed by the PQCrypto-SIDH 3.4 library yields the highest SNR for the three devices (17.4-22.4 dB); the SNR obtained from an ECDSA sign operation executed by the Libgcrypt 1.8.4 library yields the lowest SNR for the three devices (13.2-15.2 dB); and the SNR obtained from an RSA decrypt operation executed by the GnuPG 1.4.13 library yields an SNR lower than that of PQCrypto-SIDH 3.4 and higher than Libgcrypt 1.8.4 (14.5-17.6 dB). This is due to the fact that: (1) The leakage from the power LED present in the power LED of the connected peripheral to the device (the USB hub) and in the connected peripherals (USB speakers) that are connected to the peripheral connected to the target device. The power LED of a connected peripheral may amplify or reduce the SNR (depending on the peripheral and the device under attack); in our case, the USB hub increases the optical SNR by ∼1.1-4.3 dB, while the USB speakers decrease the SNR by ∼0.7-2.0 dB (compared to the SNR obtained directly from the power LED of the Raspberry Pi). (4) Even devices that do not consist of an integrated power LED and devices whose power LEDs do not leak information from their integrated power LEDs may be vulnerable to *video-based cryptanalysis* when they are connected to another peripheral with a vulnerable power LED.

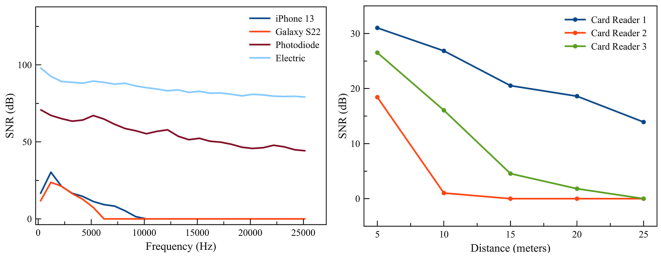

Fig. 5. Left: Right: SNR of smart card readers obtained from various distances.

C. Influence of Distance

Next, we examine how the SNR is affected by the distance between the target device's power LED and the video camera. This experiment was conducted using three smart card readers that we bought from Amazon (to ensure confidentiality as part of our process of responsible disclosure, we have not mentioned the specific models used in this version of the paper; we will include this information in the next version). Each of the smart card readers contains an indicative power LED (type 2; described earlier in the paper) which makes it easier to detect the changes in their color from a distance (and determine the beginning/end of an operation). We note that the differences in the RGB values of standard on/off power LEDs (type 1) can be detected from a maximum range of one meter, and they are not vulnerable to video-based cryptanalysis performed from a distance (i.e., using remote data acquisition).

Experimental Setup. We conducted three experiments. In each experiment, we connected a smart card reader to a laptop and inserted the Athena IDProtect smart card into the reader. We wrote a script that triggers an ECDSA sign operation using the smart card every 200 milliseconds. In response to an ECDSA sign operation, color of the power LED of the smart card readers changes. Then, we placed the SUNBA video camera at five different distances from the smart card reader (5, 10, 15, 20, 25 meters) and filmed the power LED of the smart card reader in two states: idle and sign.

Results. We applied the $Extract - Frames$ function and extracted the associated signals for the blue channel. We compared the SNR, using the values of the idle episode and the RGB values of the ECDSA sign operation (see Fig. 5). Based on these results, we concluded that the beginning/end of ECDSA sign operations can be detected up to a range of 25 meters in one specific case.

D. Influence of Ambient Light

Next, we examine how the optical SNR is affected by ambient light for two types of optical data acquisition: (1) close data acquisition, in which the video camera of a smartphone placed on the device is used to film the power LED, and (2) remote data acquisition, in which a remote video camera is used to film the power LED.

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

	Ambient Light		
	Darkness	Room Lighting (Fluorescent)	Sunlight
Data Acquisition	0 Lux	300 Lux	3000 Lux
Close via a smartphone (2 cm)	26.8 dB	14.6 dB	0 dB
Remote via a security camera (10 meters)	16.9 dB	17.2 dB	16.6 dB

Experimental Setup. We conducted two experiments. In the first experiment, we connected a USB hub (Gold Touch 4 Ports USB3.0 Slim HUB) to a Samsung Galaxy S8. We wrote a program that alternates between one-second repetitions of integer multiplications (MUL) and one-second sleep operations (WFI). We executed the code on the Samsung Galaxy S8. We placed an iPhone 13 Pro Max on the USB hub and filmed its power LED in three environmental settings: a dark room (0 lux), a room lit with fluorescent lighting (300 lux fluorescent tubes), and a sunlit room (3000 lux). In the second experiment, we inserted the Athena IDProtect smart card into a smart card reader that was connected to a laptop. We wrote code that triggers an ECDSA sign operation every 200 ms. We placed the SUNBA video camera 10 meters away from the smart card reader and filmed its power LED in the same environmental settings: 0, 300, and 3000 lux.

Results. We analyzed the six videos and calculated the SNR which are presented in Table II. Based on these results, we concluded that: (1) in close data acquisition (when the smartphone's video camera is placed directly on the power LED), the ambient light does not affect the SNR (in this case, there is a change of up to 0.6 dB in the SNR, which is a reasonable sampling error), and (2) in remote data acquisition (where the video camera is placed at a distance from the power LED of the device), the ambient light highly affects the SNR (there is a change of up to 26.8 dB in the SNR), and dark environments yield higher SNR values.

V. RECOVERING ECDSA KEYS

In this section, we describe the Minerva attack [8] that we performed to demonstrate the recovery of a 256-bit ECDSA private key from a smart card, using video footage obtained by an Internet-connected security camera directed at the power LED of a smart card reader from a range of five meters.

Minerva Attack. As observed in the papers presenting the Minerva [8] and TPM-FAIL [9] attacks, many common cryptographic libraries optimize the computation time of ECDSA signing by truncating any leading zeros. This optimization results in a variable number of loop iterations that is associated with a variable execution time for the entire main loop, which is determined by the number of leading zeros in the randomly generated nonce. Thus, by measuring the signing time, attackers can detect the number of loop iterations and determine the number of leading zeros in the nonce k, which can be used to extract the target's private key using lattice techniques, in which 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 [8] for details).

