Charger-Surfing: Exploiting a Power Line Side-Channel for Smartphone Information Leakage

Patrick Cronin
University of Delaware
ptrick@udel.edu

Xing Gao
University of Delaware
xgao@udel.edu

Chengmo Yang
University of Delaware
chengmo@udel.edu

Haining Wang
Virginia Tech
hnw@vt.edu

Abstract

Touchscreen-based mobile devices such as smartphones and tablets are used daily by billions of people for productivity and entertainment. This paper uncovers a new security threat posed by a side-channel leakage through the power line, called Charger-Surfing, which targets these touchscreen devices. We reveal that while a smartphone is charging, its power trace, which can be measured via the USB charging cable, leaks information about the dynamic content on its screen. This information can be utilized to determine the location on the touchscreen where an animation is played by the mobile OS to indicate, for instance, that a button press has been registered. We develop a portable, low cost power trace collection system for the side-channel construction. This leakage channel is thoroughly evaluated on various smartphones running Android or iOS, equipped with the two most commonly used screen technologies (LCD and OLED). We validate the effectiveness of Charger-Surfing by conducting a case study on a passcode unlock screen. Our experiments show that an adversary can exploit Charger-Surfing across a wide range of smartphone models to achieve an average accuracy of 98.7% for single button inference, and an average of 95.1% or 92.8% accuracy on the first attempt when cracking a victim's 4-digit or 6-digit passcode, respectively. The inference accuracy increases to 99.3% (4-digit) or 96.9% (6-digit) within five trials. We further demonstrate the robustness of Charger-Surfing in realistic settings and discuss countermeasures against it.

1 Introduction

Touchscreen devices such as smartphones and tablets have become a daily tool for a variety of business and entertainment activities, including mailing, banking, browsing, gaming, and photography. While these devices have ushered in an era of great convenience, their rich functionality has lead to ever-increasing usage, draining batteries faster, and necessitating that users seek out areas to charge their smartphones. One study suggests that city dwellers charge their phones multiple times per day [6]. To allow users to conveniently charge their devices, facilities such as USB power lines and charging stations have been widely deployed in public areas, including airports [10], parks [2, 11], hotels [3], and hospitals [1]. The market for shareable power banks is also thriving [7], allowing users to simply scan a QR code to rent a public power bank and charge their devices.

Despite their convenience, USB charging interfaces and stations also introduce a number of security threats, as the USB interface in a public area is not under the user's control [8]. A typical USB interface is composed of one or more (depending on the protocol) differential data lines for data transmission and a 5V and ground line for delivering power. Previously it has been demonstrated that the data transmitted over the data line can be sniffed [45] or monitored through the crosstalk leakage on the power line [54]. Adversaries can also extract power consumption information from the power line to infer coarse-grained information, such as internet browsing history [25] or password length [65]. These disclosed security threats, however, do not stop users from heavily utilizing USB charging facilities in public areas, since charging usually involves no data transfer over the USB data line.

In this work, we reveal that USB charging in public areas can pose far more serious threats than previously believed. We show, for the first time, that the signals on the power line form a side channel and leak far more fine-grained information than previously believed. Specifically, the power consumption information is highly correlated with the activities on the touchscreen. Leveraging this side channel, built on the dynamic power signals, adversaries can precisely identify the location of virtual button presses on the touchscreen, with which they can steal extremely sensitive data such as a user's passcode. We call this security threat *Charger-Surfing*. We conduct a series of experiments to demonstrate the existence of fine-grained information leakage tied to smartphone touchscreen activity. For the construction of the Charger-Surfing channel, we develop a wireless, low cost, and portable power trace capture system using commercial-off-the-shelf (COTS) hardware. To further demonstrate that Charger-Surfing is a real threat, we perform a case study on a numeric passcode unlock screen and show that Charger-Surfing is able to extract a

passcode on both Android and iOS devices by leveraging signal processing and neural network techniques. We thoroughly assess this security threat on different types of smartphones, multiple phones of the same model, and across different users. Our results show that Charger-Surfing can achieve an average accuracy of 98.7% for single button inference on all the tested smartphones. For an *unknown* user[1], Charger-Surfing has, on average, a 95.1% or 92.8% chance to accurately crack a 4-digit or 6-digit passcode on its first attempt, respectively, and a 99.3% (4-digit) or 96.9% (6-digit) success rate within five trials.

In a nutshell, this is the first work that demonstrates fine-grained information leakage over the power line of the USB charging cable regarding the content of the touchscreen. More importantly, our studies show that the effectiveness of Charger-Surfing is victim-independent, meaning that adversaries can train the neural network using touchscreen data on their own smartphones with different configurations without any prior knowledge of a victim. The major contributions of this work include:

- A comprehensive study on the dynamic power usage of the touchscreen to demonstrate the location, causes, and granularity of information leakage over the USB power line. To the best of our knowledge, this is the first work to explore the classification of dynamic screen animations and induced information leakage.
- A new security threat, Charger-Surfing, which exploits a side channel through the USB power line to infer user interactions with the content on the touchscreen. The techniques used by Charger-Surfing for signal processing and model learning are given.
- A portable microcontroller-based power trace capture system using COTS hardware, which demonstrates the feasibility of exploiting the disclosed leakage channel at a low cost.
- A thorough evaluation on multiple smartphones, showing high accuracy in inferring a victim's private information, such as their passcode, without any prior knowledge of the victim, and that this leakage vulnerability is not tied to a specific smartphone or mobile OS.

The rest of this paper is organized as follows. Section 2 presents our threat model and a brief primer on USB charging, touchscreen technology, and touchscreen animations. The existence of fine-grained information leakage over the USB power line is demonstrated in Section 3. The security threat posed by Charger-Surfing is detailed in Section 4, followed by an in-depth case study in Section 5. Section 6 discusses the attack practicality of Charger-Surfing. Section 7 describes countermeasures against Charger-Surfing. Section 8 surveys related work, and finally, Section 9 concludes the paper.

[1]To show the effectiveness of Charger-Surfing, the model of a target device is trained with the data created by an adversary and tested with victim users whose data were not used to train the model.

2 Threat Model and Background

This section first presents the threat model, and then discusses the various components of a smart device involved in the new side channel, including (1) USB charging, (2) touchscreen technology, focusing on the dynamic power consumed when displaying different colors, and (3) the dynamic content of the touchscreen that could be potentially leaked.

2.1 Threat Model

The objective of this work is to highlight the vulnerabilities of the power line side-channel in smartphones which, if exploited, can lead to serious information leakage. We consider a realistic scenario in public places, where users charge their smartphones with a USB charger that is not owned/controlled by themselves. The USB charger could be a charging station in a public area, such as airports (Figure 1a), or simply an interface where users bring their own USB cables (Figure 1b). It could also be a shareable power bank rented from a third-party (Figure 1c), or the USB outlets provided in a hotel (Figure 1d). The USB charger provides the standard functionality (i.e., charging) and looks ordinary.

However, since these chargers are controlled by third-parties, the power consumption of the connected device could be monitored by a device hidden inside the packaging or behind the charging interface. The voltage monitor would not cause any adverse impact to the charging speed, and would thus be quite stealthy. With a low power microcontroller concealed inside the packaging, power traces can be recorded, or streamed wirelessly, for analysis.

Finally, we assume that adversaries have *no* prior knowledge of a specific victim, and have no need or have never had the chance to collect the power trace of the victim's smartphone. However, we assume that adversaries can easily profile the power dynamics of most popular smartphone models beforehand, enabling them to attack a wide range of smartphone users.

Security Threats Posed by Leakage. We observe that the dynamic power trace of a smartphone is highly correlated to the animation played on the touchscreen. Unfortunately, leaking the animations played on the touchscreen could cause severe security threats. The owner of such a specialized "surfing" charger can steal a victim's private data entered through the touchscreen, such as passcode, credit card number, and banking information. To expose such threats, we demonstrate Charger-Surfing's capability in inferring a numeric passcode.

While there are a myriad of potential biometric lock mechanisms available (fingerprints, faceID, etc.), many of these can be deceived [5, 9] and require a backup PIN (personal identification number) code if they are unavailable (gloves, sweat, etc.). Other authentication mechanisms such as Android's

pattern-based lock are not available on all phones and have been shown to be less secure than a PIN code [12]. Thus, we focus on the passcode-based lock as it is the most widely used primary or secondary authentication mechanism to unlock touchscreen devices, and it acts as one of the only barriers to gain complete control of a smartphone.

A passcode is extremely valuable to a dedicated adversary. When a victim can be easily identified (e.g., using a USB port at a hotel room), knowledge of the passcode would be sufficient for an adversary with physical access (e.g., *evil maid attack* [4]) to the victim's smartphone to steal private information or even reset other online passwords (e.g., Apple ID and iCloud passwords). Even for an adversary without physical access (e.g., a shareable power bank), a compromised passcode could still lead to severe consequences, as users tend to reuse their passcodes (recent studies show that each passcode is reused around 5 times [31]) and a smartphone's passcode may be reused as the PIN code of a credit/debit card or online payment system (e.g., Apple Pay or Alipay). Overall, there are many possible real scenarios, where this type of information would be very useful to law enforcement or an adversary for espionage, fraud, identity theft, etc.

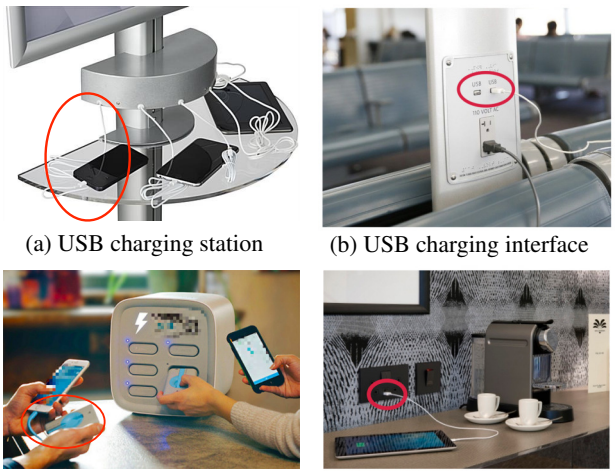
2.2 USB Charging

USB has become a standard interface for charging portable devices such as smartphones, while enabling serial communications at the same time. Standard USB plugs contain four pins and a shield: one pin delivers +5VDC [51], one pin connects to the shield forming the ground, while the other two pins are used for differential data transmission and carry negligible current when charging the battery. Newer USB protocols include more differential data pairs, but leave the +5VDC and ground pins the same. When charging a device, its battery enters the charging state, and the device's power is supplied not from the battery but from the power source connected by the USB power line.

