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

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

[1]Cornell Tech, New York, USA
[2]Ben-Gurion University of the Negev, Beersheba, Israel
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 affects the brightness of the device's power LED. Based on this observation, we demonstrate 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 frames per second (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 inferring the device's power consumption 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 obtained by 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 used to charge the Galaxy S8 obtained by an iPhone 13 Pro Max's camera. We also discuss countermeasures, limitations, and the future of *video-based cryptanalysis* in light of the expected improvements in video camera specifications.**

1. Introduction

Over the past 25 years, research has highlighted the fact that high-end hardware can be used by attackers to recover secret keys from devices. Numerous studies have demonstrated innovative secret key extraction techniques that rely on dedicated professional equipment to capture data-dependent physical leakage from target devices. These methods employ equipment like scopes to obtain power traces (e.g., [1, 2]), software-defined radio (SDR) and probes to capture electromagnetic radiation (EMR) traces (e.g., [3–9]), photodiodes and dedicated photon detectors to obtain optical traces (e.g., [10–14]), as well as ultrasonic microphones to capture acoustic traces (e.g., [15]). While these methods have deepened understanding of the security risks high-end hardware poses to secret keys, it is important to note that the equipment required to perform these techniques is not readily available, is often costly, and demands specialized expertise to operate. These considerations naturally give rise to the question: Can easily accessible ubiquitous equipment be used to recover secret keys from devices instead of specialized hardware?

In this paper, we present *video-based cryptanalysis*, a new side-channel cryptanalytic attack that attackers can perform 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 such video footage to recover a device's secret key. This is possible, because the intensity/brightness of a device's power LED correlates with its power consumption. This correlation stems from the fact that in many devices, the power LED is connected directly to the power line of the electrical circuit, which lacks an effective means (e.g., filters, voltage stabilizers) of decoupling the correlation.

We empirically analyze the sensitivity of video cameras in the task of secret key recovery and show that they can be used to conduct cryptanalysis because: (1) the 8-bit resolution (a discrete space of 256 values) of a single RGB channel of a device's power LED in video footage is sufficient for the detection of the differences in the device's power consumption caused by the cryptographic computations, and (2) the video camera's rolling shutter can be exploited to increase the sampling rate of the video camera by three orders of magnitude to the level needed to perform cryptanalysis, i.e., from the FPS rate (60 measurements per second) to the rolling shutter rate (60K measurements per second in the iPhone 13 Pro Max), by zooming the video camera in on the power LED of the target device so that the

view of the LED fills the entire frame of the video footage. By doing so, attackers can use a standard video camera to perform cryptanalysis remotely, instead of the professional-grade sensors typically used (e.g., a scope, SDR).

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*, which are based on the type of power LED the target device has: (1) for devices with standard on/off power LEDs, attackers can obtain video footage using their smartphone's video camera (meaning that 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 (meaning that attackers can apply the attack remotely over the Internet).

Next, we demonstrate *video-based cryptanalysis* by recovering: (1) a 256-bit ECDSA key from a smart card (exploiting the vulnerability presented in [16–18]) by analyzing video footage obtained via an Internet-connected video camera located 16 meters away from the smart card; and (2) a 378-bit SIKE key from a Samsung Galaxy S8 (exploiting the vulnerability presented in [19]) by analyzing video footage obtained by the video camera of an iPhone 13 Pro Max of the power LED of Logitech Z120 USB speakers that were connected to the same USB hub used to charge the Samsung Galaxy S8.

Finally, we discuss the potential of *video-based cryptanalysis* today given the use of professional video cameras and raise concern regarding the possibility that in the near future, more devices will be vulnerable to *video-based cryptanalysis* in light of the recent and expected improvements in video camera specs (including increased shutter speed, a wider RGB space, and improved zoom capabilities).

Contributions. (1) We extend the recent CCS'23 study by *Nassi et. al* [10, 11] that recovered secret keys by using professional high-end photodiodes to obtain optical traces from power LEDs. Instead of using photodiodes, we demonstrate key recovery from power LEDs using ubiquitous commercial off-the-shelf (COTS) video cameras. The video cameras used in our study are considered more common and readily available than the equipment used to conduct cryptanalysis in prior work (e.g., scopes [1, 2], probes and SDR [3–9], near-infrared photon detectors [12–14], and ultrasonic microphones [15]). (2) We demonstrate two non-intrusive attack vectors (one requiring physical proximity and another that can be performed remotely over the Internet) for the application of *video-based cryptanalysis*, which can be exploited to perform existing and new cryptanalytic side-channel attacks. (3) We show that at least six commercial smart card readers (which we purchased on Amazon) and a smartphone leak information that can be exploited in order 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 2, we review related work. The threat model is presented in Section 3. In Section 4, we analyze the bandwidth of power LEDs captured by video cameras. In Sections 5-6, we demonstrate the application of video-based cryptanalysis and recover ECDSA and SIKE keys from various devices. In Section 7, we describe countermeasures, in Section 8, we discuss limitations, and in Section 9, we discuss our findings.

2. 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., [1, 2]) or other side effects, including EMR leakage (e.g., [3–9]), acoustic noise (e.g., [15]), and optical leakage [10–14]. These methods utilized high-end and dedicated sensors to perform cryptanalysis.

Optical Side-Channel Attacks. Optical sensors (video cameras, photodiodes, and LiDAR) were used to perform side-channel attacks to recover: (1) content from monitors [20–22], (2) keystrokes from physical and virtual keyboards [23–29], (3) the content of a conversation/meeting from light bulbs [30], a bag of chips [31], desktop ornaments [32], and other objects [33–35]. Discussion regarding the risks posed to information confidentiality stemming from the correlation between the intensity of a power LED and the power consumed by the device [36] began over 20 years ago [37]. However, prior research demonstrating methods capable of exploiting a device's power LED for data exfiltration relied on preinstalled malware [38–40] that actively triggered and controlled a device's power LED (e.g., the power LED of a keyboard [38], router [39], hard drive [40]) in order to establish optical *covert channels*. Two recent studies presented side-channel attacks against power LEDs by obtaining optical traces from power LEDs to recover (1) speech from virtual meetings [41, 42], and (2) secret keys from devices [10, 11].

Video Cameras. Many studies analyzed and discussed the risks video cameras pose to individuals' privacy in the physical world (e.g., spying [43, 44], sound/speech eavesdropping [31, 33, 35]). However, little is known about the risks video cameras pose to information confidentiality in the digital world.

3. Threat Model and Sampling Rate

3.1. 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 the power LED of a connected peripheral (*i.e., an indirect attack*) whose power consumption is also affected by the power

TABLE 1. COMPARISON OF THE TWO ATTACK VECTORS.

	Close Video Acquisition	Over-the-Internet Video Acquisition
Video Camera Required	A smartphone's video camera	Security camera
Vulnerable Type of Power LED	Type 1 or 2	Type 2
Distance	Very close distance	Depends on the video camera's zoom capabilities
Attacker Capabilities	Physical access to the target device	Capable of remotely controlling a hijacked Internet-connected video camera
Needed Ambient Light	Not dependent on ambient light	Darkness

consumption of the target device. The attacker exploits the correlation between the brightness/color 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 increase 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 cryptographic library vulnerable to a timing side-channel attack is installed on the target device and cryptographic operations are performed by the device. 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 has a power LED or is connected to another device/peripheral that has a power LED (e.g., speakers, USB hub) of one of the following types: (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 only emits light when the device is turned on. The brightness of the LED changes very slightly in response to the changes in 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 the cryptographic operations triggered.