We performed the Minerva attack to recover a smart card's ECDSA private key by estimating the signing time of a signature from video footage of the smart card's power LED (as opposed to the original applications [8, 9] of the attack which relied on CPU measurements of the ECDSA signing operations obtained using code installed on the target laptop/computer). We also demonstrate a remote attack in which footage obtained by an Internet-connected video camera located five meters away from the smart card reader is exploited.

A. Identifying ECDSA Operations from Various Smart Card Readers

First, we show that eight commercial smart card readers with an indicative power LED (type 2) that we bought on Amazon leak information regarding the execution time of the ECDSA signature from their power LED. The color of the power LED of these smart card readers changes in response to an operation triggered by the connected laptop. We did not name the manufacturers and models in the paper to allow for responsible disclosure.

1) Experimental Setup: We conducted five experiments, and in each experiment one of the smart card readers was connected to a laptop via a USB cable and the Athena IDProtect smart card was inserted into the reader. We placed a SUNBA video camera 25X optical zoom 5MP smart security dome[1] 20 cm away from the smart card reader. The Internet-connected video camera's remote API was used, and the camera's point of view was directed at the smart card reader's power LED. We focused the video camera on the power LED and zoomed in until it filled the entire frame. Video footage (full HD resolution, 60 FPS, shutter speed $\frac{1}{500}$, 8-bit resolution for a single channel) was obtained from six consecutive different ECDSA sign operations performed by the smart card (separated by 200 ms of sleep).

2) Results: The five videos obtained in the experiment described above were processed. For each video, we applied Algorithm 1, which creates a signal from the rows of a video's frames by averaging the RGB values in each row in a frame to a single value in the signal. Fig. 6 presents the signals (average RGB values) of the 127,440 rows of 118 consecutive frames from the five videos during six different ECDSA operations, separated by 200 ms sleep episodes. Based on this experiment, we concluded that: (1) the six ECDSA signatures can be seen in the five extracted signals, and (2) the thresholds that differentiate the signing and sleep episodes vary depending on the signal extracted from the smart card readers and the RGB channels.

B. Recovering ECDSA Keys from 16 Meters Away

We now demonstrate the end-to-end recovery of a 256-bit ECDSA private key from video footage.

[1] https://www.amazon.com/SUNBA-Ceiling-Outdoor-Security-Infrared/dp/B09Z6R48SH/r

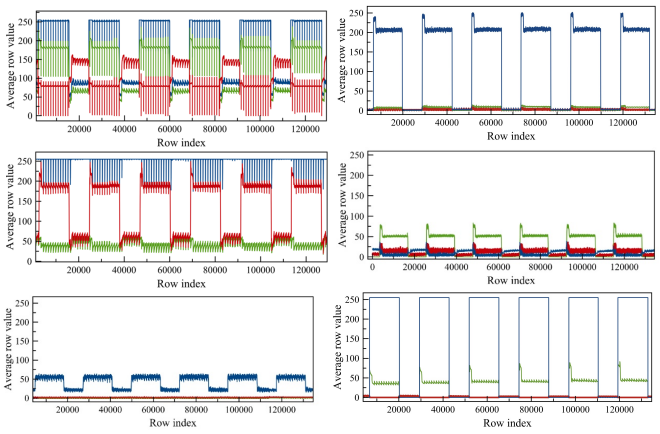

Fig. 6. The RGB values extracted from 96 consecutive frames (with 103,680 rows) from video footage of the smart card reader's power LED during the execution of six different ECDSA signing operations, separated by 200 ms sleep episodes. The colors represent the associated RGB channel.

Fig. 7. Experimental setup. As seen on the left, the video camera was directed at the smart card reader (indicated by the red arrow) from 16 meters away. On the right is an image of the smart card reader's power LED.

1) Experimental Setup: We connected one of the smart card readers to a laptop via a USB cable and inserted the Athena IDProtect smart card into the reader. We placed a SUNBA video camera 25X optical zoom 5MP smart security dome[1] 16 meters away from the smart card reader (the experimental setup can be seen in Fig. 7). We used the Internet-connected video camera remote API and directed the camera's point of view toward the power LED of the smart card reader and zoomed in on the power LED until it filled the entire frame (see Fig. 7). The experiment was conducted in a dark environment (i.e., the lights in the room were turned off). We obtained video footage (full HD resolution, 60 FPS, shutter speed $\frac{1}{500}$, 8-bit resolution for a single channel) from 10,500 different ECDSA sign operations performed by the smart card (separated by 200 ms of sleep). The 10,500 signatures were recorded in 65 minutes over 35 different videos. Each video lasted 1:50 minutes and consisted of 300 triggered ECDSA signatures.

2) Extracting Frame Series Associated with ECDSA Signatures: We applied the function percentage (see Algorithm 2) on the video frames $(f_0, ..., f_n)$ in order to determine whether a frame is associated with a state in which the smart card was in an idle mode (i.e., the frame consists only of blue rows) or used for an ECDSA signing operation (i.e., the frame consists of at least one red row). The function percentage implements the abovementioned requirement by receiving a frame f_i and applying Algorithm 1 to extract a signal s_i, where each value in the signal is the average of a row of f_i in the blue channel. The function returns p_i, which is the associated percentage of the values (averages of rows) in s_i that are below the threshold distinguishing between the two states of the smart card reader: idle (> 37.5) and operation (< 37.5). The threshold separating the two states was determined based on the experiment described in Section III-C (see Fig. 6).

For each frame f_i, the associated value p_i was used to compute b_i, a binary value {idle/sign} that determines whether the associated frame f_i is associated with a state in which the smart card reader was in an idle state (i.e., 0% of the rows are black/red, and 100% of the rows are blue) or was used for signing (i.e., some of the rows are black/red and some of the rows are blue) accordingly:

$$b_i = \begin{cases} idle, & \text{if } p_i = 0 \\ sign, & \text{otherwise} \end{cases} \quad (1)$$

We note that at the end of this process, the values of $b_0, ..., b_n$ consisted of 10,500 consecutive series of $sign$ separated by 10,501 consecutive series of $idle$ as follows: $idle, idle, ..., idle, sign, sign, sign, ..., sign, idle, idle, ..., idle$. We identified the 10,500 consecutive $sign$ series (that were separated by the $idle$ series), extracted their associated frames $(f^1_{start}, ..., f^1_{end}, ..., f^{10,500}_{start}, ..., f^{10,500}_{end})$, and mapped them to their relevant signatures $(ECDSA_1, ..., ECDSA_{10,500})$.