2.3 LCD/OLED Touchscreen Technology

The two major touchscreen technologies are Liquid Crystal Display (LCD) and Organic Light Emitting Diodes (OLED). Both technologies have many improvements or extensions, such as Active-Matrix Organic Light-Emitting Diode (AMOLED), Super AMOLED, and In-Plane Switching (IPS) LCD. The power consumption profile of these touchscreen technologies is reviewed below [21].

LCD has three major components, a backlight that is always on, vertically polarized filters, and liquid crystals. The liquid crystals are charged to different voltages to display different colors. Specifically, to display a black pixel, the crystals are charged with the highest voltage. This voltage aligns the crystals horizontally, allowing only horizontally polarized light through. As the filter layer is vertically polarized, no light can shine through and a black pixel is produced. To display a white pixel, the crystal layer voltage is relaxed, aligning



(a) USB charging station (b) USB charging interface

(c) Shareable Power Bank (d) USB charging in hotel

Figure 1: USB charging in public or shareable environments.

it vertically, allowing light to pass through the filter. **OLED** displays utilize organic molecules to produce holes and electrons to create light in an emissive layer. Individual OLEDs are used to produce each pixel. To display a black pixel, the OLED must enter a low power state, while displaying a white pixel requires the OLED to enter a high power state.

As LCDs and OLEDs use *dissimilar* mechanisms to produce an image on a screen, they generate vastly different power traces to produce the same image. Specifically, to create an animation of a white dot, most pixels will be black. The black LCD pixels will be in a high power state, and the pixels that make up the white dot in a low power state. OLEDs, on the other hand, will have their black pixels in a low power state, and their white pixels in a high power state. Thus, if it were possible to observe the voltages applied to the individual pixels, the two screen technologies should have inverse values when they are utilized to display an identical image.

2.4 Animations on the Touchscreen

Smartphones with touchscreen technology always provide graphical interfaces (e.g., the lock screen, the telephone dial pad, and the text entry keyboard in applications) for users to input data, and also use real-time animations to inform the users that their inputs have been registered. Most of these animations occur on a static screen (i.e., no other animation is playing) and always at the same location on the screen (i.e., the digit/letter does not move around). As reviewed before, displaying lighter or darker pixels consumes different amounts of power in LCD and OLED technologies. Furthermore, LCD and OLED screens refresh from left to right, row by row, leading to the potential that the dynamic power consumption, which can be measured through the USB charging cable, may leak the location on the touchscreen where a virtual button is pressed.

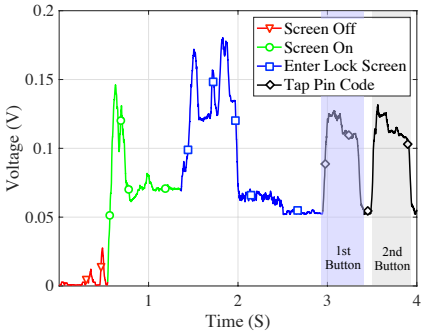


Figure 2: Power leakage on the USB power line when charging a Motorola G4. Sampling rate is 125 KHz. The signal is filtered with a moving mean filter to increase clarity.

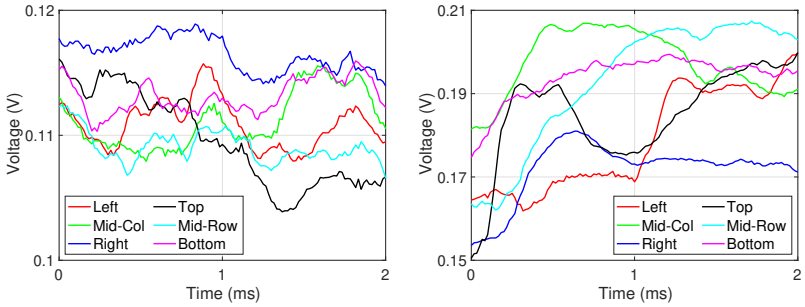


Figure 3: Averaged voltage readings for (a) Motorola G4 with LCD screen and (b) Samsung Galaxy Nexus with AMOLED screen, when displaying flickering white bars on the top, middle, and bottom rows, as well as left, middle, and right columns, of a black screen.

3 Power Line Leakage Exploration

Smartphones are sophisticated computing platforms with a complex multi-core System-on-a-Chip (SoC) handling various device drivers for touchscreens, cameras, sensors, etc. Previous research has shown that the display (i.e., touchscreen) and CPU/GPU are among the top contributors to the overall power consumption in a smartphone [20]. While previous work has shown that the power consumption of a smartphone leaks information regarding the activities on the touchscreen [64, 65], such information leakage is of coarse granularity (e.g., internet browsing history [65] or password length [64]). In comparison, the goal of this work is to demonstrate fine-grained information leakage, specifically, the ability to identify the exact locations of button presses and extract a user's input (e.g., a passcode) with dynamic power traces.

To examine the power leakage, we conduct a series of experiments utilizing a Motorola G4 connected to a USB charging cable in which the ground cable has been cut and spliced with a small resistor. An oscilloscope is used to monitor the voltage across this resistor and thereby the current utilization of the device. This section presents our experimental findings, highlights the leakage patterns, and further shows that the state of the smartphone's battery will not cause any attenuation effects on the side channel.

3.1 Button Press Detection

To explore the potential for identifying button presses, our first study observes the signal on the USB cable while charging a smartphone, utilizing the aforementioned oscilloscope and charging cable setup. The dynamic power signal is highly correlated with device activity, as illustrated in Figure 2. When the smartphone is asleep, there is a steady current utilization with minimal noise. Once the phone is perturbed from the sleep state, there is an immediate increase in its current utilization. When the phone enters the lock screen, the signal shows large spikes at different intervals. Finally, when the user starts to tap the screen and enter a passcode, the signal exhibits a clear rise and fall upon each button press.

This experiment not only demonstrates the information leakage on the power line, but furthermore illustrates two important properties underpinning our following studies: (1) from the signal measured on the USB power line, one can clearly detect the powering-on of the screen and the exact starting point of the button-press sequence; (2) in the lock screen mode, each button press made by a user is clearly observable and separable.

3.2 Button Press Location Identification

The power usage in Figure 2 shows a significant elevation when a button is pressed. This elevated usage is caused by the activities of the mobile OS. Specifically, once the mobile OS has captured an input action from the user, it provides visual feedback by rendering and drawing an animation on the screen, causing pixels to rapidly change colors and inducing two significantly different voltage states. On the lock screen, the animation for each button press is similar, albeit in a different location. These similar animations cause the power leakage to exhibit similar signals for different buttons, as the blue and grey areas in Figure 2 depict.

The unique contribution of this work is to discriminate the "similar-looking" signals and extract the *location* of the animation via power leakage. To examine this potential, we have designed a custom Android application running on two smartphones with different screen technologies: the Motorola G4 with an LCD screen, and the Samsung Galaxy Nexus with an AMOLED screen. The application divides the screen into six portions (i.e., top, middle, and bottom rows, as well as left, middle, and right columns) and displays, on a black background, flickering white bars that fill each portion of the screen in their respective tests. To mimic the way the Android OS renders user interface elements and the lock screen, we set the `hardwareAccelerated` developer flag to ensure that the GPU is involved in image rendering.

The gathered signal exhibits a steady 60Hz signal that denotes the beginning and end of a refresh cycle[2]. We isolate

[2]The screen constantly refreshes all pixels with a specific rate (typically

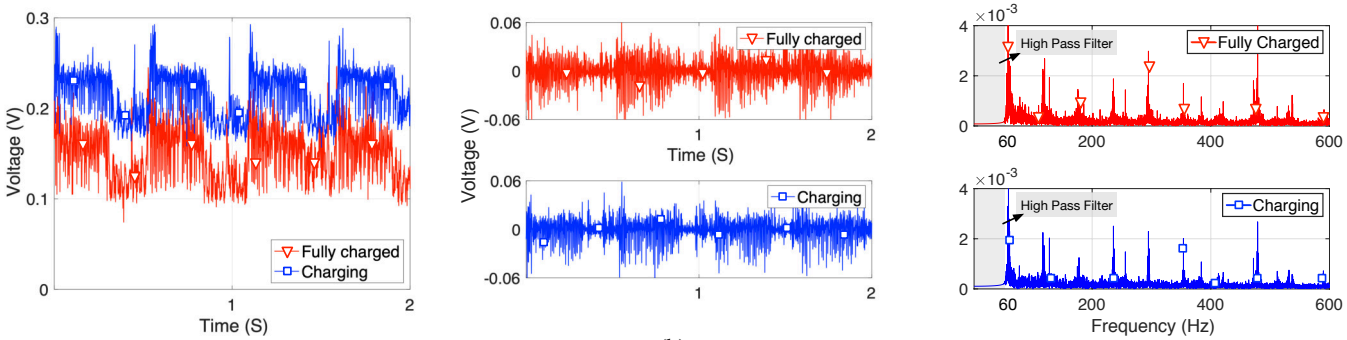


Figure 4: Comparison of voltage readings when pressing buttons on the lock screen of a Motorola G4 in two cases: fully charged vs charging. Sampling rate is 125 KHz. (a) depicts the raw unfiltered signal. (b) utilizes a high pass filter with a cutoff frequency of 60 Hz to remove the offset. (c) presents the Fourier transform of the filtered signal, demonstrating that the charging status of the phone does not affect the signal integrity.

the 60Hz signal within the sample stream and average all of the frames to reduce noise for better visual effects. The results are presented in Figure 3, which zooms in on a 2ms portion of the signals to better display the subtle differences. As can be seen, the voltage readings show that in both LCD and OLED technologies, there is an appreciable difference in the power usage of displaying the same image on different portions of the screen. These experiments demonstrate the great potential for inferring the *location* of the animation played on the screen when a user presses a virtual button, by exploiting the power leakage on the USB power line.