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

Attack Vectors. We consider two types of video footage acquisition models, which are based on the power LED type. (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) of a device or connected peripheral leaks information (which correlates with the cryptographic operations). Since this attack vector requires the attacker to be very close to the target device, we assume that the attacker has access to the target device and the lens required to zoom their smartphone in on the power LED of the target device (or the connected peripheral) while it is performing cryptographic operations. Finally, we assume that in cases in which the target device does not contain an integrated power LED, the attacker can connect the target device to a peripheral with a vulnerable LED (e.g., a USB hub, speakers). (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 that is located near the target device. We further assume that the attacker can control the video camera using its API, zoom the video camera in on the power LED of the target device (or the connected peripheral), obtain video footage of the power LED, and exfiltrate the footage over the Internet to their possession. In this video acquisition model, we also assume that the device has 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 the video camera's location. Moreover, we assume that the video recording is made in darkness. Table 1 presents the differences between the two attack vectors.

Significance. We note that the threat model is non-intrusive (in contrast to models' relying on power traces which often require connecting probes to the circuitry of the device or attaching monitoring equipment directly to the power line); relies on a smartphone's video camera or Internet-connected video camera, which are common/ubiquitous equipment (as opposed to other cryptanalysis methods that rely on specialized equipment including SDR, photodiodes, scopes, and probes); can be applied over the Internet; and can endanger devices that do not even have a power LED via the power LED of connected peripherals (e.g., speakers, USB hub splitters).

3.2. Increasing a Video Camera's Sampling Rate

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 (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

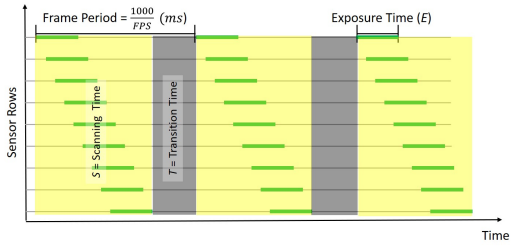

Figure 1. A video camera's rolling shutter. In every frame period, the rolling shutter scans an object vertically and exposes the shutter for a short time determined by E. The time it takes to scan a single frame is denoted by S. Between two consecutive frames there is a transition period, during which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

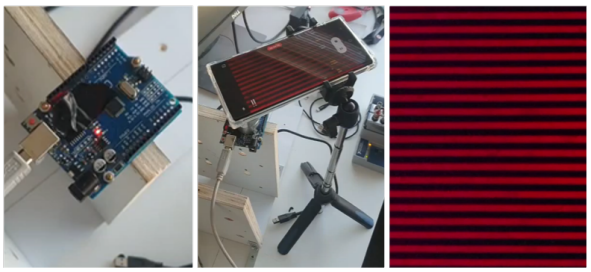

Figure 2. Increasing the sampling rate from the FPS rate of the video camera to the shutter rate: An Arduino's LED flickering at 4 kHz (left) is recorded by a Samsung Galaxy S22 Ultra using a lens that increases the size of the LED so that it fills the entire screen (middle). A frame of the video recorded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

of a scene taken at a specific point in time but rather is composed of multiple snapshots taken of vertical/horizontal parts of the scene at different times. Fig. 1 illustrates this process: With a vertical rolling shutter, the video camera scans the captured object row-by-row sequentially from top to bottom at different times according to a configurable shutter speed (E) which determines the amount of time that the video camera is exposed to light. Because in a sensor with a rolling shutter each row (or group of adjacent rows) 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. This technique has also been used in prior studies to increase the sampling rate of a video camera [31, 33, 35].

In *video-based cryptanalysis*, attackers exploit the rolling shutter to increase the number of measurements per second (sampling rate) obtained of 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 the entire amount of 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 increase 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 obtained are not ideal, because they are not uniformly sampled across time. As can be seen in Fig. 1, there are transition periods (denoted 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 distribution as semi-uniform: The sampling is uniform within a frame but is not uniform across the video due to the transitions between frames.

3.3. Determining the Transition and Scanning Time

In some cases, the exact transition time and scanning time must be determined in order to perform cryptanalytic attacks (e.g., as demonstrated later in Section 5), and we now explain how attackers can empirically determine these times.

Experimental Setup. We programmed an Arduino Uno to modulate a 4 kHz flicker using the Arduino's red integrated 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 the shutter speed at $\frac{1}{12{,}000}$.

Calculating the Scanning and Transition Time. 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) that result from the flickering LED. The scanning time can be calculated by multiplying the amount of time that the power LED was flickering (250 microseconds in our experiment) by the number of transitions between the on/off states in the frame (39 transitions in the frame presented in Fig. 2). In our case, the scanning time is $S = 9.75$ milliseconds. Since $T = \frac{1000}{FPS} - S$, the transition time of the video camera used in this experiment is $T = \frac{1000}{60} - 9.75 = 6.91$ milliseconds.

Note that the process of determining the scanning time (S) and transition time (T) 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 described above to determine T and S. In cases in which an attacker performs the attack over the Internet, using a remote video camera, the attacker would need to purchase the same camera (used to perform the remote attack) in order to empirically determine T and S (unless such information is provided in the camera's specs).

4. 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. In the experiments described in this section, we used two functions to create a signal from a given channel (red, green, or blue) in the video footage: $Average-Rows$ and $Average-Frames$ (see Algorithm 1). The $Average-Rows$ 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. The $Average-Frames$ 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. The $Average-Rows$ function is mainly used for video footage obtained from a type 2 power LED, while the $Average-Frames$ function is mainly used for video footage obtained from a type 1 power LED.

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

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

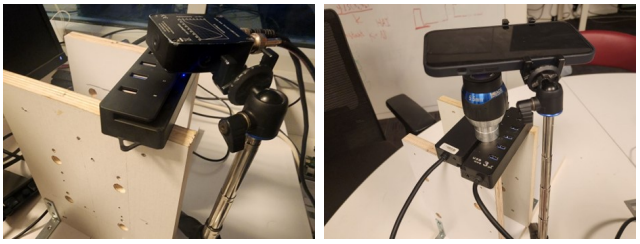

Figure 3. Recovering a frequency scan using a photodiode (left) and a smartphone (right), by capturing video footage of the power LED of a USB hub.

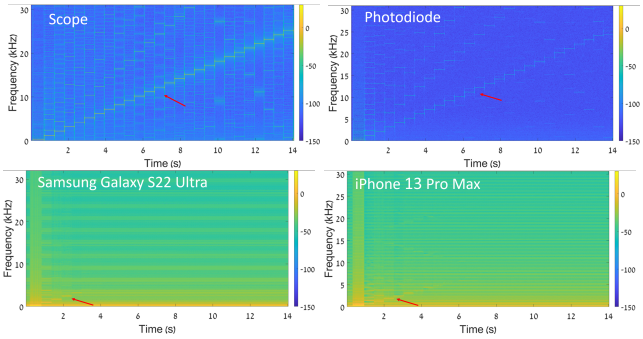

Figure 4. A function generator was used to modulate a frequency scan over the power supplied to a USB hub. Pictured are the spectrograms extracted from the scope, photodiode, and blue channel of the video cameras of the two smartphones used in these experiments.