For each $ECDSA_i$ signature $(1 \leq i \leq 10,500)$, the frame series associated with it $(f^i_{start}, ..., f^i_{end})$ was analyzed, and the signature was classified as one of two classes: **Class I**: a frame series of ECDSA signatures that started and ended during the scanning time; such signatures can be identified based on the switch between the blue and black colors somewhere in the middle of the first (f^i_{start}) and last (f^i_{end}) frames, as can be seen in Fig. 8; and **Class II**: a frame series of ECDSA signatures that started or ended during the transition time (which is not captured by a frame); such signatures can be identified based on the full black color in the first (f^i_{start}) or last (f^i_{end}) frame, as can be seen in Fig. 8.

The class of each signature $ECDSA_i$ with associated frame series $f^i_{start}, ..., f^i_{end}$ was determined by examining whether the blue rows appear in the first (f^i_{start}) or last (f^i_{end}) frame. This was done by applying the percentage function (see Algorithm 2), calculating $p^{f^i_{start}} = Percentage(f^i_{start})$ and $p^{f^i_{end}} = Percentage(f^i_{end})$, and testing their values using the following conditions:

$$class(ECDSA_i) = \begin{cases} \text{class II}, & \text{if } p^{f^i_{start}} = 1.0 \\ \text{class II}, & \text{if } p^{f^i_{end}} = 1.0 \\ \text{class I}, & \text{otherwise} \end{cases} \quad (2)$$

We note that we were unable to compute the ECDSA signature time of frame series that started or ended during the transition between frames (i.e., Class II signatures) without adding an error, since the beginning/end of the Class II signature occurs during the transition between the frames and

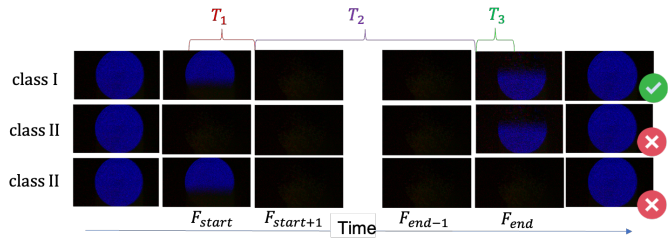

Fig. 8. Examples of ECDSA series extracted from video footage of a smart card reader's power LED. Top: A series of frames that started and ended during the rolling shutter's scanning time (a Class I series). Middle: A series of frames that started during the transition time between frames (a Class II series). Bottom: A series of frames that ended during the transition time between frames (a Class II series).

is not captured in any video frames. Since the process of performing lattice reduction followed by the time extraction is highly sensitive to errors, we filtered the 2,674 Class II signatures from the data.

3) Estimating the ECDSA Signature Time from Frame Series: First, we empirically computed the video camera's transition (T) and scanning (S) times by performing the experiment described in Section III-C. Based on that experiment, we determined that $S = 13.8ms$ and $T = 2.8ms$.

Then, we computed the execution time of the frame series of the remaining 7,826 Class I signatures. We assume a video camera with the following specifications: for every *Class I* $ECDSA_i$ signature with associated frame series $video_i = f^i_{start}, ..., f^i_{end}$ that started at index $start$ and ended at index end. The signing time was calculated by applying the CalculateSiginingTime function (Algorithm 2) accordingly: CalculateSiginingTime($video = video_i$, $scan = 13.8$, $trans = 2.8$, $threshold = 37.5$, $channel = 2$).

Algorithm 2 calculates the signing time of an ECDSA signature $ECDSA_i$ as the sum of $T_1 + T_2 + T_3$ (see Fig. 8). Algorithm 2 calculates T_1 by multiplying the relative number of rows that are associated with the sign operation in the first frame f^i_{start} by the scanning time S and adding the transition time T. Algorithm 2 calculates T_3, by multiplying the relative number of rows associated with the sign operation in the last frame f^i_{end} by the scanning time S (in this case we do not add the transition time T, because the sign operation ends in the middle of the last frame's f^i_{end} scanning time). Algorithm 2 calculates T_2 for the additional $end - start - 1$ frames ($f^i_{start+1}, f^i_{start+2}, ..., f^i_{end-2}, f^i_{end-1}$) by multiplying the number of frames by the sum of $S + T$. The algorithm returns the sum of $T_1 + T_2 + T_3$ as the signing time for $ECDSA_i$.

4) Results: We computed the ECDSA signing time for 7,826 *Class I* signatures by applying Algorithm 2 on the 7,826 associated videos. The heat map presented in Fig. 9 shows that the signatures with the shortest estimated time (that was calculated from the video footage) have nonces with many leading zeros, as needed for the Minerva attack. We then executed the Minerva cryptanalysis script (downloaded from the official Minerva GitHub repository [37]) and recovered the full 256-bit ECDSA key in two minutes.

Algorithm 2 Minerva Attack

Inputs. video: (f_{start}, ..., f_{end}) // a series of frames
ch: numeric value {0,1,2} for the RGB channel
scan: numeric a rolling shutter scanning time (ms)
tran: a rolling shutter transition time (ms)
thresh: a cuttoff to distinguish the idle/sign states
Output. signal: a time series of rows' average values
procedure SIGNINGTIME(video, ch, scan, tran, thresh)
$T_1 = Percentage(f_{start}) \times$ scan + trans
$T_2 = (end - start + 1) \times$ (scan + trans)
$T_3 = Percentage(f_{end}) \times$ scan
return $T_1 + T_2 + T_3$
procedure PERCENTAGE($frame, ch$)
signal = Average-Rows(frame, ch)
sum = 0
for (i = 0; i < length(signal); i++) **do**
if (signal[i] < threshold) **then**
sum++
return sum/length(signal)

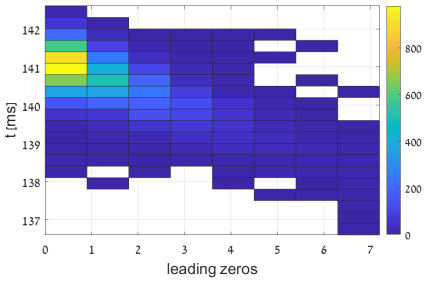

Fig. 9. A heat map of the estimated execution times of 7,826 ECDSA sign operations as a function of the number of leading zero bits in the nonce.