3.3 Impact of Battery Charging

One important question is whether the state of the smartphone's battery will cause any attenuation effects on the power side channel. This is critical as the smartphones charged at public USB charging facilities will likely have arbitrary battery levels. Once plugged into a charger, the smartphone draws its power from the charger and uses any excess power to charge its battery. Not only will a charging battery lead to a higher power draw than a fully charged battery, but the battery charging circuitry might attenuate the power leakage information, since high frequency signals contained in current spikes might be filtered by the reactive components of the battery charging controller.

To study the difference between a fully charged phone and a charging phone, we collect the power traces under the same workload, i.e., when entering a single digit on the virtual keypad repeatedly. The power traces are presented in Figure 4a. The figure shows a positive offset for the "charging" case, demonstrating that a larger base amount of current is being drawn by the phone to perform its tasks and additionally charge the battery. However, upon applying a high-pass filter to remove all frequencies under 60Hz that correspond to the DC offset in the signal, the filtered signals of the two phones match each other quite well, as shown in Figure 4b. We also conduct a Fourier transform on both signals, and display the resulting frequency spectrum in Figure 4c. In the figure, the high-band frequency signals still exist in both cases, preserving the high speed dynamic fluctuations attributed to the user touching the screen. Although the charging battery illustrates a slightly smoothed frequency signal, there is no obvious visual difference in the frequency spectrum between a charging phone and a fully charged phone.

60Hz), in a manner from left to right, and from top to bottom. This phenomenon can be observed with a slow motion camera, such as the one on an iPhone, which films at 240 frames per second.

4 Sensitive Information Inference

This section presents the method with which Charger-Surfing exploits the fine-grained power line leakage described above to infer button presses made by a smartphone user.

Figure 5 presents the working mechanism of Charger-Surfing. An adversary first acquires raw signals from a "surfing" charger with a hidden voltage monitor (provided in step ❶). The raw signal is searched to detect a button sequence (step ❷), which is further isolated to individual buttons (step ❸). Next, a neural network processes the signal to determines the target device model (step ❹). This information is used to select the exact model for button identification from a set of pre-trained neural networks. The button press signal is preprocessed (step ❺) for the phone model specific neural network, which finally infers the virtual buttons pressed by the user on the touchscreen (step ❻). The rest of this section details the techniques used in each step of Charger-Surfing.

4.1 Raw Signal Acquisition

The prerequisite for sensitive information inference is to covertly and comprehensively capture the power trace of the user's smartphone without losing any useful information. In Charger-Surfing, this is performed at step ❶, as shown in Figure 5, via a hidden voltage monitor that is attached to the charger without a user's knowledge.

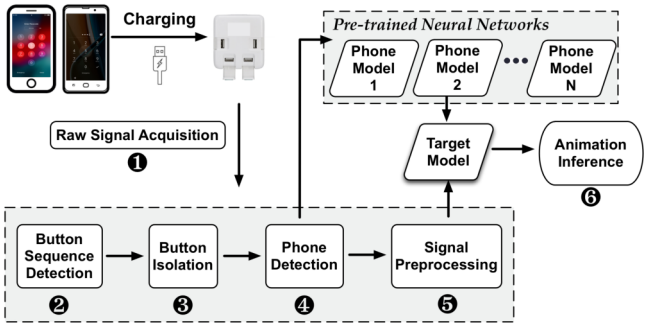


Figure 5: Overview of Charger-Surfing's working flow.

The voltage monitor should be able to collect the raw signal of the charging device at a **sampling frequency** that is carefully determined. Utilizing a very high frequency will result in unnecessarily large and cumbersome data, while sampling too slowly will miss key information. There are two factors that affect the sampling frequency: the refresh cycle of the screen and the resolution of the screen. As mentioned in Section 3.2, screens typically refresh pixel by pixel, from left to right and from top to bottom. To observe both the row and column portion of an animation, it is preferable to sample at a rate that is slightly greater (or less) than the per row update speed, so that (1) the power utilization can be monitored on a per row basis, and (2) samples can be taken in different columns as the refresh moves down the screen. Most of today's flagship smartphones use a screen resolution between 1920×1080 and 2960×1440 and have a refresh rate of 60 Hz. A single sample per row would require a sample rate in the range of 115–178 KHz. Our design uses a sample rate at 125 KHz, which takes one sample per every 0.9 – 1.4 rows on many flagship smartphones. This rate ensures that consecutive samples are not taken on the same vertical line, thus providing more useful location information.

4.2 Button Sequence Detection

Step ❷ of Charger-Surfing processes the captured power trace and isolates the portions of the signal corresponding to the sequence of button presses.

When the user presses a virtual button on the touchscreen, the mobile OS determines the location of the input and acknowledges the user by lighting up the button (or playing an animation around it). With a text or numeric entry, it also displays the corresponding letter or number on the screen.

Each of these activities increases the power consumption, collectively generating a visible spike in the captured raw power utilization signal, as shown in Figure 2. To detect these signals, Charger-Surfing utilizes a moving mean filter and a level detector. The filter removes noise from signals, allowing the level detector to isolate portions of the signal belonging to a button press sequence once the level is above an empirically determined threshold.

4.3 Individual Button Isolation

Upon detecting a sequence of button presses, Charger-Surfing moves to step ❸ which detects and isolates each individual button press. Since users press buttons at different rates, inferring individual button signals is much easier and more practical than blindly classifying the entire sequence with button presses possibly occurring at any arbitrary speed.

The process of detecting individual presses also utilizes a combination of a moving average filter and a level detector. When passed through a moving average filter, the button sequence displays spikes, each of which corresponds to the beginning of a button press, as shown in Figure 6.

Depending on the button press rate, the raw power signal (e.g., the top picture in Figure 6) may show either a single and isolated press, or multiple overlapping button presses. In the latter case, it is important to select the **signal portion** containing the most distinctive information. The lower picture of Figure 6 shows the pattern of a single button press, wherein the biggest changes occur at the beginning of the signal. This trend is consistent with the typical behavior of the screen, which is usually static but comes alive as soon as a button is pressed. Accordingly, for overlapping button presses, Charger-Surfing discards the end of the signal and keeps the beginning, which is the most important, distinctive, and potentially identifiable portion.

4.4 Phone Detection

In the envisioned threat model, adversaries can profile the power charging dynamics of most popular smartphone models beforehand, and pre-train a neural network model for each of these popular phones. A victim's signal collected over the USB power line can be fed into the pre-trained model, once the phone type is determined.

While the steps ❶–❸ performed up to this point are generally applicable to all smartphones, step ❹ of Charger-Surfing focuses on detecting the phone type. This task is much easier than classifying individual button presses as the screen technology, the screen resolution, and different components within the phone (CPU, GPU, screen driver, etc.) lead to vastly different power trace patterns, as demonstrated in Figure 3. To accomplish this identification task, we utilize a neural network that is trained with the isolated button press signals. The raw signal is passed through a high-pass filter to preserve the high-frequency components, which are highly correlated to the phone model, while removing the less informative DC offsets that can be a result of brightness changes, charging/charged, or different charging rates.

As the victim's phone model may not belong to the set that the attacker utilized to train Charger-Surfing, the system further examines the confidence values of each output class when inferring the phone model. If the confidence values are all low, it will not pass the samples to the phone-specific neural networks for classification.

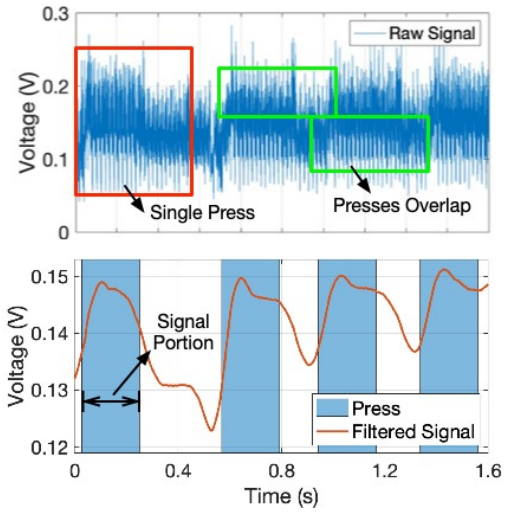


Figure 6: The top displays the raw signal of multiple overlapping button presses. The bottom demonstrates how peak detection can be utilized to determine non-overlapping portions of individual button presses. The signal is collected from Motorola G4 and filtered for clarity.

4.5 Signal Preprocessing

After determining the phone model, Charger-Surfing then scales and standardizes the power signal in step ❺ following the characteristics of the specific phone model. The signals gathered from the USB power line are commonly between 0 and 100 mV. After passing through the high-pass filter, the signal is mostly distributed between -50 mV and 50 mV. We preprocess the data with a scaler designed for the target phone model, which is created by pre-training with a few samples from the adversary's own device. The resultant signal's range is between -1 and 1, which typically leads to the best inference results for most neural networks.

4.6 Animation Inference

In the final step (i.e., step ❻ in Figure 6), the preprocessed power signal is sent to a neural network trained for that specific type of device, to reconstruct the multi-press sequence that the victim types into the device.

As the collected signal is a one-dimensional time series of voltage measurements, Charger-Surfing utilizes a one-dimensional convolutional neural network (CNN). The network includes a repeated series of convolutional and max-pooling layers, followed by a softmax regression layer, which classifies the input signal into one of the possible buttons and provides a confidence value associated with each class.

Why Utilize a CNN? CNNs are known for their high accuracy when processing data with spatial correlation and classifying time series data [36]. Furthermore, as discussed in Section 4.1, Charger-Surfing uses a single sampling rate for all the phones and the sampling rate (125KHz) is chosen to modulate around the screen rather than continually sampling the same pixels. This implies that for phones with different screen resolutions, features of button presses appear at different locations of the power signal. CNNs are well suited to recognize features that can be found in any area of a signal.