4.1. 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 1,000 Hz (starting from 200, 1,200, 2,200, ..., 25,200 Hz), each of which was modulated for a half of a second. 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; in each experiment a different smartphone was used: 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. In each experiment, the smartphone was placed on the USB hub's power LED, and its video camera was used to film the LED, using a lens, as seen in 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. We also obtained a power trace by inserting an

TABLE 2. COMPARISON OF THE OPTICAL SNR OBTAINED DIRECTLY AND INDIRECTLY FROM A RASPBERRY PI RUNNING THE CRYPTOGRAPHIC LIBRARIES TARGETED BY [17, 19, 45].

	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

adapter between the function generator and the USB using a Digilent Analog Discovery 2 (scope).

Results. First, we extracted a spectrogram from the optical trace obtained by the photodiode (see 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 channel and extracted a spectrogram for each signal (see Fig. 4). As can be seen in the figure, only the first six sine waves appear in the spectrogram extracted from the video footage obtained by the Samsung Galaxy S22 Ultra, and only the first nine sine waves appear 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 their maximum shutter speed. Moreover, as can be seen in the spectrogram extracted from the Samsung Galaxy S8, the sixth sine wave that was 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 power LED, which stems from the fact that the video cameras do not capture the power LED during transitions between frames.

Then, we examined the SNR in the examined spectrum, comparing the SNR of the video cameras of the two smartphones, the optical trace, and the power trace generated by the function generator (see Fig. 5). Based on the results presented in the figure, 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; and (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).

4.2. 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.

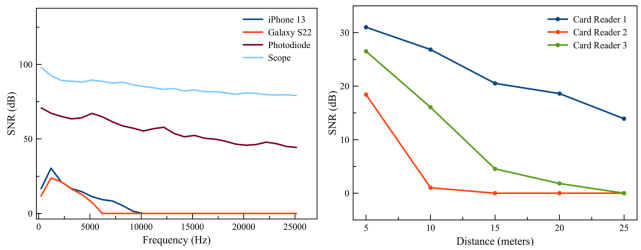

Figure 5. Left: SNR obtained from various devices. Right: SNR of smart card readers obtained from various distances.

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 an SIKE operation).

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

Results. We applied the $Extract - Frames$ function on each video, extracted nine signals, and calculated the SNR for the signals. As can be seen in the results presented in Table 2, the target library under attack greatly affects the optical SNR: the SNR obtained from an 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). Based on these results, we concluded that: (1) the optical leakage is present both in the power LED of the device that performs the cryptographic operations and the power LED of peripherals connected to the device (the USB hub and USB speakers); (2) the power LED of a connected peripheral can 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 $\sim$1.1-4.3 dB, while the USB speakers decrease the SNR by $\sim$0.7-2.0 dB (compared to the SNR obtained directly from the power LED of the Raspberry Pi); and (3) even devices that do not contain an integrated power LED or whose integrated power LEDs do not leak information may be vulnerable to *video-based cryptanalysis* when they are connected to another peripheral with a vulnerable power LED.

4.3. 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 purchased from Amazon (to ensure confidentiality in 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 paper in coordination with the manufacturers). Each of the smart card readers contains an indicative power LED (type 2, which is described earlier in the paper), allowing the power LED color changes to be detected from a distance. 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 over-the-Internet video 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 every 200 milliseconds, using the smart card; the color of the power LED of the smart card readers changes in response to an ECDSA sign operation. Then, we placed the SUNBA video camera 25X optical zoom 5MP smart security dome [46] 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 idle state values 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 by analyzing the video footage of a type 2 power LED from a range of up to 25 meters for a specific smart card reader.

4.4. 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) over-the-Internet video acquisition, in which a remote video camera is used to film the power LED.

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) and 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 as the previous experiment: 0, 300, and 3000 lux.

TABLE 3. THE INFLUENCE OF AMBIENT LIGHT AND DATA ACQUISITION ON THE OPTICAL SNR.

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

Results. We analyzed the six videos and calculated the SNR. Based on these results, which are presented in Table 3, we concluded that: (1) in close video acquisition (where 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 over-the-Internet video acquisition (where the video camera is placed a distance away 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.

5. Recovering ECDSA Keys

In this section, we describe the Minerva attack [17] we performed to demonstrate the recovery of the 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 16 meters.

Minerva Attack. As observed by [16–18], many common cryptographic libraries optimize the computation time of ECDSA signing by truncating the leading zeros. This optimization results in a variable number of loop iterations that is associated with the 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, by 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 [17] 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 reader's power LED (as opposed to the original applications [17, 18] of the attack which relied on CPU measurements of the

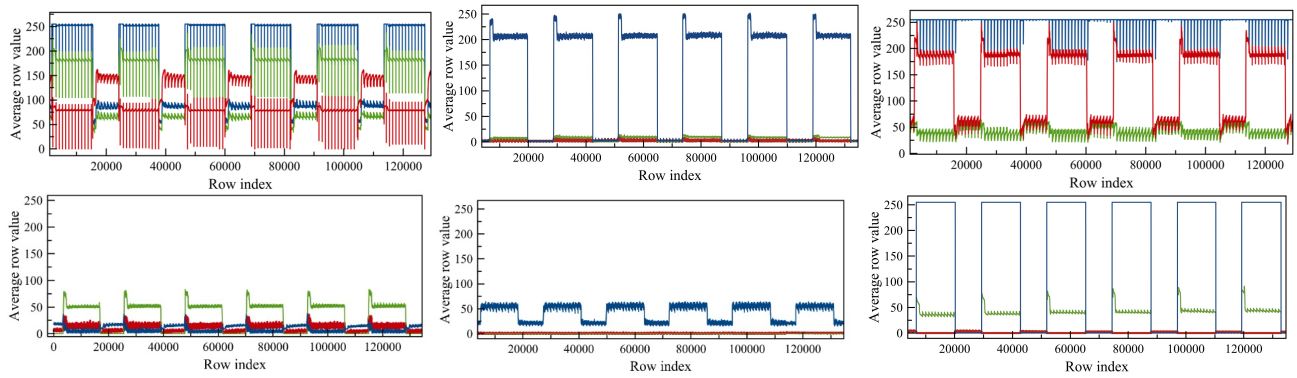

Figure 6. The RGB values extracted from 96 consecutive frames (with 103,680 rows) from video footage of the smart card reader's power LED during the execution of six different ECDSA signing operations, separated by 200 ms sleep episodes. The colors represent the associated RGB channel.

Figure 7. An obfuscated picture (we blurred the visual identifiers of the manufacturers) of the six smartcard readers, from which we recovered ECDSA keys by analyzing their power LED.

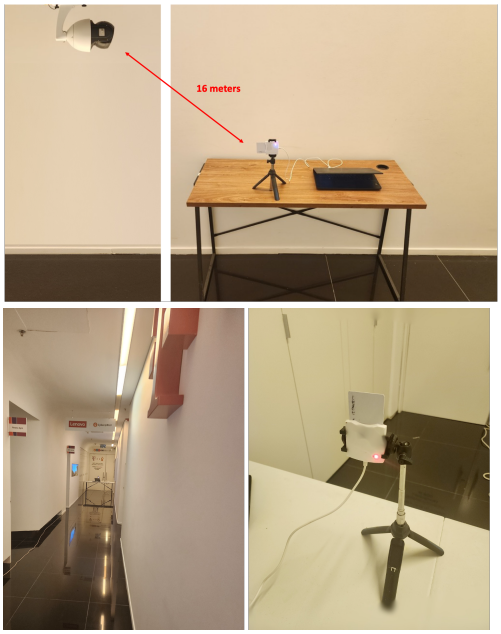

Figure 8. Experimental setup. Top: The video camera located 16 meters away from the smartcard reader. Bottom: As seen on the left, the smart card reader (pointed at by the red arrow) is pictured from 16 meters away. On the right is an image of the smart card reader's power LED.