For completeness, we note that we also recovered the ECDSA key from the power LED of the additional five smart card readers using the same video camera from a distance of 30 cm. The five associated heatmaps extracted from the power LED of the five smart card readers can be seen in Fig. **??**.

VI. RECOVERING SIKE KEYS

In this section, we describe the recovery of a secret key from supersingular isogeny key encapsulation (SIKE), a post-quantum key encapsulation mechanism based on the supersingular isogeny Diffie-Hellman (SIDH) key exchange protocol [38]. We perform the Hertzbleed attack [10] against a Samsung Galaxy S8 and recover a secret key (378-bit) from the implementation of SIKE-751 in the PQCrypto-SIDH library by using the video camera of an iPhone 13 Pro Max to obtain video footage of the power LED of USB speakers that were connected to the USB hub used to charge the Samsung Galaxy S8 (which stored the SIKE key).

Hertzbleed Attack. As seen in the Hertzbleed attack [10], the SIKE implementation in the PQCrypto-SIDH library leaks information regarding the bits of the key due to Intel's DVFS (dynamic voltage and frequency scaling) mechanism, which in certain circumstances can be exploited by an attacker to induce variations in the CPU frequency by overloading the CPU

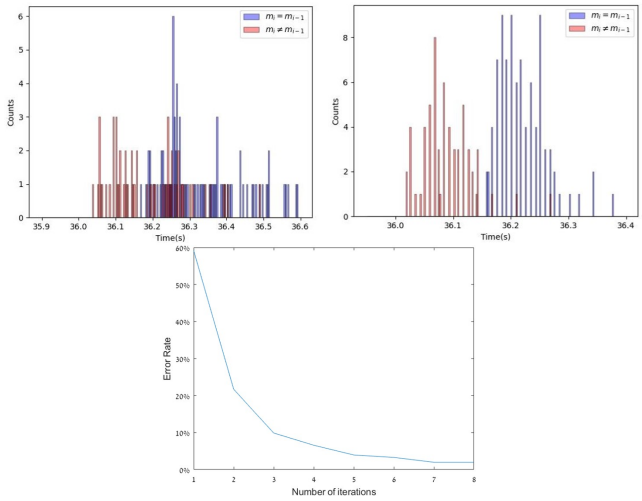

Fig. 10. Top: A histogram of video-footage-based estimation of the minimal runtime based on one iteration (left) and eight iterations (right). The error rate (when a threshold of 36.15 is used to distinguish between $m_i = m_{i-1}$) and $m_i \neq m_{i-1}$) vs. the number of iterations used to determine the minimal runtime.

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 [10] for details).

We apply the Hertzbleed attack to recover a Galaxy S8's SIKE private key by estimating the running time of various SIKE decapsulation operations, using video footage, obtained by an iPhone 13 Pro Max, of the power LED of a connected USB hub which is used to charge the smartphone. By doing so, we show that the Samsung Galaxy S8, which is based on the ARM architecture, is also vulnerable to the Hertzbleed attack (as opposed to the original attack which targeted an x86 architecture).

The Hertzbleed key extraction attack targets the smartphones's 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 least significant bit (LSB) of m), an attacker can construct points P and Q so 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 LSB of m when constructing the ciphertext, then the $(i + 1)$st 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 (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 private key's bit under attack and then determining whether $m_i = m_{i-1}$ or not, based on a timing threshold.

A. Determining the Threshold

First, we examine whether the behavior (the time difference) on the x86 architecture reported in the original paper on Hertzbleed [10] is also seen on the ARM architecture of the Samsung Galaxy S8.

1) Experimental Setup: We downloaded the code published in the Hertzbleed repository [39], 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 four different SIKE-751 keys. For each key $m = (m_{l-1}, ..., m_0)_2$, we uniformly targeted 38 bit positions: 5, 15, 35, 45,...,375. For each of the bit positions, we executed a series of 800 SIKE operations divided into eight 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); eight iterations were used to measure the execution time of each bit. In this case, the execution time of the SIKE iterations was calculated using CPU measurements obtained with the code downloaded from the official Hertzbleed repository.

2) Results: For each bit, we only used the last seven iterations (which consisted of 700 SIKE decapsulation operations) and disregarded the first iteration (which consists of 100 SIKE decapsulation operations), since we found that the first iteration was unstable and 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, 12.5% of the measurements were filtered out (1,064 iterations were used).

Fig. 10 presents the distribution of the execution times of the 1,064 iterations, calculated from the CPU measurements. As can be seen, the distribution is very noisy, and there is no clear threshold that can be used to differentiate the cases. The execution times in red represent the case of a switch ($m_i \neq m_{i-1}$), with a mean = 36.354 and STD = 0.7478, and the execution times in blue represent the case of a non-switch ($m_i = m_{i-1}$), with a mean = 36.527 and STD = 0.8211.

As a result, for each bit and its seven iterations, we decided to compute the minimal execution time of the iterations. The distribution of the 152 bits (based on the minimal execution time for the associated iterations) is presented in Fig. 10. The execution times in red represent the case of a switch ($m_i \neq m_{i-1}$), with a mean = 36.092 and STD = 0.073, and the execution times in blue represent the case of a non-switch ($m_i = m_{i-1}$), with a mean = 36.223 and STD = 0.084. A threshold of 36.15 can be used to differentiate between the two classes with a negligible error. We also computed the error as a function of the number of iterations (1-8) used to calculate the minimal execution time with a threshold of 36.15 s. (the

Fig. 11. Experimental setup. The video camera of an iPhone 13 Pro Max is directed (through a lens) at the power LED of Logitech Z120 speakers that are connected to a USB hub used to charge a Samsung Galaxy S8 (which contains the SIKE key).