Model Classifier Configuration. An important consideration of any CNN is the size of the convolutional kernels. Small kernels may not be able to recognize features that manifest themselves over a large portion of the input signal, while large kernels may be too coarse, missing the fine details and features of an input signal.

The ideal size of the convolutional kernels depends on the size of the features in the power trace, which in turn depends on the sampling rate, screen layout, and *size of the animation to be detected*. If one desires to classify individual keys on the device text entry keyboard, for example, it would be necessary to calculate the size of the key press animation with respect to the screen size and modify the kernel size accordingly. This allows the first layer of the network to capture features that are large enough to identify a button press, while not being so large as to oversimplify or miss a feature, and not being so small as to only capture noise. Furthermore, our CNN design adopts a typical architecture consisting of sets of a convolutional layer followed by a max-pooling layer, which potentially increases the receptive field[3] of the network. This allows the subsequent layers of the network to leverage the highlighted features and correlate their location across multiple frames of the signal when inferring the key press.

5 Case Study: Passcode Inference

To demonstrate that Charger-Surfing poses a genuine security threat, we conduct a case study of passcode inference. We divide our evaluation into two major sections. This section details the experimental evaluation for a broader range of devices, including data collection, single button inference, 4- and 6-digit passcode inference, and impact of sampling frequency upon inference accuracy, demonstrating the wide applicability of Charger-Surfing. Section 6 tightens the scope of our evaluations, focusing on a low-cost hardware implementation of the Charger-Surfing attack, its insensitivity to different smartphone configuration variables (wallpapers, brightness, vibration, charging status), and the transferability of the attack between different smartphones of the same model. In total, we gather data from 33 volunteers[4] and on 6 different devices. Our participants are about 30% female, including members of varied races, heights, and weights. The age of our participants ranges from 20 to 60 years old. This section utilizes the data of 15 volunteers and four devices, while Section 6 uses an additional set of 18 volunteers and two devices.

[3]The receptive field is the portion of the input signal affecting the current convolutional layer.

[4]The human-user-involved experiments have been filed and approved by the **Institutional Review Board (IRB)** to ensure participants are treated ethically.

5.1 Data Collection

To ensure that Charger-Surfing is not tied to a specific phone model, screen technology, or mobile OS, we collect data from a spectrum of smartphones running both iOS and Android OS, listed in Table 9 in Appendix A. For Android devices, the Galaxy Nexus represents smartphones with aging hardware, while the Motorola G4 provides an example of a more recent and advanced smartphone. A similar strategy is applied in selecting the iOS devices. The iPhone 6+ represents an aging but still widely used device, while the iPhone 8+ provides an example of a more recent smartphone that shares a large amount of hardware with the current iPhone SE 2nd generation released in 2020.

To assess the impact that individual users might have on the accuracy of Charger-Surfing, we collected input data from 15 volunteers who regularly use passcode based authentication in smartphones. Our participants have diverse backgrounds and are varied in height, weight, gender, race, and age. The goal is to demonstrate that Charger-Surfing is *victim-independent*, as the different users likely interact with the same smartphone differently (e.g., placing their finger on different areas of the button or holding their finger on the screen for different amounts of time), which could lead to variations in the duration of the animations played on the smartphones tested. Each user was tasked to input a pre-determined sequence of 200+ buttons on the numerical lock screen. The sequence was designed to gather a uniform distribution of button presses such that no button had a disproportionate amount of samples.

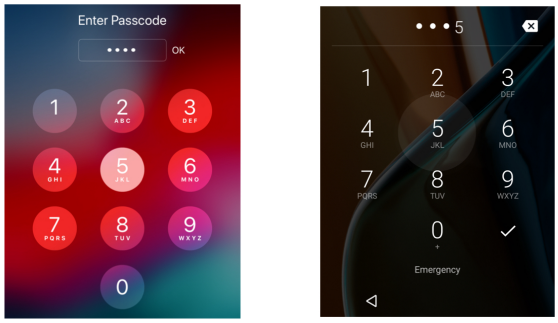
Our data collection utilizes a modified charging cable and a Tektronix MDO4024C oscilloscope. The charging cable is modified by cutting the ground wire and inserting a 0.3Ω resistor. The oscilloscope is used to measure the voltage drop across the resistor, providing a fine-grained and repeatable method of observation. It is configured to sample at a rate of 125,000 samples per second.

5.2 Classifier Configuration and Training

As discussed in Section 4.6, for the best performance, it is necessary to tune the kernel sizes of the CNN based on the screen layouts and animations that are being classified.

Figure 7 presents the typical lock screen layouts implemented by Android and iOS systems as well as the animations on the lock screen. As shown, the animations caused by a button press range from about $1/10$ of the vertical screen height on iPhones (button 5 in Figure 7a) to about $1/5$ of the vertical screen height on Android phones (button 5 in Figure 7b). With a sampling rate of 2,083 samples per frame[5], the most pertinent features for button identification are within 208 (iPhone) - 416 (Android) samples. Thus, when considering the receptive field of the network, we choose an initial kernel size of 50 for the iPhone network and 100 for the Android network. This sizing configuration ensures that we capture the smaller features of the signal in the initial layers of the network while still considering both the larger features of the signal in intermediate layers and the location on the screen across multiple frames of animation in the final layer. Detailed network configurations are listed in Tables 10 and 11 in Appendix A.

[5]The power trace signal is sampled at 125KHz, and the lock screen refreshes at a rate of 60Hz. Under this configuration, 2,083 samples are gathered within each refresh cycle. Each sample contains information about the content of the screen progressing vertically, as the screen refreshes from top to bottom.



(a) iPhone (b) Android

Figure 7: Passcode lock screen layout and animation.

Our threat model assumes that adversaries are unable to obtain the victim's data before training the system, and thus can only train the classifier using their own collected data. To emulate this scenario, we divide the users into two separate sets: one set for training (i.e., adversary) and the other set for testing (i.e., victim). To examine the robustness of the network to the composition of the training data, we randomly select five users to create the training set. The remaining 10 users form the testing set, ensuring that there is no overlap between the training and testing users. We train five neural networks for each device such that the ith ($1 \leq i \leq 5$) network is trained with the data from i different users. In testing, each network's performance is evaluated on the 10 testing users, and the average accuracy is reported.

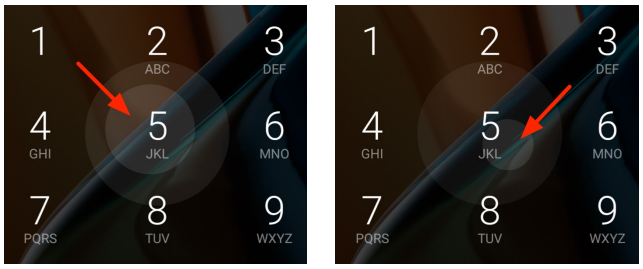
5.3 Phone Identification

Our experimental steps closely follow the process in Figure 5. After the signal is acquired, it is passed through button isolation (as described in Section 4.3). The next step is to correctly identify the target phone model so that the signal can be processed by the appropriate preprocessing system (Section 4.5) and classifier.

We train a primary neural network using high-pass filtered data from a subset of the collected users and test on the data from the remaining users. Our results show that the network can determine the correct phone model 100% of the time. This identification step is also applicable to phones that might run multiple OS versions. Different OS versions would be detected and classified at this step before being passed to the more specific secondary neural networks.

Table 1: Single Button Accuracy

# of Training Users	Phone			
	Motorola G4	Galaxy Nexus	iPhone 6+	iPhone 8+
1	82.0%	50.0%	23.8%	44.6%
2	90.0%	95.0%	93.3%	67.1%
3	99.6%	99.1%	96.9%	88.7%
4	99.7%	99.4%	98.5%	94.5%
5	99.9%	99.6%	99.5%	95.8%



(a) Press button on the left side. (b) Press button on the right side.

Figure 8: Android's animation on touching different parts of a button.

5.4 Single Button Inference

We first evaluate the accuracy for inferring a single button press, which is the most fundamental aspect of the system, as, without the ability to robustly classify a single button, it is impossible to accurately infer the entire passcode.

Table 1 lists the accuracy of a single button inference for each smartphone. When the training data was collected from only one user, we observe divergent accuracy results for different phones, ranging from 23.8% for iPhone 6+ to 82.0% for Motorola G4. Once we increase the training data size to two users, however, there is a significant accuracy improvement for single button inference: 67% for iPhone 8+ and more than 90% for all the other phones. The increasing accuracy trend is mainly attributed to the differences in user behavior when interacting with touchscreens, which can have direct effects on the power usage of the screen. More specifically, Android devices demonstrate *spatial* and *temporal* variations while iOS devices demonstrate *temporal* and *processing* variations. On the Android lock screen, the screen plays an animation that depends on where users place their finger. An example of this scenario is shown in Figure 8, where a user placing the finger on the left or right side of the button can create different animations. Furthermore, the longer the user holds their finger in this position, the larger the darker white circle grows. On iOS devices, when users press a button on the lock screen, no matter where exactly they press it, the entire button lights up completely and immediately. This animation does not end until the user removes their finger, imparting temporal variations to the recorded power trace. Furthermore, devices newer than the iPhone 6S (such as the tested iPhone 8+) make use of so-called "3D-Touch" to measure the force of the screen press. This extra processing and information further introduces subtle noise or processing variations into the measured signals.