ECDSA signing operations obtained using code installed on the target laptop/computer).

5.1. Identifying ECDSA Operations from Various Smart Card Readers

First, we show that six commercial smart card readers with an indicative power LED (type 2) that we purchased on Amazon leak information about the execution time of the ECDSA signature from their power LED (see Fig. 7). The color of these smart card readers' power LED changes in response to an operation triggered by a connected laptop.

Experimental Setup. We conducted six 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 [46] 20 cm away from the smart card reader. The Internet-connected video camera's remote API was used, and the camera's lens 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).

Results. The six videos obtained in the experiment described above were processed. For each video, we applied Algorithm 1, which creates a signal from the rows of frames by averaging the RGB values in each row in a

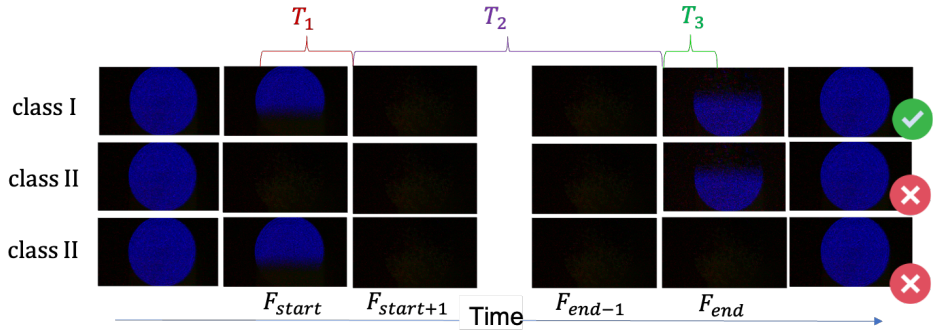

Figure 9. Examples of ECDSA series extracted from video footage of a smart card reader's power LED. Top: a series of signatures' frames that started and ended during the rolling shutter's scanning time (a Class I series). Middle: A series of frames that started during the transition time between frames (a Class II series). Bottom: A series of frames that ended during the transition time between frames (a Class II series).

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 six extracted signals, and (2) the thresholds that differentiate the signing and sleep episodes vary depending on the signal extracted from the smart card reader and the RGB channels.

5.2. Recovering ECDSA Keys from a Distance

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

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 16 meters away from the smart card reader (the experimental setup can be seen in Fig. 8). We used the Internet-connected video camera's remote API and directed the camera lens at the power LED of the smart card reader and zoomed in on the power LED until it filled the entire frame (see Fig. 8). 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 35 different videos. Each video was an hour and fifty minutes long and consisted of 300 ECDSA signatures.

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 (1) the idle state of the smart card reader (i.e., the frame consists of just blue rows), or (2) the smart card reader used for an ECDSA signing operation (i.e., the frame consists of at least one red row). The function percentage fulfills 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 sign operation (< 37.5). The threshold separating the two states was determined based on the experiment described in Section 3.3 (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 the idle state of the smart card reader (i.e., 0% of the rows are blue/red, and 100% of the rows are blue) or was used for signing (i.e., some of the rows are blue/red, and some of the rows are blue) as follows:

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

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 $(f^1_{start}, ..., f^1_{end}, \quad ..., \quad f^{10,500}_{start}, ..., f^{10,500}_{end})$, 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 series of signatures' frames that started and ended during the scanning time; such signatures can be identified based on the switch between the blue and red colors somewhere in the middle of the first (f^i_{start}) and last (f^i_{end}) frames, as can be seen in Fig. 9; and **Class II** - a series of signatures' frames that started or ended during the transition time (which is not captured by a frame); such signatures can be identified based on the completely black color in the first (f^i_{start}) or last (f^i_{end}) frame (see Fig. 9).

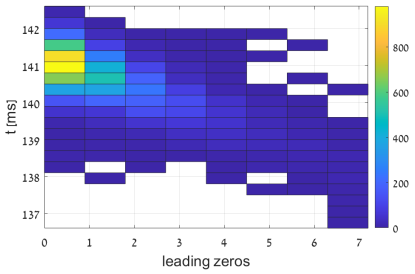

Figure 10. A heat map of the estimated execution times of the ECDSA sign operations as a function of the number of leading zero bits in the nonce.

The class of each signature $ECDSA_i$ with its associated frame series $f^i_{start}, ..., f^i_{end}$ was determined by examining whether 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 examining whether their values according to:

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

Estimating the ECDSA Signature Time from Frame Series. 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 signatures occurred during the transition between the frames and are not captured in any video frames. Since the process of performing lattice reduction and time extraction is highly sensitive to errors, we filtered the 2,674 Class II signatures from the data.

Next, we empirically computed the video camera's scanning (S) and transition (T) times by performing the experiment described in Section 3.3. Based on that experiment, we determined that $S = 13.8$ ms and $T = 2.8$ ms.

Then, we computed the execution time of the frame series of the remaining 7,826 Class I signatures. For every $ECDSA_i$ signature with associated frames' series $video_i = f^i_{start}, ..., f^i_{end}$ that started at index $start$ and ended at index end, we computed the signing time by applying the SigningTime function (Algorithm 2) as follows: SigningTime($video = video_i$, $scan = 13.8$, $trans = 2.8$, $threshold = 37.5$, $channel = 1$).

Algorithm 2 calculates the signing time of an ECDSA signature $ECDSA_i$ as the sum of $T_1 + T_2 + T_3$ (see Fig. 9). The algorithm 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 to the product. 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 sum of $T_1 + T_2 + T_3$ is returned as the signing time for $ECDSA_i$.

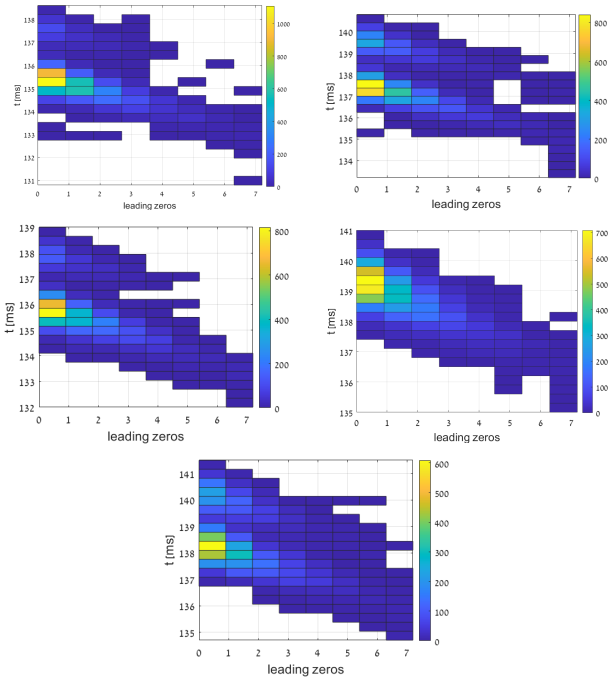

Figure 11. Heat maps extracted from five smart card readers which present the estimated execution times of the ECDSA sign operations as a function of the number of leading zero bits in the nonce.