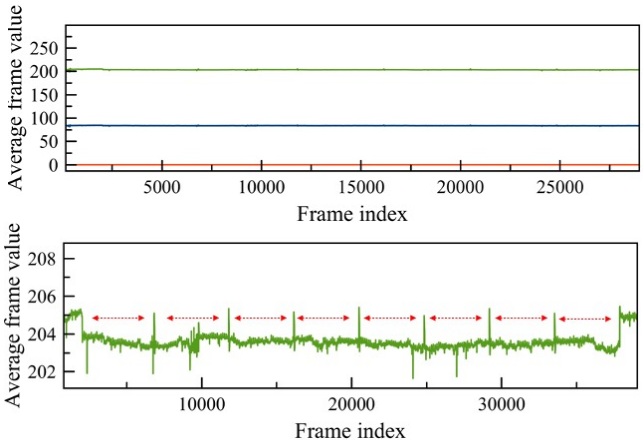

Fig. 12. The RGB values of eight SIKE iterations extracted from a video (top). Zooming in on the green channel (bottom).

results are presented in Fig. 10). As can be seen from the results, the error converges to 1% in the seventh iteration.

Based on this experiment, we concluded that: (1) the behavior (the time difference) reported in the Hertzbleed paper [10] 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 36.15 seconds that differentiates the switch cases from the non-switch cases, and (2) there is a need to employ an error correction algorithm to handle the expected 1% of errors.

Algorithm 3 Hertzbleed Attack

```
Inputs. vid: (f_start, ..., f_end) // a series of frames
        chan: numeric value {0,1,2} for the RGB channel
Output. signal: a time series of rows' average values
procedure EXTRACT-INDEXES (SIGNAL, THRESH)
    indexes = [], i = 0
    signal = Average-Frames (video,1)
    for (j = 0; j < length(signal); j++) do
        if ( thensignal [j] > threshold)
            indexes [i] = signal [j]
            i++
    return indexes
procedure MINITERTIME (VID, CHAN, THRESH)
    estimatedTimes = {} , i =0
    signal = Average-Frames (vid, chan)
    indexes = Extract-Indexes (signal, threshold)
    for (j1 = 0; j1 < length(indexes)-1; j1++) do
        for (j2 = j1+1; j2 < length(indexes)-1; j2++) do
            n = indexes [j2] - indexes [j1])
            time = n × (1/fps)
            if (time > 36) then
                estimatedTimes [i] = time
                i++
    min = minimum (estimatedTimes)
    return min
```

B. SIKE Key Recovery

We now demonstrate the recovery of a full (378-bit) private key from the SIKE-751 implementation using video footage, obtained by an iPhone 13 Pro Max, from the power LED of USB speakers that were connected to a USB hub that was used to charge a Samsung Galaxy S8 (which contains the SIKE key) in a series of adaptively chosen ciphertext attacks.

1) Experimental Setup: We connected the Samsung Galaxy S8 to a USB hub (Gold Touch 8 Ports USB3.0 Slim HUB). We also connected USB speakers (Logitech Z120) to the USB hub. We used the video camera of an iPhone 13 Pro Max and zoomed it onto the power LED of the USB speakers using a lens (see Fig. 11). We obtained video footage from the iPhone (resolution: 1920x1080, FPS: 120, rolling shutter speed: $\frac{1}{61400}$) of the power LED of the USB speakers while the Samsung Galaxy was attacked in a series of adaptive chosen ciphertext attacks.

The series of adaptively chosen ciphertexts was created as follows: For each index i of the private key we wanted to recover, we created a dedicated input M_i which was used to attack the implementation of SIKE-751 in the PQCrypto-SIDH library, as described in the paper presenting Hertzbleed [10] (using the $i-1$ bits already recovered). We used M_i to trigger 800 SIKE operations, which were divided into eight 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}.

2) Processing the Signal: We processed each video obtained as we triggered the adaptive chosenly ciphertext attack as follows:

We applied the $MinIterTime$ function (see Algorithm 3) on the video footage obtained. This function calls to $Average-Frames$ to extract a signal for the green channel of the video footage. An example of the signal extracted from one of the videos is presented in Fig. 12; as can be seen in the image presenting the green channel, the eight iterations can be detected visually, and the indexes associated with the beginning of the iterations can be detected based on their values which are greater than a threshold of 205.

Next, $MinIterTime$ calls to $Extarct-Indexes$ to extract the indexes of the frames associated with the beginning of the iterations. This is done by determining whether the values of

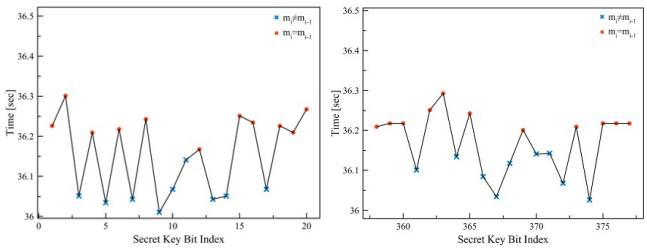

Fig. 13. Minimum times used to extract the first 20 bits (1 to 20) and last 20 bits (358 to 377) of the SIKE key based on eight iterations.

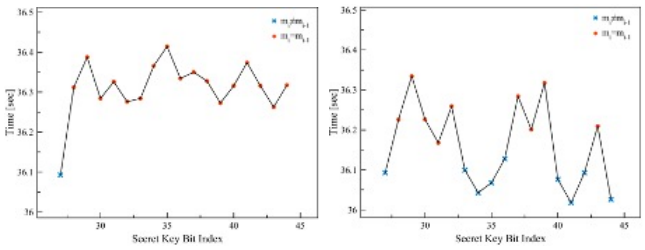

Fig. 14. The error detection (left) and correction (right) of bit index 33. The graphs for the additional five indexes that were corrected appear in Fig. **??**

the indexes of the frames are greater than 204.9. We note that due to added noise, the $Extarct - Indexes$ function may return more than eight indexes (i.e., the function may return additional errors). Next, $MinIterTime$ computes the number of frames between every two indexes (due to the errors that may have been added in indexes) and calculates the time of the iterations by multiplying the number of frames by (1/fps), the number of seconds it takes to capture a frame (including the transition time). The function filters any result that is lower than 36 seconds (and caused by the added errors). Finally, $MinIterTime$ returns min, the shortest iteration time.

We determined the value of the i-th index of the key according to the value of min:

$$m_i = \begin{cases} m_{i-1} & \text{if } (min > 36.15) \\ \neg(m_{i-1}) & \text{otherwise} \end{cases} \quad (3)$$

on the $m_i \neq m_{i-1}$.

3) 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 Hertzbleed, 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, 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. The minimal values among the seven iterations of the first 20 least significant bit (LSB) positions (bits 1–20) and the last 20 most significant bit (MSB) positions (bits 358–377) of the key we recovered are presented in Fig. 13. The 378 bits of the key were recovered with six errors that we encountered and corrected during the recovery process.