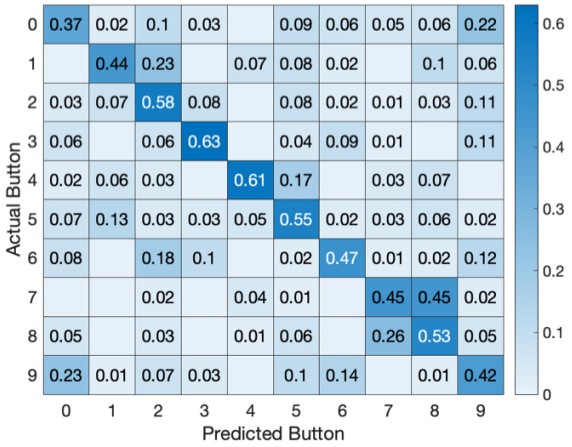


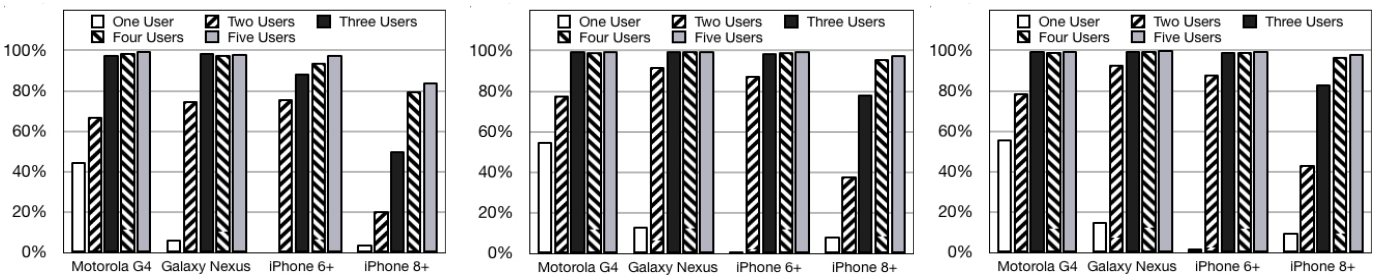
Figure 9: Breakdown of actual and predicted button classifications for the Galaxy Nexus when trained with one user's data. An entry on row i and column j corresponds to button i being classified as j.

The aforementioned user-oriented uncertainties and randomness can be dramatically mitigated by integrating more users into the training process. Once the neural network is presented with a robust dataset demonstrating diverse user behaviors, these abnormalities can be recognized and classified correctly. Table 1 confirms that by training on four users' data, Charger-Surfing can achieve more than 94% accuracy when classifying the single button presses of new users (i.e., the victims) for all devices. The average accuracy across all four test phones for single button inference further reaches 98.7% when there are five training users. By this point, the improvements demonstrate diminishing returns as more users are included. This indicates that our system only requires a few users' training data to achieve near optimal accuracy.

5.5 Misclassification Analysis

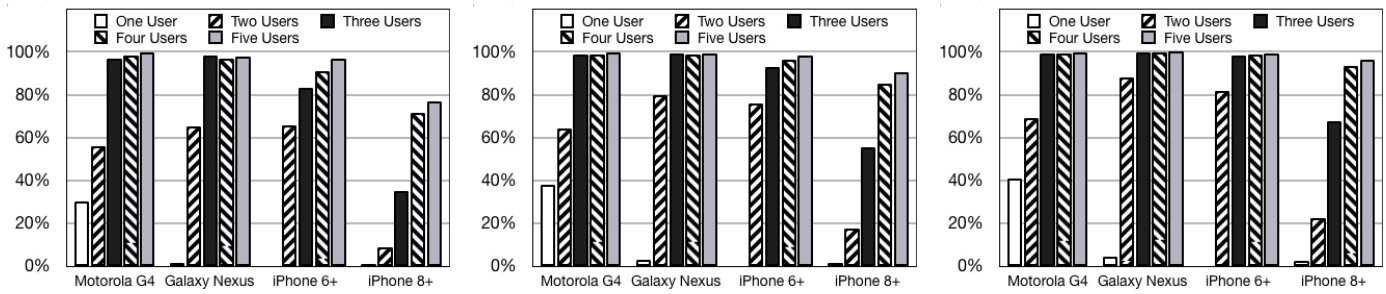
To further evaluate the effectiveness of Charger-Surfing, we examine how the neural networks perform when they guess incorrectly. Figure 9 presents the confusion matrix of the inference results of the Galaxy Nexus, when trained on only one user's data. The figure shows the actual pressed buttons as rows and predicted buttons as columns. An entry on row i and column j corresponds to button i being classified as j.

Figure 9 shows the highest prediction rate in the diagonal for all buttons except for button `7`, which can be classified as `7` or `8` with equal probability of 0.45. Five buttons (`0`, `1`, `6`, `7`, `9`) demonstrate performance lower than 50%, however, usually the incorrect inference is only off by a single row or column, indicating that the screen region it guessed is correct. Excellent examples of this phenomenon are the pairs (`0`,`9`)



(a) 1st Trial (b) 5th Trial (c) 10th Trial

Figure 10: Accuracy of 4-digit passcode inference.



(a) 1st Trial (b) 5th Trial (c) 10th Trial

Figure 11: Accuracy of 6-digit passcode inference.

and (7,8) that are frequently mis-predicted as one another.

In many buttons, the mis-predictions are not uniformly distributed but tend to cluster into one or two buttons, implying that a second or third guess would result in the correct prediction for these buttons. The results of the first three guesses of the system trained by only one user's data are shown in Table 2. The second guess achieves an average accuracy increase of 11.7%, and the third guess further increases accuracy by an average of 9.9%. This rapid accumulation trend will assist in the reducing the search space when classifying a user's passcode.

Table 2: Cumulative Accuracy of 3 Classification Attempts for Single User Trained Model

Attempts	Phone			
	Motorola G4	Galaxy Nexus	iPhone 6+	iPhone 8+
1	82.0%	50.0%	23.9%	44.6%
2	86.6%	63.0%	40.6%	57.3%
3	89.0%	72.0%	51.9%	65.5%

5.6 Passcode Inference

With ability to classify single button presses, it is possible to infer passcodes. Many Android and iOS smartphones allow up to ten passcode attempts before erasing the content of a device, thus we report the accuracy of Charger-Surfing in inferring 4-digit and 6-digit passcodes within 10 trials.

4-digit passcode: We select 1,000 random 4-digit combinations to test the classifier. To construct the candidates for a passcode guess, we examine the confidence vectors of each single button inference in the passcode. We rank these confidence vectors to produce the top candidates for each press and then construct combinations of the top candidates to produce guesses for the passcode. Figure 10 illustrates the accuracy for 4-digit passcode inference. We utilize the networks trained in Section 5.4, where each phone is trained on its own network with i ($1 \leq i \leq 5$) users. Figures 10 (a), (b), and (c) show the accuracy results after the first, fifth, and tenth trials, respectively.

In a brute force attack scenario, the success rate on the first trial is only 0.01%. By contrast, with only one user in the training set, Charger-Surfing achieves an average success rate of 13.9% on the first trial and a 20.8% success rate after the 10th trial. Clearly, there is a strong trend towards improved accuracy as the number of training users increases, showing that with more users, Charger-Surfing can develop a more general and accurate model that is robust against irregularities caused by user interactions with the smartphone. When two users are involved in training, the average success rate increases substantially, scoring 59.5% on the first trial and 75.8% by the tenth trial. This improvement trend continues but slows down as more users are included. Finally, it achieves an average success rate of 95.1% on the first trial and 99.5% on the tenth trials when trained with five users. The diminishing return indicates a strong convergence of Charger-Surfing's inference accuracy with only a few users in the training set.

6-digit passcode: We further evaluate the effectiveness of Charger-Surfing when cracking a longer, 6-digit passcode. Similarly to the 4-digit case, we select 1,000 random 6-digit

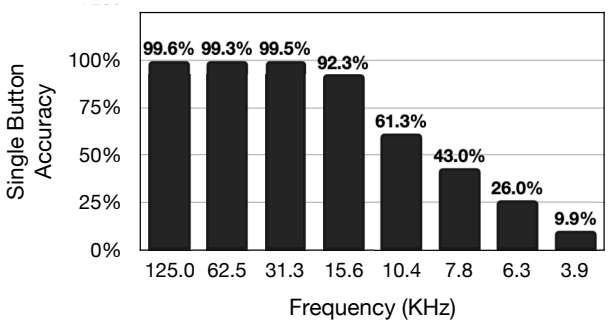


Figure 12: Impact of different sampling rates on single button accuracy, based on 3-user data of Motorola G4.

combinations and test them against our inference system. Figures 11 (a), (b), and (c) illustrate the accuracy after the first, fifth, and tenth trials, respectively. Although the search space for a 6-digit passcode is much larger (a 6-digit passcode has 1,000,000 combinations), Charger-Surfing demonstrates high success rates similar to those achieved when cracking a 4-digit passcode. When trained on five users, the success rate of the first trial is greater than 90% for all phones except the iPhone 8+, which has an accuracy of 77.0%. Even for iPhone 8+, the success rate then increases to 90.3% after the fifth trial; and the accuracy for all phones is more than 96% by the tenth trial. In comparison to a brute force approach that has a success rate of 0.001% within ten trials, Charger-Surfing is more than 96,000 times more effective.

5.7 Impact of Sampling Frequency

As mentioned in Section 4.1, Charger-Surfing utilizes a sampling rate of 125 KHz, which takes about 1 sample every 0.9–1.4 rows on many flagship smartphone screens. As sampling at a higher frequency requires more expensive and powerful equipment, we examine the impact of sampling at lower frequencies on single button inference accuracy. We downsample the raw signal to different frequencies, and preprocess the signal in the manner described in Section 4.5. The neural networks are resized and retrained to work with the data collected at reduced sampling rates.

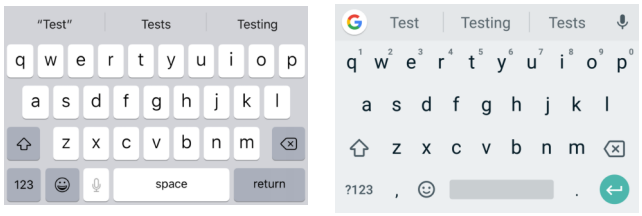
Figure 12 illustrates the accuracy of single button inference on a Motorola G4 using networks trained with three users. A decreasing trend in accuracy can be seen when lowering the sampling frequency. The drop is slow at first: when the sampling rate decreases to 31.3 KHz, the accuracy degrades from 99.6% to 99.5%, a drop of only 0.1%. When the sampling rate is reduced to 15.6 KHz, there is a larger drop in accuracy but it still remains above 90%. However, further decreases in the sampling rate leads to dramatic losses in accuracy.