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 cutoff to distinguish the idle/sign states
Output: signal: a time series of rows' average values
procedure SIGNINGTIME(video, ch, scan, tran, thresh)
$T_1 = Percentage(f_{start}, ch, thresh) \times$ scan + trans
$T_2 = (end - start + 1) \times$ (scan + trans)
$T_3 = Percentage(f_{end}, ch, thresh) \times$ scan
return $T_1 + T_2 + T_3$
procedure PERCENTAGE($frame, ch, thresh$)
signal = Average-Rows(frame, ch)
sum = 0
for (i = 0; i < length(signal); i++) **do**
if (signal[i] < thresh) **then**
sum++
return sum/length(signal)

Results We computed the ECDSA signing time for the 7,826 Class I signatures by applying Algorithm 2 on the

7,826 associated videos. The heat map presented in Fig. 10 shows that the signatures with the shortest estimated time (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 [47]) and recovered the full 256-bit ECDSA key in two minutes.

We also recovered the ECDSA key from the power LED of the other five smart card readers (using the same video camera) from varied distances of 5-10 meters. An obfuscated picture of the six smartcard readers and the heat maps extracted from the power LED of the other five smart card readers can be seen respectively in Figs. 11 and 7.

6. 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 [48]. We perform the Hertzbleed attack [19] 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 contained the SIKE key).

Hertzbleed Attack. As seen in the paper presenting the Hertzbleed attack [19], 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 with computations. This results in differences in the execution time associated with the data processed; these differences can be amplified to a distinguishable level (at a granularity of milliseconds) by executing a large number of operations in parallel (see [19] for details).

We performed 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 was used to charge the smartphone. By doing so, we show that the ARM architecture is also vulnerable to the Hertzbleed attack (the original attack 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 bits 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 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 determining whether $m_i = m_{i-1}$ or not, based on a timing threshold.

6.1. Determining the Timing Threshold

First, we examine whether the behavior (the time difference) seen on the x86 architecture reported in the original paper on Hertzbleed [19] is also seen on the ARM architecture of the Samsung Galaxy S8.

Experimental Setup. We downloaded the code published in the Hertzbleed repository [49], 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 execution time of the Samsung Galaxy S8's CPU 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 8,800 SIKE operations divided into eight iterations, where in each iteration 1,100 SIKE operations were executed on 1,100 threads spawned concurrently. In total, we executed 1,337,600 SIKE operations that consisted of 1,216 iterations (each of which consisted of 100 SIKE operations); eight iterations were used to measure the execution time of each bit. In these experiments, the execution time of the SIKE iterations was calculated using CPU measurements obtained with the code downloaded from the official Hertzbleed repository.

Results. For each bit, we only used the last seven iterations (which consisted of 7,700 SIKE decapsulation operations) and disregarded the first iteration (which consisted of 1,100 SIKE decapsulation operations), since we found that the first iteration was unstable and mainly used to overload the CPU in order to produce stable execution differences associated with the data processed in the next seven iterations. As a result, 12.5% of the measurements were filtered out.

Fig. 12 presents the distribution of the execution times of the 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 case of a switch ($m_i \neq m_{i-1}$) and non-switch ($m_i = m_{i-1}$). The execution times in red represent cases of a switch ($m_i \neq m_{i-1}$), with mean = 36.354 and standard deviation (STD)

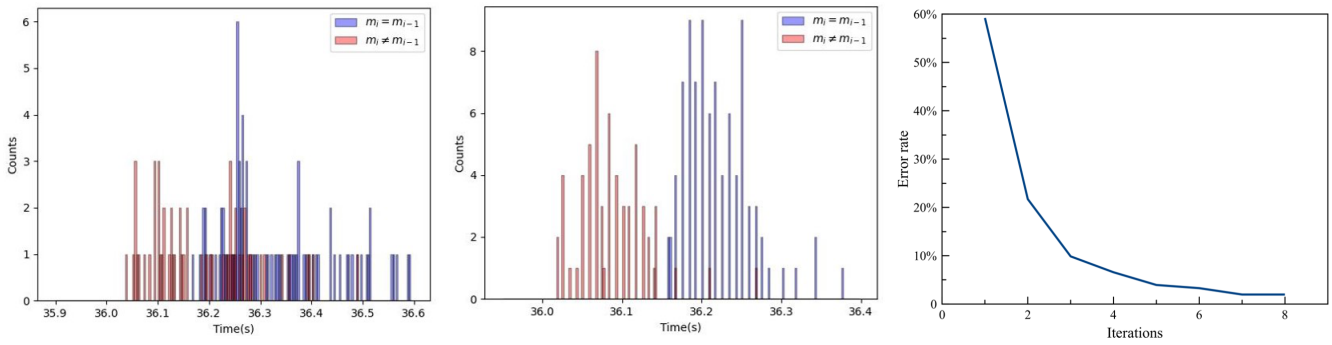

Figure 12. A histogram of video-footage-based estimations of the minimal running time based on one iteration (left) and eight iterations (center). The error rate (when a threshold of 36.15 is used to distinguish between $m_i = m_{i-1}$ and $m_i \neq m_{i-1}$) vs. the number of iterations used to determine the minimal running time (right).

= 0.7478, and the execution times in blue represent cases of a non-switch ($m_i = m_{i-1}$), with mean = 36.527 and STD = 0.8211.

Therefore, we examined the seven iterations of each bit to identify the one with the shortest execution time. The distribution of the 152 bits (based on the minimal execution time for the associated iterations) is presented in Fig. 12. The execution times in red represent cases of a switch ($m_i \neq m_{i-1}$), with mean = 36.092 seconds and STD = 0.073, and the execution times in blue represent cases of a non-switch ($m_i = m_{i-1}$), with mean = 36.223 seconds and STD = 0.084. As seen in Fig. 12, a threshold of 36.15 seconds 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 (2-8) used to produce the minimal execution time with a threshold of 36.15 seconds (the results are presented in Fig. 12). As can be seen, 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 [19] on the x86 architecture is also observable on the ARM architecture at the granularity of a series of 1,100 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.

6.2. 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) of 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.

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 downloaded and installed the application Rec [50] on the iPhone 13 Pro Max. This application allows

Figure 13. 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).

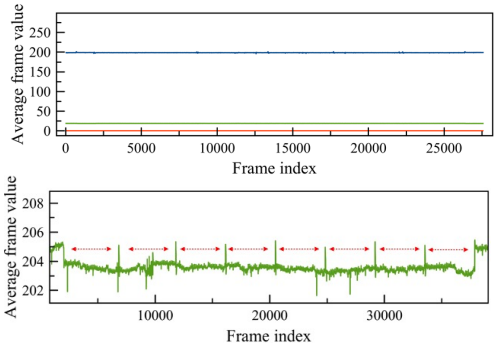

Figure 14. Top: The RGB values of eight SIKE iterations extracted from video footage. |Bottom: Zooming in on the green channel (Bottom).