4) Error Detection and Correction: We used an error detection and correction algorithm to detect and correct the six errors we encountered during the recovery process, based on the error correction and detection algorithm suggested in the original method suggested in the Hertzbleed paper [10]. In the case of an error in the recovery of a bit with an index i, the phenomenon that causes anomalous zeros (which is expected to happen with a probability of $\frac{1}{2}$) will not be triggered in the subsequent bits recovered (see the Hertzbleed paper [10] for more details). The untriggered anomalous zeros in the recovery of the subsequent indexes will result in a non-switch case and a longer execution time (that will cross the threshold used to distinguish between $m_i = m_{i-1}$ and $m_i \neq m_{i-1}$). This will result in a chain of recovered bits with similar values (the result of a chain of non-switches) for the subsequent indexes.

In order to detect such errors, we set the detection algorithm to raise an alert after 17 consecutive recoveries classified as non-switch cases (whose execution time crossed the threshold of 36.15 s). A chain of 17 consecutive non-switch bits is expected to be the result of an error in a recovered bit with 99.9992% (except for a negligible error with a probability of $\frac{1}{131,072}$ which is the result of 17 consecutive bits with the same value in the key). Figs. 14 and 17 (in the appendix) present six chains of 17 consecutive non-switch bits that we encountered during the key recovery. For these chains, we repeated the sampling process from the last index in which we encountered a switch case. The corrected classifications for the chains of the six bits are also presented in Figs. 14 and 17.

VII. Countermeasures

In this section, we describe several methods that can be used to mitigate or prevent optical cryptanalysis attacks.

We note that the best way to prevent attackers from recovering secret keys from devices is to ensure the cryptographic library used does not leak any information that can be exploited to recover the key. However, hardware manufacturers and users can apply their own mitigations in order to prevent attackers from applying video-based cryptanalysis.

Manufacturer Side Methods. We differentiate between two types of power LEDs: type 1 power LEDs (standard on/off power LEDs) and type 2 power LEDs (that provide an indication regarding CPU operations by changing their color). We would advise manufacturers against integrating type 2 power LEDs in devices, because, as we have shown in this work, they expose the device to remote secret key recovery via an Internet-connected video camera; such LEDS enable attackers to calculate the execution time of an operation by analyzing the video footage from a distance (16 meters) due to the significant changes in the color of the power LED in response to CPU operations. In many devices, the type 1 power LED is connected directly to the power line of the PCB (see Fig. 15a). 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. 15b). This is an inexpensive solution for reducing the fluctuations in the power consumption. However in devices with high power consumption, the integrated capacitor's capacitance must be

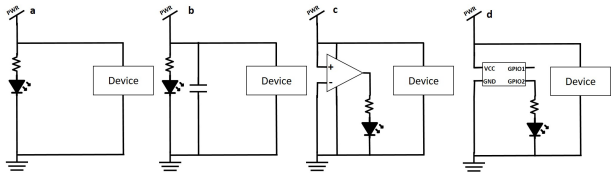

Fig. 15. A circuit that leaks information via its power LED (a). Countermeasures using a capacitor (b), an additional OPAMP (c), and an existing OPAMP (d).

large enough to support the power supply to the device. (2) Using an operational amplifier (OPAMP); this can be implemented by integrating an OPAMP between the power line and the power LED (see Fig. 15c) or by using an existing GPIO port of an integrated microcontroller as a power supply for the power LED (see Fig. 15d). 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.

VIII. Limitations

In this section, we discuss the limitations of video-based cryptanalysis and how attackers can overcome them.

Limited Sampling Rate. Currently, the fastest shutter speed of a commonly used commercial video camera supports a speed of $\frac{1}{60{,}000}$ (iPhone 14 Pro Max) which allows a sampling rate of 60K measurements per second. As a result, devices with high CPU rates (e.g., servers) may not be at risk of video-based cryptanalysis, even if their power LED leaks fine-grained information that could have been used for cryptanalysis. We note that attackers can overcome this limitation by: (1) combining video footage obtained from a set of video cameras triggered by a C&C application that launches the video footage of each video camera at different offsets in order to increase the sampling rate in the factor of the number of video cameras, or (2) by using dedicated video cameras that support a shutter speed of (e.g., Fujifilm X-H2 supports a shutter speed of $\frac{1}{180{,}000}$).

Semi-Uniform Sampling. We note that for a video camera with an FPS rate, the time given by $\frac{1}{FPS} = S + T$ consists of S, which denotes the top-bottom scanning time of a single frame, and T, which denotes the transition time between frames (see Fig. 1 for more details). As a result, while the rolling shutter provides uniform sampling of the intensity of the power LED within a frame, the video of the power LED does not provide a uniform distribution of the intensity of the power LED across time (due to the fact that the power LED is not captured by a frame during the transition time). As a result, in the video footage there is no indication for a cryptographic operation that starts/ends during the transition time. Attackers can use one of the following two approaches to resolve this: (1) Estimate the time of the missing beginning/end indication of an operation by adding half of the transition time ($\frac{T}{2}$). The main disadvantage of this simple approach is the fact that some cryptanalytic attacks cannot tolerate errors. (2) Calculate the accurate beginning/end time by collecting additional measurements. In general, if we denote the transition time as T and the scanning time as S, where $S = \frac{1}{FPS} - T$, the probability that the beginning and the end of a cryptographic operation will be captured in the video is: $S^2 \times FPS^2$. Based on this observation, attackers can obtain a few video recordings of the LED (while triggering the same cryptographic operation) and increase the probability that the video footage will consist of at least one cryptographic operation that started and ended during the scanning time. By doing so, attackers can estimate the accurate beginning/end time from samples for which the starting and ending indication was captured in the video footage.

Low/No Indication from the Integrated Power LEDs. We note that some devices do not leak fine-grained information from their integrated power LEDs. In such devices, the device manufacturers decoupled the correlation between the power consumption of the device and the intensity of the integrated power LED in the design of the electrical circuits. However, assuming that the power consumption of the device actually leaks fine-grained information which can be used for cryptanalysis, attackers can overcome this challenge by performing an indirect attack (i.e., exploiting the leakage from the power LED of a connected device), as we demonstrated in Section VI. In devices that leak fine-grained information from their power LEDs, we note that we observed different SNRs and different/changing behaviors of signal extracted from the video footage obtained by different video cameras.