To better understand the reason for the accuracy drop, we further examine the row and column accuracy degradation[6] as the sampling rate decreases. The results are listed in Table 3. It turns out that the column accuracy is the limiting factor. While the row accuracy remains above 94% even at 10.4KHz, the column accuracy degrades from 99.5% at 31.3KHz to 62.3% at 10.4KHz. Such a result is consistent with the screen refresh behavior: as the screen refreshes row by row and from left to right on each row, the row signal changes much slower than the column signal. Thus, a decreased sampling rate can still capture the row signal, but becomes incapable of fully capturing the column signal.

[6]Row (column) accuracy is defined as the percentage of classifications that fall within the correct row (column) (e.g., a '1' that is misclassified as a '2' is still in the correct row).

Table 3: Impact of sampling frequency on row, column, and overall classification accuracy, based on 3-user data of Motorola G4.

Frequency (KHz)	Accuracy		
	Row	Column	Overall
62.5	99.4%	99.4%	99.3%
31.3	99.8%	99.6%	99.5%
15.6	98.5%	92.4%	92.3%
10.4	94.1%	62.3%	61.3%
7.8	85.3%	46.9%	43.0%
6.3	59.5%	38.5%	26.0%
3.9	30.8%	33.4%	9.9%



(a) iOS Keyboard (b) Android Keyboard

Figure 13: Android and iOS keyboards. Each keyboard has a similar layout, with 4 rows of buttons. Each keyboard contains a maximum of 10 buttons per row (top row).

5.8 Detection Granularity Analysis

So far we have demonstrated that by monitoring the power usage of a charging smartphone, an adversary can extract the location of animations on the touch screen, compromising a user's passcode. Another particularly enticing target is the onscreen virtual keyboard. Each press of the keyboard provides feedback to the user by either displaying an enlarged version of the pressed character or by darkening the pressed key. Thus, an adversary with a voltage monitoring setup might attempt to infer a user's input by locating and classifying the animations of the onscreen keyboard. However, one important question remains: is Charger-Surfing able to achieve sufficient precision for classifying smaller animations on the screen?

To gain a better understanding of the achievable precision of Charger-Surfing, we examine the relationship among animation positioning, animation size, and inference accuracy at different sampling rates. Specifically, the results in Table 3 show that the column accuracy is the limiting factor in classification accuracy. Using the examples of the onscreen keyboard in Figure 13, we can see that both iOS and Android keyboards have a maximum of 10 columns (top row) that must be classified accurately. Table 3 shows that a sampling rate of 31.3 KHz is required to accurately classify 3 columns. Thus, to classify 10 columns, the sampling rate should be increased by at least 10/3 times to around 105 KHz.

While this sampling rate ensures that the signal contains enough information, it is equally important to tune the filter size in the neural network for identifying the patterns present in the data. As previously discussed in Section 5.2, the convolutional kernels must be sized such that they are smaller than the number of samples that encompass the animation. For example, in the iOS keyboard presented in Figure 13a, each key takes up about $1/17^{th}$ of the vertical space on the screen. Using the sampling rate determined above, of 105 KHz, 1,750 samples are taken during each screen refresh. Thus, each keypress animation can be recorded in about 103 samples. Leveraging our experience in training the CNN for passcode inference (a kernel size of 50 for 208 samples, as described in Section 5.2), a kernel size close to 25 should provide an adequate starting point for tuning the network to detect keyboard press animations.

6 Attack Practicality

The analysis on sampling rate shows the potential of developing a low-cost data acquisition system with cheap and compact commercial off-the-self (COTS) hardware, which can be easily integrated and hidden inside shared power banks or public USB charging facilities, making the Charger-Surfing attack more practical. In this section, we demonstrate the practicality of Charger-Surfing by (1) detailing a portable, low-cost power trace collection system, and (2) testing the system under different smartphone settings and across different devices of the same model.

6.1 A Portable Data Collection System

We design and develop a portable, low-cost microcontroller-based system for data acquisition, as shown in Figure 14. It consists of an Espressif ESP32 chip with a dual-core Tensilica Extensa LX6 processor, built-in WiFi, and Bluetooth radio. In the system, the microcontroller is connected to a 10-bit analog-to-digital converter (ADC) manufactured by Analog Devices (AD7813). One of the ESP32 cores is dedicated to gathering samples from the ADC, while the other core handles all WiFi communication and data storage needs. The sampling rate is configurable (up to 62.5KHz) and, as each sample is only 10 bits, the maximum data rate is quite low, at only 78.125KBps. The cost of the whole data collection system is less than $20.

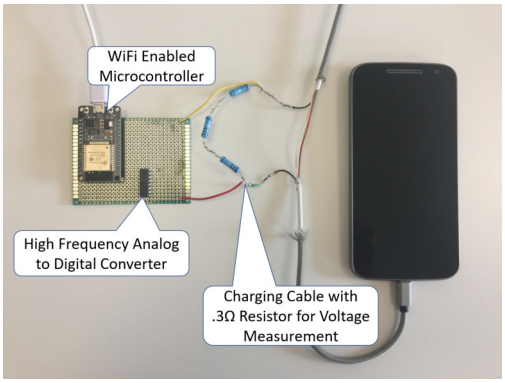


Figure 14: The portable, low-cost data collection setup. A WiFi enabled microcontroller can send acquired data to a custom webserver in real-time.

Table 4: Single Button and Passcode Inference Accuracy (5 training users / 15 testing users).

Attempt	Single Button Press
1	98.6%
2	99.4%
3	99.6%

Trial	Passcode 4-Digit	Passcode 6-Digit
1	94.9%	92.4%
5	97.4%	94.9%
10	97.5%	96.3%

A Motorola G4 is used to test the accuracy and effectiveness of this portable, low-cost data collection system. We set the sampling rate to 62.5KHz, and collect button press data from 20 different users. Based on the studies in Section 5, we randomly select five users to train the network and validate with the remaining 15 users. The results are shown in Table 4. We can see that even with a low-end (less than $20) data acquisition setup, an adversary can correctly identify single button presses with 98.6% accuracy on the first attempt: a drop of only 1.3% compared to a much more expensive, faster sampling and bulky setup (e.g., an oscilloscope). For cracking a 4-digit passcode, the system achieves an average accuracy of 94.9% in the first attempt and 97.4% by the fifth attempt. The results of cracking a 6-digit passcode are also promising: an average accuracy of 92.4% in the first attempt and 96.3% by the tenth attempt.

6.2 Testing of Varied Device Settings

In an attack scenario, it is unlikely that a victim's device is configured exactly like the attacker's training device. For example, it is likely that a victim has a different screen background, brightness setting, etc. To examine how these con-

Table 5: Single Button Inference Accuracy (5 training users / 1 testing user) with Varied Configurations.

Configuration	Static Wallpaper		Brightness			Charge	Haptics
	1	2	0%	50%	100%		
Accuracy (1st Attempt)	99.3%	98.0%	98.0%	97.3%	100%	99.2%	100%

Table 6: Cross-device training and testing configurations.

Training		**Testing**
Phone A		Phone B
Users: 1,2		Users: 3-12
Wallpaper: 1,2,3	⇒	Wallpaper: 4
100 Presses of each button		Balanced 200 button sequence
Total: 6,000 Presses		Total: 2,000 Presses

Table 7: iPhone 6+ cross device testing classification results. 2 training users on an iPhone 6+ and 10 testing users on a different iPhone 6+.

	Single Button
Attempt	Press
1	99.1%
2	99.4%
3	99.4%

	Passcode	
Trial	4-Digit	6-Digit
1	96.5%	94.6%
5	97.4%	95.6%
10	97.4%	96.2%

figuration variations may affect the accuracy of the attacker network, we test the network on a victim with different configurations. We gather data from a Motorola G4 in which we, one at a time, change the wallpaper (two different wallpapers), modify the brightness (0%, 50%, 100%), use an uncharged phone, and enable haptic feedback. We then test the data against the network trained with 5 users in Section 6.1. The results listed in Table 5 indicate that the configuration difference has very little impact upon the inference accuracy, which remains above 97% for single button inference in all cases. This demonstrates that Charger-Surfing is quite robust against device configuration changes.

6.3 Cross Device Testing

To further demonstrate that Charger-Surfing poses a real threat, we launch attacks under a more strict cross-device scenario wherein attackers can only train the classifiers on their own phone and then test them against a different phone (i.e., a victim's phone). Also, while attackers can collect data from multiple different wallpapers during training, they might not know the exact wallpaper used by the victim. This set of 'cross-device' experiments are conducted given two phone models, iPhone 6+ (iOS 12.4) and iPhone 8+ (iOS 13.4). Under each model, there are two phones (e.g., two iPhone 6+ phones) used separately for training and testing. For each training phone at the attacker side, we have two users who gather 100 presses for each button. We then train the model using three different wallpapers: black, white, and multi-colored. For each testing phone at the victim side, we gather 200 test presses from 10 users (different from the two users at the attacker side), with wallpapers that are not used in training. The exact training and testing configurations are listed in Table 6.

The obtained accuracy results of the two phone models, iPhone 6+ and iPhone 8+, are presented in Tables 7 and 8, respectively, demonstrating that both cross-device tests achieve

Table 8: iPhone 8+ cross device testing classification results. 2 training users on an iPhone 8+ and 10 testing users on a different iPhone 8+. High initial accuracy meant that subsequent attempts realized minimal improvement.

	Single Button
Attempt	Press
1	99.7%
2	99.8%
3	99.8%

	Passcode	
Trial	4-Digit	6-Digit
1	99.0%	98.6%
5	99.1%	98.6%
10	99.1%	98.7%

greater than 99% accuracy on the first attempt when classifying single buttons and greater than 94% accuracy when classifying 6-digit passcodes. Note that the accuracy results here are slightly higher than those in the oscilloscope-based experiments shown in Section 5. This slight difference could be caused by the different iOS versions (the oscilloscope experiments are performed on older iOS versions), or oscilloscope vs ADC quantization at low voltages.