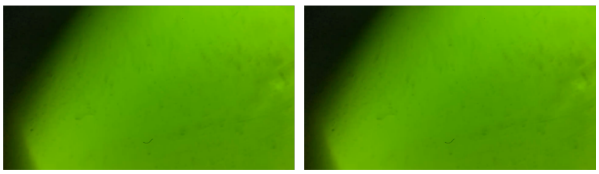

Figure 15. Snapshots taken from the video footage of the power LED of the speakers captured by the iPhone. The difference in the brightness of the power LED of the speakers when the smartphone is idle and when it performs SIKE operations cannot be discerned by the human eye.

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 (signal [j] > threshold) **then**
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

the user to configure the shutter speed of the recorded video. We used the camera of the iPhone and zoomed the video camera in on the power LED of the USB speakers (via the application) with a lens (see Fig. 13). Using the application, we obtained video footage (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 adaptive chosen ciphertext attacks 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 Hertzbleed paper [19] (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.

Processing the Signal. We processed the video footage obtained as we triggered the adaptive chosen ciphertext attack as follows: We applied the $MinIterTime$ function (see Algorithm 3) on the video footage obtained. This function calls $Average-Frames$ to extract a signal based on the green channel. An example of the signal extracted from one of the videos is presented in Fig. 14; as can be seen in the image presenting the green channel, the eight iterations can be detected by analyzing the RGB values, although the difference between the brightness of the power LED when the smartphone performs SIKE operations and when the smartphone is idle cannot be discerned by the human eye

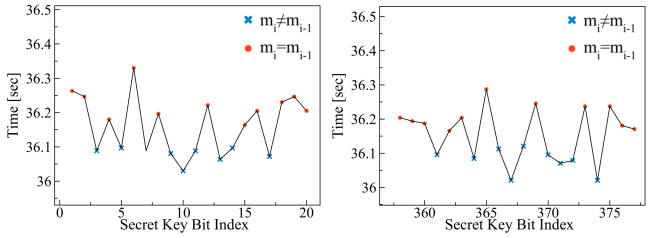

Figure 16. Minimum times used to extract the first 20 bits (1 to 20) and last 20 bits (358 to 377) of the SIKE key based on eight iterations.

(see Fig 15).

Next, $MinIterTime$ calls $Extract-Indexes$ to extract the indexes of the frames associated with the beginning of the iterations. This is done by determining whether the values of the indexes of the frames are greater than 204.9. We note that due to the added noise, the $Extract-Indexes$ function may return more than eight indexes (i.e., the function may produce errors). Next, $MinIterTime$ computes the number of frames between every two indexes (due to the errors that may have been produced by $Extract-Indexes$) and calculates the associated running time by multiplying the number of frames by $\frac{1}{fps}$ (the fraction of a second it takes to capture a frame, including the transition time). The function filters any result that is under 36 seconds and caused by the added errors. Finally, $MinIterTime$ returns min, the minimal running time. We determined the value of the i-th index of the key according to the value of min:

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

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 the Hertzbleed paper [19], 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 zeros will appear from this point on). We verified that our guess for the first index was correct by using the next bit index (where $j = 1$), which was predicted to be a switch case. The minimal values among the seven iterations of the first 20 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. 16.

The 378 bits of the key were recovered with six errors that we encountered and corrected during the recovery process using an error detection and correction algorithm.

Error Detection and Correction. We used an error detection and correction algorithm to detect and correct the six errors we encountered during the key recovery. The algorithm was implemented based on the error detection and correction algorithm suggested in the Hertzbleed paper [19]. 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 [19] for more

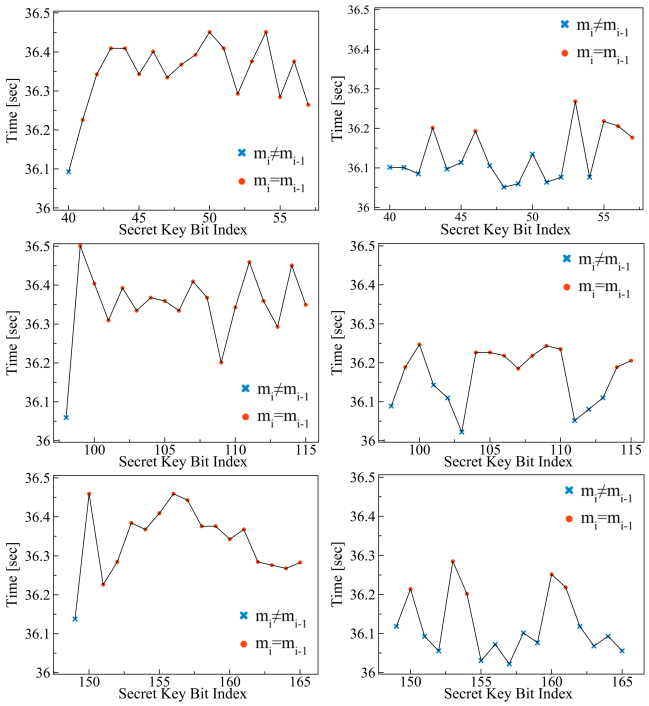

Figure 17. The error detection (left) and correction (right) of bit indexes 41, 101, and 151.

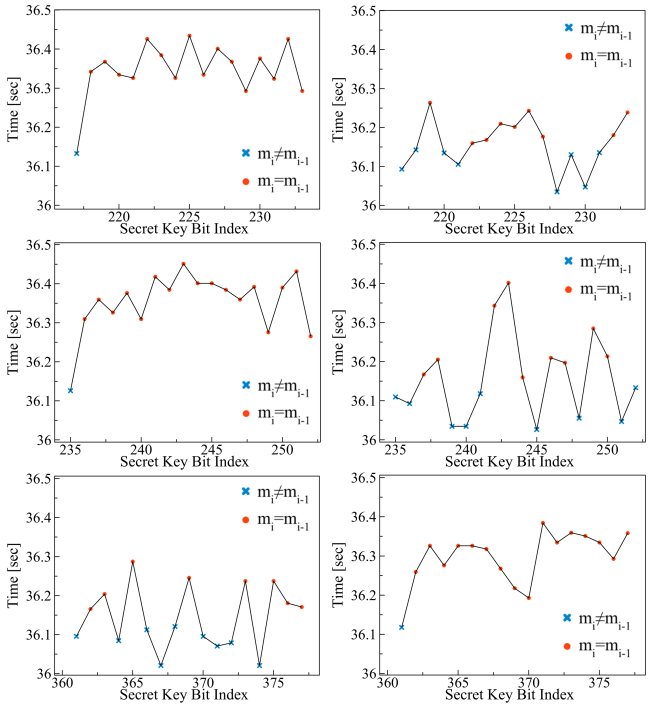

Figure 18. The error detection (left) and correction (right) of bit indexes 218, 236, and 361.

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 (cases in which execution time crossed the threshold of 36.15 seconds). A chain of 17 consecutive non-switch bits is expected to be the result of an error in a recovered bit with 99.9992% probability (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 as the key). Figs. 17 and 18 present six chains of 17 consecutive non-switch bits that we encountered during the key recovery process (the errors appeared in indexes 41, 101, 151, 218, 236, and 361 and created a chain of 17 non-switch cases for each error). For these chains (which we encountered during the recovery), we repeated the sampling process. This was done by sampling each index again, starting from the last index in which we encountered a switch case, before the beginning of the chain (indexes 28, 74, 148, 198, 234, and 326). The corrected classifications for the chains of the six bits are also presented in Figs. 17 and 18.