Limited Sampling Sensitivity. We note that video cameras are less sensitive than photodiodes, which can capture much more subtle changes in the brightness of the LED. As a result, a photodiode provides greater sensitivity and can yield a higher SNR than a video camera (as can be seen in Fig. 5).

IX. Discussion & Disclosure

In this paper we showed that power LEDs can be exploited by attackers as an infrastructure to obtain timing measurements from devices and apply cryptanalytic side-channel attacks against vulnerable cryptographic libraries. One might question the contribution of *video-based cryptanalysis*, arguing that this method is really used only to facilitate timing-based cryptanalytic side-channel attacks, and that as a result, the video footage is not needed, since the API used to trigger the cryptographic operation during the attack on the target device can be used to obtained time measurements by calculating the time between the API request and the API response. In response to such an argument, we note that in many cases, the latency of networks and the Internet prevents attackers from performing timing-based cryptanalytic side-channel attacks remotely, because the timing measurements are compromised by the latency of the network. For example, in the FAQ section of the GitHub repository published by the authors of the Minerva attack [8], the authors mentioned that they were unable to perform the attack remotely for this reason and were only able to perform the attack using timing measurements obtained using code that was installed on the target device and

used to obtain timing measurements by directly probing the CPU (see *Is this exploitable remotely?* in Minerva's GitHub [37]).

We also raise concern regarding the real potential of *video-based cryptanalysis* in our days, given existing improvements in video cameras' specifications. In our research, we focused on commonly used and popular video cameras to demonstrate *video-based cryptanalysis* (i.e., 8-bit space for a single RGB channel, Full-HD resolution, and maximum supported shutter speed). However, new versions of smartphones already support video footage of 10-bit resolution (e.g., iPhone 14 Pro MAX and Samsung Galaxy S23 Ultra). Moreover, professional video cameras with a resolution of 12-14 bits already exist, [2] Such video cameras may provide much greater sensitivity, which may allow attackers to perform attacks with the ability to detect very subtle changes in the device's power consumption via the intensity of the power LED. In addition, many Internet-connected security cameras with greater optical-zoom capabilities than the video camera used in our research (25X) already exist (30X, 36X) and are likely already widely deployed. Such security cameras may allow attackers to perform *video-based cryptanalysis* against target devices from a greater distance than that demonstrated in this paper. Finally, new professional video cameras for photographers currently support a shutter speed of $\frac{1}{180{,}000}$ (e.g, Fujifilm X-H2.[3]) The use of such video cameras may allow attackers to obtain measurements at a higher sampling rate which may expose other devices to the risk of *video-based cryptanalysis*.

The existing advancements in video cameras (that were stressed earlier) already raise a question regarding the real potential of *video-based cryptanalysis* in our days with the use of professional video cameras (instead of the popular and available video cameras that we used in this research). Factoring the expected advancements for video cameras in the next 20 years (assuming the improvements of video cameras will continue to follow Moore's Law) and the fact that many functional IoT devices with limited CPU capabilities (e.g., sensors, and home appliances) are deployed every day, we expect that the number of devices exposed to video-based cryptanalysis will increase every year (unless dedicated precautions will be added to the electrical circuits).

We disclosed our findings to the manufacturers of the devices used in our study via their bug bounty programs and contact us email addresses (besides one manufacturer that we were unable to find any information about it on the web). We encouraged the manufacturers to speak to us to ensure that they understood the problem and assist them in developing a countermeasure. A few manufacturers responded to our email and asked us for more details which we shared with them. While the origin of the vulnerability that is exploited is the result of the implementation or execution of the cryptographic library and not of the hardware manufacturer, we recommend that other hardware manufacturers empirically test whether their devices are vulnerable to *video-based cryptanalysis* and if needed, redesign their electrical circuits (according to the suggestions provided in Section VII). 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 *video-based cryptanalysis* attacks in the future.

For future work, we suggest: (1) testing the potential of professional video cameras to recover cryptographic keys, (2) testing the effectiveness of the suggested countermeasures against the application of *video-based cryptanalysis*, and (3) extending the understanding in privacy risks posed by power LEDs to information confidentiality in the digital domain.

[2]https://www.globalspec.com/ds/39/areaspec/bits_per_pixel_14

[3]https://fujifilm-x.com/en-us/products/cameras/x-h2/

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, and the Technion's Viterbi Fellowship for Nurturing Future Faculty Members.

REFERENCES

[1] B. Nassi, R. Bitton, R. Masuoka, A. Shabtai, and Y. Elovici, "Sok: Security and privacy in the age of commercial drones," in *2021 2021 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, may 2021, pp. 73–90. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SP40001.2021.00005

[2] C. Bloom, J. Tan, J. Ramjohn, and L. Bauer, "Self-driving cars and data collection: Privacy perceptions of networked autonomous vehicles," in *Symposium on Usable Privacy and Security (SOUPS)*, 2017.

[3] A. Schwartz, "Chicago's video surveillance cameras: a pervasive and poorly regulated threat to our privacy," *Northwestern Journal of Technology and Intellectual Property*, vol. 11, no. 2, p. 47, 2013.

[4] N. H. Tan, R. Y. Wong, A. Desjardins, S. A. Munson, and J. Pierce, "Monitoring pets, deterring intruders, and casually spying on neighbors: Everyday uses of smart home cameras," in *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, 2022, pp. 1–25.

[5] A. Davis, M. Rubinstein, N. Wadhwa, G. J. Mysore, F. Durand, and W. T. Freeman, "The visual microphone: passive recovery of sound from video," 2014.

[6] M. Sheinin, D. Chan, M. O'Toole, and S. G. Narasimhan, "Dual-shutter optical vibration sensing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 16 324–16 333.

[7] Y. Long, P. Naghavi, B. Kojusner, K. Butler, S. Rampazzi, and K. Fu, "Side eye: Characterizing the limits of pov acoustic eavesdropping from smartphone cameras

with rolling shutters and movable lenses," *arXiv preprint arXiv:2301.10056*, 2023.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

[13] J.-J. Quisquater and D. Samyde, "Electromagnetic analysis (ema): Measures and counter-measures for smart cards," in *International Conference on Research in Smart Cards*. Springer, 2001, pp. 200–210.