Overall, this set of experiments clearly indicate that Charger-Surfing works well not only across different users but across different devices of the same model, posing a real security threat.

7 Countermeasures

Our experiments show that on different smartphones, Charger-Surfing is highly effective at locating the button presses on a touchscreen and inferring sensitive information such as a user's passcode. While it would be difficult to completely fix the leakage channel, which is related to USB charging and hardware, there exist some possible countermeasures.

The side channel exploited by Charger-Surfing leaks information about dynamic motion on the touchscreen. This attack is so effective as the layout of the lock screen is fixed: the buttons for a passcode are in the same positions every time the screen is activated. On the contrary, randomizing a number's position on the keypad for code entry would likely hamper Charger-Surfing's ability to detect a user's sensitive information. However, this position randomization may inconvenience users as it will take more time for them to locate each button. Furthermore, this approach scales poorly; randomizing a keyboard layout, for example, would be highly undesirable to users. Likewise, it is possible for smartphone vendors to remove button input animations, a change that would significantly reduce the information leakage in the power line, but provide minimal feedback to users as to whether they have correctly pressed the intended button. While both features are available in some customized versions of Android, they are not widely deployed in currently available devices.

At first glance, one likely solution is not to eliminate the leakage, but to drown it out via noise. One such option would be to utilize a moving background such as the readily available live/dynamic wallpapers on Android/iOS, which act similarly

to videos and constantly animate the screen. While this idea seems initially attractive, it has a few major drawbacks: 1) the live wallpaper only works on the lock or home screen and would not prevent similar attacks against onscreen keyboards in applications, and 2) the noise generated by this system is random and can be filtered out with sufficient samples. In a preliminary study of this defense technique, we built a neural network trained with 100 samples per button taken with two live wallpapers and tested on another live wallpaper. The network was able to realize greater than 98% single button accuracy, demonstrating that with sufficient samples of live wallpapers, Charger-Surfing can discern the true user input signal from the noise signal of the moving background.

To fully address the leakage channel exploited by Charger-Surfing, one solution is to eliminate the leakage channel by inserting a low pass filter in the charging circuitry of the device. This modification will remove the informative high frequency component from the signal. In a preliminary testing, we applied a low-pass filter with a cutoff of 60Hz to the collected iPhone 6+ cross-device data and the accuracy dropped to 10% (expected accuracy of random guessing). This result demonstrates that this approach can effectively mitigate the information leakage that Charger-Surfing relies upon.

Until an effective countermeasure is widely adopted, it is important for users to be increasingly aware of the security threats associated with USB charging. Users should avoid inputting a passcode or other sensitive information while charging their smartphones in public or shared environments.

8 Related Work

In this section, we briefly survey the research efforts that inspire our work and highlight the differences between our work and previous research. We mainly discuss research work in the following four areas:

Smartphone authentication. Smartphones are commonly equipped with two popular authentication methods: numeric-based passcodes or pattern-based passcodes. Both methods, however, are vulnerable to various types of attacks, including shoulder surfing [50], smudges [13], and keyloggers [19]. Previous work has demonstrated that sensory data (e.g, accelerometer, gyroscope, and orientation) can be used to extract a user's input on the touchscreen [43, 46, 63]. In addition to in-device sensors, attackers can also utilize acoustic signals to infer keystroke information on physical keyboards [17, 73]. Recently, Zhou et al. [72] proposed PatternListener to crack Android's pattern lock password through the acoustic signals gathered by a malicious application accessing the in-device microphone. Unlike these works, our work does not require malicious apps to be installed on the target smartphone.

Another type of keystroke inference on smartphone devices leverages video recording [66], where attackers use a camera to record finger behaviors [49, 62, 67] or the users' movements [55]. The reflections off of an eyeball, captured by special equipment, can also be exploited to leak device passwords [14, 15, 26]. Our work differs from these in that our approach does not require attackers to be in close physical proximity to the victim.

Other authentication methods utilize physiological biometrics (e.g., face [56]) and behavioral biometrics for authentication, including touch patterns [71], gait [40, 61], hand movements, and grasp features [38, 52]. However, these approaches can suffer from replay attacks and insufficient accuracy and do not satisfy industry requirements.

Power analysis. Extensive efforts have been devoted to analyzing the power consumption of smartphones [18, 39, 47, 48]. Carroll et al. [20] presented a detailed analysis showing that the touchscreen is one of the major consumers of power in a smartphone. Furthermore, many works [24, 29, 68] attempt to understand the energy consumed by the touchscreen.

The power consumption of a smartphone could be exploited as a side channel to extract information such as mobile application usage [25] or password length [64]. Yang et al. demonstrated that public USB charging stations allow attackers to identify the webpages being loaded when a smartphone is being charged [65]. Michalevsky et al. [42] demonstrated that power consumption could be used to infer the location of mobile devices. Spolaor et al. [53] showed that the USB charging cable can be used to build a covert channel on smartphones by controlling a CPU-intensive app over 20 minutes. To the best of our knowledge, we are the first to show that the power consumption of a smartphone can be used to infer animations on a touchscreen and steal sensitive data, such as a user's passcode.

Other side channel attacks. Chen [23] demonstrated that the shared *procfs* in the Linux system could be exploited to infer an Android device's activities and launch UI inference attacks. Without *procfs* (e.g., iOS devices), attackers can still infer sensitive information and private data by exploiting exposed APIs [69]. Genkin et al. [32] acquired secret-key information from electromagnetic signals by attaching a magnetic probe to a smartphone. Radiated RF signals can also be used to eavesdrop screen contents remotely [41]. Recent research [33] has also shown the possibility to infer broad information on large computer monitors via acoustic emanations from the voltage regulator. Similar to traditional computers, smartphones are also vulnerable to classical cache-based side-channel attacks [70]. Our work differs from these prior works by showing much finer grained information leakage of screen animation locations through the power line.

USB and other power vulnerabilities. As modern smartphones rely on USB to charge their batteries, multiple vulnerabilities have been found in the USB interface [60], including traffic monitoring [45], crosstalk leakage [54], keylogging side channels [44], malicious command execution [58], and trust exploitation [16]. While prior research has tried to filter malicious USB actions [57, 59], our work demonstrates that, even without any data transmission over the USB cable,

the power consumed can be exploited to extract fine-grained information such as user passcodes.

While ethernet over power line techniques have been utilized in both homes and data centers [22], Guri et al. demonstrated the possibilities of building covert channels over a power line [35]. Prior research has also shown that power consumption information can lead to various privacy issues, including key extraction on cryptographic systems [37] and laptops [34], state inference of home appliances [30], webpage identification of computers [27] and laptop user recognition [28]. Unlike these attacks, our work classifies ten on-screen animations in real time, directly exposing precise user input over the charging port.

9 Conclusion

This paper reveals a serious security threat, called Charger-Surfing, which exploits the power leakage of smartphones to infer the location of animations played on the touchscreen and steal sensitive information such as a user's passcode. The basic mechanism of Charger-Surfing monitors the power trace of a charging smartphone and extract button presses by leveraging signal processing and neural network techniques on the acquired signals. To assess the security risk of Charger-Surfing, we conduct a comprehensive evaluation of different types of smartphones and different users. Our evaluation results indicate that Charger-Surfing is victim-independent and achieves high accuracy when inferring a smartphone passcode (an average of 99.3% and 96.9% success rates when cracking a 4-digit and 6-digit passcode in five attempts, respectively). Furthermore, we build and test a portable, low-cost power trace collection system to launch a Charger-Surfing attack in practice. We then utilize this system to demonstrate that Charger-Surfing works well in real settings across different user configurations and devices. Finally, we present different countermeasures to thwart Charger-Surfing and discuss their feasibility.

Acknowledgements

We would like to thank the anonymous reviewers for their insightful and detailed comments, which helped us to improve the quality of this paper. This work was supported in part by the US ONR grant N00014-20-1-2153, ARO grant W911NF-19-1-0049, NSF grants CNS-2054657 and CPS-1739390.

References

[1] Behind The Charge: A Big Challenge for Hospitals. http://www.mkelements.com/blog/behind-charge-big-challenge-hospitals.

[2] Briant Park Blog: Solar-Powered Charging Stations Land in Bryant Park. http://blog.bryantpark.org/2014/07/solar-powered-charging-stations-land-in.html.

[3] Chargeport Hotel Charging Station. http://www.teleadapt.com/hospitality-products/powercharging/chargeport.

[4] Evil Maid Attack. https://en.wikipedia.org/wiki/Evil_maid_attack.

[5] Hackers Claim 'Any' Smartphone Fingerprint Lock Can Be Broken In 20 Minutes. https://www.forbes.com/sites/daveywinder/2019/11/02/smartphone-security-alert-as-hackers-claim-any-fingerprint-lock-broken-in-20-minutes/.

[6] Phone Battery Statistics Across Major US Cities. https://veloxity.us/phone-battery-statistics/.

[7] Phone Chargers: China's Latest Sharing Economy Fad. http://www.sixthtone.com/news/2182/phone-chargers-chinas-latest-sharing-economy-fad.

[8] Please Stop Charging Your Phone in Public Ports. https://money.cnn.com/2017/02/15/technology/public-ports-charging-bad-stop/index.html.

[9] Politician's Fingerprint 'Cloned from Photos' by Hacker. https://www.bbc.com/news/technology-30623611.

[10] Power Up: A Guide to US Airport Charging Stations. http://www.cheapflights.com/news/power-up-a-guide-to-us-airport-charging-stations/#ewr.

[11] Solar-Powered Phone Charging Stations Launch in Union Square. https://www.dnainfo.com/new-york/20130620/union-square/solar-powered-phone-chargingstations-launch-union-squarer.

[12] Adam Aviv, John Davin, Flynn Wolf, and Ravi Kuber. Towards Baselines for Shoulder Surfing on Mobile Authentication. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, 2017.

[13] Adam Aviv, Katherine Gibson, Evan Mossop, Matt Blaze, and Jonathan M Smith. Smudge Attacks on Smartphone Touch Screens. *Proceedings of 4th USENIX Workshop on Offensive Technologies*, 2010.