7. Countermeasures

In this section, we describe several methods that can be used to mitigate or prevent *video-based cryptanalysis*.

Software-Based Countermeasures. The best way to prevent attackers from recovering secret keys from devices is to ensure that the cryptographic library used does not leak any information that can be exploited to recover the key. However, we note that attackers can still apply *video-based cryptanalysis* using zero-day attacks found in the most updated cryptographic libraries.

Hardware- and Firmware-Based Countermeasures. We differentiate between two types of power LEDs: type 1 power LEDs (standard on/off power LEDs) and type 2 power LEDs (power LEDs that provide an indication regarding CPU operations by changing their color). We advise manufacturers to use constant-time LED indication for type 2 power LEDs (that are integrated into devices), i.e., use constant time flickering which is independent of the data being processed by the device (e.g., always use a 100 millisecond on/off LED blink). In many devices, a type 1 power LED is connected directly to the power line of the device. 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 capacitor can be integrated parallel to the power LED indicator; in this case, the capacitor behaves as a low-pass filter. This is an inexpensive solution for reducing the fluctuations in power consumption. However,

in devices with high power consumption, the integrated capacitor's capacitance must be large enough to support the power supplied to the device.

Consumer Side Countermeasures. Remote attacks can be prevented by placing black tape over a device's power LED. However, attackers can easily remove the tape from the LED if they apply the attack in close video acquisition.

8. 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). 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. However, we note that professional video cameras already support a higher shutter speed (e.g., Fujifilm X-H2 supports a shutter speed of $\frac{1}{180,000}$).

Semi-Uniform Distribution of Sampling. In a video camera with an FPS rate, the time provided by $\frac{1}{FPS} = S + T$ consists of S, which denotes the 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 samples the distribution of the power LED's intensity within a frame uniformly, the video of the power LED does not reflect 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). Some cryptographic operations may start or end during the transition time (which will prevent attackers from accurately calculating the exact time that the cryptographic operation was performed). Attackers can use one of the following two approaches to address this: (1) Estimate the missing beginning/end time of an operation as 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) Accurately calculate the 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 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 power LED (while triggering the same cryptographic operation) to ensure that the beginning and end of a cryptographic operation are captured in the frames.

Low SNR from Integrated Power LEDs. We note that some devices do not leak fine-grained information from their integrated power LEDs or leak fine-grained information with a very low SNR. 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 does leak fine-grained information that 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 6.

Limited Sampling Sensitivity. We note that the video cameras used in this research (with an 8-bit RGB depth) are less sensitive than photodiodes, which can be sampled with a 16-bit ADC and can capture much more subtle changes in the brightness of the power LED (see Fig. 5). Therefore, only a portion of the cryptanalytic attacks that can be applied using a photodiode can be applied with a video camera. We note that attackers can improve the sensitivity of video footage by using professional video cameras with greater sensitivity (e.g., a video camera with a 12-bit RGB depth already exists).

Limited Exposure. The number of devices exposed to *video-based cryptanalysis* is affected by various factors. One factor limiting the exposure of this attack is the fact that *video-based cryptanalysis* relies on the existence of a cryptographic vulnerability in a cryptographic library. With that in mind, *video-based cryptanalysis* can be applied using known attacks (e.g., Minerva and Hertzbleed) against old cryptographic libraries installed on devices. However, attackers can also apply *video-based cryptanalysis* to recover secret keys using zero-day cryptographic vulnerabilities that exist in the most updated cryptographic libraries (just as in the past, when the cryptographic libraries of that time were considered the most up-to-date versions until a vulnerability was found). Another factor that may limit the exposure of the attack is related to over-the-Internet video acquisition (using an Internet-connected video camera). This attack vector can only be applied from a distance, against devices that contain a type 2 power LED and are located in dark rooms (see Fig. 3), using video cameras that have the required optical zoom capabilities to capture the LED from the distance.

Long Application Time. We note that some of the attacks (e.g., Hertzbleed) may take a long time to perform. While this has negligible implications on remote video acquisition, it has a substantial effect on close video acquisition in that it requires the attacker to be near the target device when the video footage is obtained (during filming). However, we note that this is not only a disadvantage of *video-based cryptanalysis* but rather it is a disadvantage of the time it takes to apply the cryptanalytic attack used to recover the secret key even with the use of specialized hardware (e.g., SDR, scope, photodiode).

9. Conclusions, Discussion, and Disclosure

In this research, we showed how a COTS video camera can be used to extract secret keys from a device by analyzing the device's power LED. In doing so, we raise awareness regarding the ability of adversaries to recover secret keys from devices whose power LEDs leak information, without the use of specialized hardware (scopes, photodiodes, probes, SDR, or ultrasonic microphones).

One might argue against the novelty of the paper and claim that Minerva and Hertzbleed are well-known crypto-

graphic vulnerabilities that we did not discover. 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. One might also question the contribution of *video-based cryptanalysis*, arguing that this method can really only be used 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 obtain time measurements by calculating the time elapsed between the API request and the API response. In response to this, 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 network's latency. For example, in the FAQ section of the GitHub repository published by the authors of the Minerva attack [17], the authors mentioned that they were unable to perform the attack remotely (using network measurements) for this reason (see *Is this exploitable remotely?* in Minerva's GitHub [47]). In contrast, *video-based cryptanalysis* can be used to apply the Minerva attack remotely over the Internet using video footage obtained by a hijacked video camera.

We also raise concern regarding the potential of *video-based cryptanalysis* today, given recent improvements in video camera specifications. In our research, we used COTS video cameras to obtain the video footage (i.e., video cameras with an 8-bit space for a single RGB channel, full HD resolution, and a shutter speed of $\frac{1}{60{,}000}$). However, new versions of smartphones already support 10-bit resolution video footage (e.g., iPhone 14 Pro). Moreover, professional video cameras with a resolution of 12-14 bits already exist and may provide much greater sensitivity, allowing attackers to perform additional attacks that require the ability to detect very subtle changes in 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 and 36X) and are likely already widely deployed, allowing attackers to obtain video footage of a target device's power LED from a greater distance than that demonstrated in this paper. Finally, new professional video cameras support a shutter speed of $\frac{1}{180{,}000}$ (e.g., Fujifilm X-H2) which may allow attackers to obtain measurements at a higher sampling rate, potentially exposing other devices to the risk of *video-based cryptanalysis*.

Considering the expected advancements in COTS smartphones and security video cameras (based on Moore's law) in the near future and the fact that more and more functional IoT devices with limited CPU capabilities (e.g., sensors, home appliances) are being deployed each day, we expect that the number of devices exposed to *video-based cryptanalysis* will increase each year (unless precautionary measures are employed to protect electrical circuits). Given this, we believe that the infosec/security community needs to encourage manufacturers to build resilient devices that are robust against the recovery of keys using COTS devices.

We disclosed our findings to the manufacturers of the devices used in our study via their bug bounty programs and contact us email addresses (except for one manufacturer for which we were unable to find any information on the web). A few manufacturers responded to our email and asked us for additional details which we shared with them. We recommend that other hardware manufacturers empirically test their devices to determine if they are vulnerable to *video-based cryptanalysis* and redesign their electrical circuits (according to the suggestions provided in Section 7) as needed. We are, however, uncertain whether they will choose to do so, as some solutions may increase the manufacturer's overall costs, 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 examining the potential of professional video cameras to recover cryptographic keys and the influence of video compression algorithms on the quality of the video footage used to recover the key.