[14] 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.

[15] 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.

[16] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic analysis: Concrete results," in *International workshop on cryptographic hardware and embedded systems*. Springer, 2001, pp. 251–261.

[17] 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.

[18] 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.

[19] D. Genkin, N. Nissan, R. Schuster, and E. Tromer, "Lend me your ear: Passive remote physical side channels on {PCs}," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 4437–4454.

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

[21] 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.

[22] 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.

[23] 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.

[24] S. King, "Luminous intensity of an LED as a function of input power," *ISB J. Phys*, vol. 2, no. 2, 2008.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] J. Bugeja, D. Jönsson, and A. Jacobsson, "An investigation of vulnerabilities in smart connected cameras," in *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, 2018, pp. 537–542.

[32] J. Bugeja, D. Jönsson, and A. Jacobsson, "An investigation of vulnerabilities in smart connected cameras," in *2018 IEEE international conference on pervasive computing and communications workshops (PerCom workshops)*. IEEE, 2018, pp. 537–542.

[33] J. Liranzo and T. Hayajneh, "Security and privacy issues affecting cloud-based ip camera," in *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*. IEEE, 2017, pp. 458–465.

[34] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis *et al.*, "Understanding the mirai botnet," in *26th {USENIX} security symposium ({USENIX} Security 17)*, 2017, pp. 1093–1110.

[35] K. Angrishi, "Turning internet of things (iot) into inter-

net of vulnerabilities (iov): Iot botnets," *arXiv preprint arXiv:1702.03681*, 2017.

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

[37] "Minerva github," https://github.com/crocs-muni/minerva/tree/master/poc/attack.

[38] D. Jao and L. D. Feo, "Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies," in *International Workshop on Post-Quantum Cryptography*. Springer, 2011, pp. 19–34.

[39] "Hertzbleed github," https://github.com/FPSG-UIUC/hertzbleed.

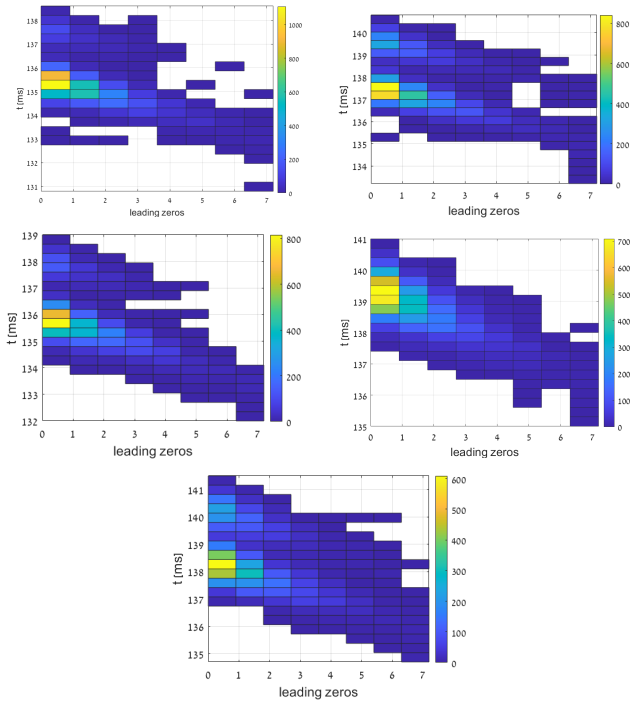

Fig. 16. Heatmaps extracted from five smart card readers which present the estimated execution times of ECDSA sign operations as a function of the number of leading zero bits in the nonce

X. Appendix - Additional Figures & Graphs

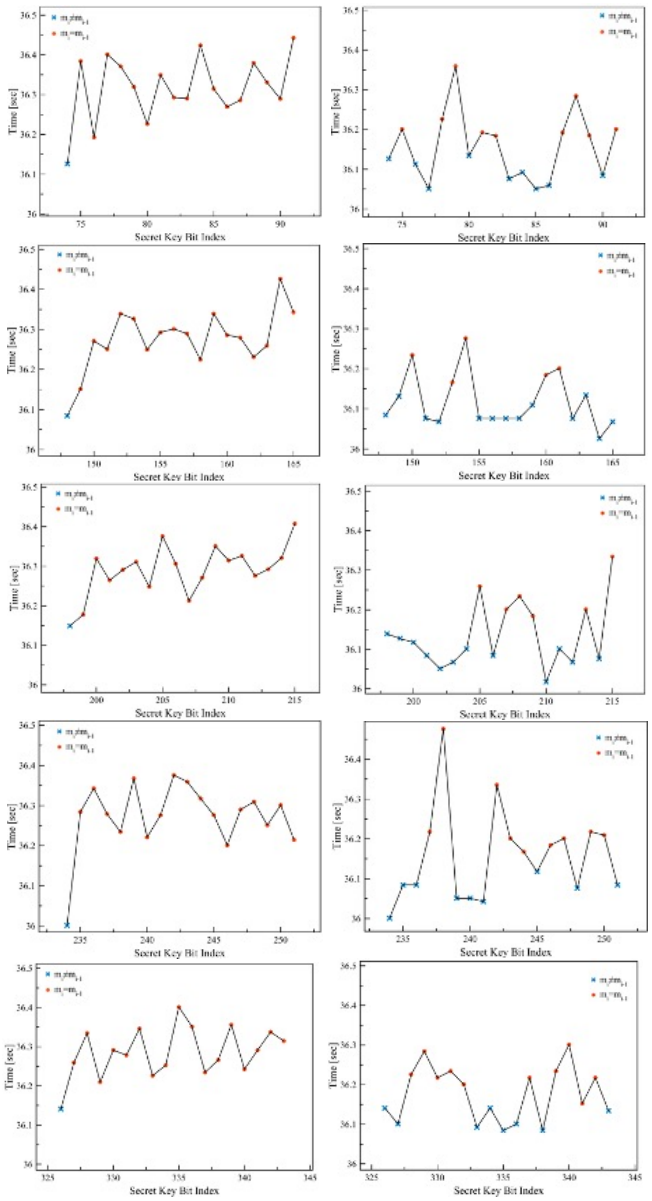

Fig. 17. The error detection (left) and correction (right) of bit indexes 76, 149, 199, 235, and 327.