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. We would like to Daniel Genkin, Eran Tromer, and Jan Jancar for providing us with valuable insights.

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] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] B. Nassi, O. Vayner, E. Iluz, D. Nassi, O. H. Cohen, J. Jancar, D. Genkin, E. Tromer, B. Zadov, and Y. Elovici, "Optical cryptanalysis: Recovering cryptographic keys from power led light fluctuations," Cryptology ePrint Archive, Paper 2023/1068, 2023, https://eprint.iacr.org/2023/1068. [Online]. Available: https://eprint.iacr.org/2023/1068

[11] B. Nassi, O. Vayner, E. Iluz, D. Nassi, J. Jancar, D. Genkin, E. Tromer, B. Zadov, and Y. Elovici, "Optical cryptanalysis: Recovering cryptographic keys from power led light fluctuations," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, 2023, pp. 268–280.

[12] 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.

[13] 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.

[14] J. Ferrigno and M. Hlaváč, "When aes blinks: introducing optical side channel," *IET Information Security*, vol. 2, no. 3, pp. 94–98, 2008.

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

[16] B. B. Brumley and N. Tuveri, "Remote timing attacks are still practical," in *Computer Security–ESORICS 2011: 16th European Symposium on Research in Computer Security, Leuven, Belgium, September 12-14, 2011. Proceedings 16*. Springer, 2011, pp. 355–371.

[17] 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.

[18] 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.

[19] 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.

[20] M. G. Kuhn, "Optical time-domain eavesdropping risks of crt displays," in *Proceedings 2002 IEEE Symposium on Security and Privacy*. IEEE, 2002, pp. 3–18.

[21] M. Backes, M. Dürmuth, and D. Unruh, "Compromising reflections-or-how to read lcd monitors around the corner," in *2008 IEEE Symposium on Security and Privacy (sp 2008)*. IEEE, 2008, pp. 158–169.

[22] M. Backes, T. Chen, M. Dürmuth, H. P. Lensch, and M. Welk, "Tempest in a teapot: Compromising reflections revisited," in *2009 30th IEEE Symposium on Security and Privacy*. IEEE, 2009, pp. 315–327.

[23] Y. Xu, J. Heinly, A. M. White, F. Monrose, and J.-M. Frahm, "Seeing double: Reconstructing obscured typed input from repeated compromising reflections," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 1063–1074.

[24] R. Raguram, A. M. White, Y. Xu, J.-M. Frahm, P. Georgel, and F. Monrose, "On the privacy risks of virtual keyboards: automatic reconstruction of typed input from compromising reflections," *IEEE Transactions on Dependable and Secure Computing*, vol. 10, no. 3, pp. 154–167, 2013.

[25] R. Raguram, A. M. White, D. Goswami, F. Monrose, and J.-M. Frahm, "ispy: automatic reconstruction of typed input from compromising reflections," in *Proceedings of the 18th ACM conference on Computer and communications security*, 2011, pp. 527–536.

[26] D. Balzarotti, M. Cova, and G. Vigna, "Clearshot: Eavesdropping on keyboard input from video," in *2008 IEEE Symposium on Security and Privacy (sp 2008)*. IEEE, 2008, pp. 170–183.

[27] K. Mowery, S. Meiklejohn, and S. Savage, "Heat of the moment: Characterizing the efficacy of thermal camera-based attacks," in *Proceedings of the 5th USENIX conference on Offensive technologies*, 2011, pp. 6–6.

[28] Q. Yue, Z. Ling, X. Fu, B. Liu, K. Ren, and W. Zhao, "Blind recognition of touched keys on mobile devices," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014, pp. 1403–1414.

[29] D. Shukla, R. Kumar, A. Serwadda, and V. V. Phoha, "Beware, your hands reveal your secrets!" in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014, pp. 904–917.

[30] B. Nassi, Y. Pirutin, R. Swisa, A. Shamir, Y. Elovici,

and B. Zadov, "Lamphone: Passive sound recovery from a desk lamp's light bulb vibrations," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 4401–4417.

[31] 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.

[32] B. Nassi, R. Swissa, J. Shams, B. Zadov, and Y. Elovici, "The little seal bug: Optical sound recovery from lightweight reflective objects," in *2023 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2023, pp. 298–310.

[33] 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.

[34] S. Sami, Y. Dai, S. R. X. Tan, N. Roy, and J. Han, "Spying with your robot vacuum cleaner: Eavesdropping via lidar sensors," in *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, ser. SenSys '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 354–367. [Online]. Available: https://doi.org/10.1145/3384419.3430781

[35] 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.

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

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] B. Nassi, Y. Pirutin, T. Galor, Y. Elovici, and B. Zadov, "Glowworm attack: Optical tempest sound recovery via a device's power indicator led," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 1900–1914.

[42] 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.

[43] 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.

[44] 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

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

[46] "Sunba-ceiling-outdoor-security-infrared," https://www.amazon.com/SUNBA-Ceiling-Outdoor-Security-Infrared/dp/B09Z6R48SH/r.

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

[48] 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.

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

[50] "Rec - pro video camera," https://apps.apple.com/us/app/rec-pro-video-camera/id1175490870.

Appendix A.
Meta-Review

The following meta-review was prepared by the program committee for the 2024 IEEE Symposium on Security and Privacy (S&P) as part of the review process as detailed in the call for papers.

A.1. Summary

This paper proposes a novel type of side channel attacks in which a (possibly remote attacker) analyses the video of a power/status LED to learn about the timings of cryptographic implementations running on the device. This is possible either because power consumption directly affects the brightness of the power LED, or, because the color of the LED is used to explicitly mark the execution of some sensitive operation. The rolling shutter of a camera is used is used to achieve a sufficient sampling rate even with COTS cameras. The paper evaluates the impact of brightness and distance on the attack and shows two concrete examples of key recovery via known timing vulnerabilities/attacks.

A.2. Scientific Contributions

- Provides a Valuable Step Forward in an Established Field

A.3. Reasons for Acceptance

1) This paper shows that the optical side channel leak provided by a power/status LED can be exploited using only a COTS video camera.
2) In the best case, attacks are possible from a large distance or even remotely, if an attacker compromises a security camera.
3) The paper solves many practical challenges to demonstrate the feasibility of the attacks in various scenarios.
4) Overall, the paper raises awareness about the risks of optical indicators leaking side channel information to video cameras controlled by an attacker.

A.4. Noteworthy Concerns

1) While the optical side channel attacks with COTS cameras presented in this paper are novel, some of the underlying techniques have been studied in previous work. The optical side channel caused by power/status LEDs was shown and exploited using a photodiode in [10, 11]. Using a rolling shutter to increase the sample rate of cameras was explored, for different goals, in [31, 33, 35]. As the paper focuses on demonstrating a novel method to measure optical side channel leakage, the attacks are based on existing timing vulnerabilities [17, 19]. These works are discussed in the paper.
2) The threat model has very strong requirements (e.g., presence of a vulnerable LED or connection to a vulnerable device, presence of a timing side channel vulnerability, proximity, darkness, long acquisition time). These requirements are described in the paper for each attack, and acknowledged in the limitations on applicability.