[14] Michael Backes, Tongbo Chen, Markus Duermuth, Hendrik Lensch, and Martin Welk. Tempest in a Teapot: Compromising Reflections Revisited. In *Proceedings of the 30th IEEE Symposium on Security and Privacy*, 2009.

[15] Michael Backes, Markus Dürmuth, and Dominique Unruh. Compromising Reflections-or-How to Read LCD monitors around the Corner. In *Proceedings of the 29th IEEE Symposium on Security and Privacy*, 2008.

[16] Darrin Barrall and David Dewey. Plug and Root, the USB Key to the Kingdom. *Presentation at Black Hat Briefings*, 2005.

[17] Yigael Berger, Avishai Wool, and Arie Yeredor. Dictionary Attacks Using Keyboard Acoustic Emanations. In *Proceedings of the 13th ACM conference on Computer and Communications Security*, 2006.

[18] Niels Brouwers, Marco Zuniga, and Koen Langendoen. NEAT: a Novel Energy Analysis Toolkit for Free-Roaming Smartphones. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, 2014.

[19] Liang Cai and Hao Chen. TouchLogger: Inferring Keystrokes on Touch Screen from Smartphone Motion. *Proceedings of the USENIX HotSec*, 2011.

[20] Aaron Carroll and Gernot Heiser. An Analysis of Power Consumption in a Smartphone. In *Proceedings of the USENIX Annual Technical Conference*, 2010.

[21] Hai-Wei Chen, Jiun-Haw Lee, Bo-Yen Lin, Stanley Chen, and Shin-Tson Wu. Liquid Crystal Display and Organic Light-Emitting Diode Display: Present Status and Future Perspectives. *Light: Science & Applications*, 2018.

[22] Li Chen, Jiacheng Xia, Bairen Yi, and Kai Chen. PowerMan: An Out-of-Band Management Network for Datacenters Using Power Line Communication. In *Proceedings of the 15th USENIX Symposium on Networked Systems Design and Implementation*, 2018.

[23] Qi Alfred Chen, Zhiyun Qian, and Zhuoqing Morley Mao. Peeking into Your App without Actually Seeing It: UI State Inference and Novel Android Attacks. In *Proceedings of the 23rd USENIX Security Symposium*, 2014.

[24] Xiang Chen, Yiran Chen, Zhan Ma, and Felix Fernandes. How is Energy Consumed in Smartphone Display Applications? In *Proceedings of the 14th ACM Workshop on Mobile Computing Systems and Applications*, 2013.

[25] Yimin Chen, Xiaocong Jin, Jingchao Sun, Rui Zhang, and Yanchao Zhang. POWERFUL: Mobile App Fingerprinting via Power Analysis. In *Proceedings of the IEEE Conference on Computer Communications*, 2017.

[26] Yimin Chen, Tao Li, Rui Zhang, Yanchao Zhang, and Terri Hedgpeth. EyeTell: Video-Assisted Touchscreen Keystroke Inference from Eye Movements. In *Proceedings of the 2018 IEEE Symposium on Security and Privacy*, 2018.

[27] Shane Clark, Hossen Mustafa, Benjamin Ransford, Jacob Sorber, Kevin Fu, and Wenyuan Xu. Current Events: Identifying Webpages by Tapping the Electrical Outlet. In *European Symposium on Research in Computer Security*. Springer, 2013.

[28] Mauro Conti, Michele Nati, Enrico Rotundo, and Riccardo Spolaor. Mind the Plug! Laptop-User Recognition Through Power Consumption. In *Proceedings of the 2nd ACM International Workshop on IoT Privacy, Trust, and Security*, 2016.

[29] Mian Dong and Lin Zhong. Chameleon: a Color-Adaptive Web Browser for Mobile OLED Displays. In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, 2011.

[30] Jingyao Fan, Qinghua Li, and Guohong Cao. Privacy Disclosure Through Smart Meters: Reactive Power Based Attack and Defense. In *Proceedings of the 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2017.

[31] Dinei Florencio and Cormac Herley. A Large-Scale Study of Web Password Habits. In *Proceedings of the 16th International Conference on World Wide Web*. ACM, 2007.

[32] Daniel Genkin, Lev Pachmanov, Itamar Pipman, Eran Tromer, and Yuval Yarom. ECDSA Key Extraction from Mobile Devices via Nonintrusive Physical Side Channels. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[33] Daniel Genkin, Mihir Pattani, Roei Schuster, and Eran Tromer. Synesthesia: Detecting screen content via remote acoustic side channels. In *IEEE Symposium on Security and Privacy (SP)*, 2019.

[34] Daniel Genkin, Itamar Pipman, and Eran Tromer. Get Your Hands off My Laptop: Physical Side-Channel Key-Extraction Attacks on PCs. *Journal of Cryptographic Engineering*, 2015.

[35] Mordechai Guri, Boris Zadov, Dima Bykhovsky, and Yuval Elovici. PowerHammer: Exfiltrating Data from Air-Gapped Computers through Power Lines. *IEEE Transactions on Information Forensics and Security*, 2020.

[36] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep Learning for Time Series Classification: A Review. *Data Mining and Knowledge Discovery*, 2019.

[37] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In *Proceedings of the Annual International Cryptology Conference*. Springer, 1999.

[38] Lingjun Li, Xinxin Zhao, and Guoliang Xue. Unobservable Re-Authentication for Smartphones. In *Proceedings of the 20th Network and Distributed System Security Symposium*, 2013.

[39] Xiao Ma, Peng Huang, Xinxin Jin, Pei Wang, Soyeon Park, Dongcai Shen, Yuanyuan Zhou, Lawrence Saul, and Geoffrey Voelker. Edoctor: Automatically Diagnosing Abnormal Battery Drain Issues on Smartphones. In *Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation*, 2013.

[40] Jani Mantyjarvi, Mikko Lindholm, Elena Vildjiounaite, S-M Makela, and HA Ailisto. Identifying Users of Portable Devices from Gait Pattern with Accelerometers. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2005.

[41] Martin Marinov. TempestSDR. https://github.com/martinmarinov/TempestSDR, 2013.

[42] Yan Michalevsky, Aaron Schulman, Gunaa Arumugam Veerapandian, Dan Boneh, and Gabi Nakibly. PowerSpy: Location Tracking Using Mobile Device Power Analysis. In *Proceedings of the 24th USENIX Security Symposium*, 2015.

[43] Emiliano Miluzzo, Alexander Varshavsky, Suhrid Balakrishnan, and Romit Roy Choudhury. Tapprints: Your Finger Taps Have Fingerprints. In *Proceedings of the 10th ACM International Conference on Mobile Systems, Applications, and Services*, 2012.

[44] John Monaco. SoK: Keylogging Side Channels. In *Proceedings of the 2018 IEEE Symposium on Security and Privacy*. IEEE, 2018.

[45] Matthias Neugschwandtner, Anton Beitler, and Anil Kurmus. A Transparent Defense Against USB Eavesdropping Attacks. In *Proceedings of the 9th ACM European Workshop on System Security*, 2016.

[46] Emmanuel Owusu, Jun Han, Sauvik Das, Adrian Perrig, and Joy Zhang. ACCessory: Password Inference Using Accelerometers on Smartphones. In *Proceedings of the 12th ACM Workshop on Mobile Computing Systems and Applications*, 2012.

[47] Abhinav Pathak, Charlie Hu, and Ming Zhang. Where is the Energy Spent Inside My App?: Fine Grained Energy Accounting on Smartphones with eprof. In *Proceedings of the 7th ACM European Conference on Computer Systems*, 2012.

[48] Abhinav Pathak, Charlie Hu, Ming Zhang, Paramvir Bahl, and Yi-Min Wang. Fine-Grained Power Modeling for Smartphones Using System Call Tracing. In *Proceedings of the 6th ACM Conference on Computer Systems*, 2011.

[49] Rahul Raguram, Andrew White, Dibyendusekhar Goswami, Fabian Monrose, and Jan-Michael Frahm. iSpy: Automatic Reconstruction of Typed Input from Compromising Reflections. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, 2011.

[50] J Rogers. Please Enter Your Four-Digit Pin. *Financial Services Technology, US Edition*, 2007.

[51] Len Sherman. The Basics of USB Battery Charging: A Survival Guide. *Maxim Integrated Products, Inc*, 2010.

[52] Zdeňka Sitová, Jaroslav Šeděnka, Qing Yang, Ge Peng, Gang Zhou, Paolo Gasti, and Kiran Balagani. HMOG: New Behavioral Biometric Features for Continuous Authentication of Smartphone Users. *IEEE Transactions on Information Forensics and Security*, 2016.

[53] Riccardo Spolaor, Laila Abudahi, Veelasha Moonsamy, Mauro Conti, and Radha Poovendran. No Free Charge Theorem: A Covert Channel via USB Charging Cable on Mobile Devices. In *International Conference on Applied Cryptography and Network Security*. Springer, 2017.

[54] Yang Su, Daniel Genkin, Damith Ranasinghe, and Yuval Yarom. USB Snooping Made Easy: Crosstalk Leakage Attacks on USB Hubs. In *Proceedings of the 26th USENIX Security Symposium*, 2017.

[55] Jingchao Sun, Xiaocong Jin, Yimin Chen, Jinxue Zhang, Yanchao Zhang, and Rui Zhang. VISIBLE: Video-Assisted Keystroke Inference from Tablet Backside Motion. In *Proceedings of the 23rd Network and Distributed System Security Symposium*, 2016.

[56] Di Tang, Zhe Zhou, Yinqian Zhang, and Kehuan Zhang. Face Flashing: a Secure Liveness Detection Protocol based on Light Reflections. *Proceedings of the 25th Network and Distributed System Security Symposium*, 2018.

[57] Dave Jing Tian, Adam Bates, and Kevin Butler. Defending Against Malicious USB Firmware with GoodUSB. In *Proceedings of the 31st Annual Computer Security Applications Conference*, 2015.

[58] Dave (Jing) Tian, Grant Hernandez, Joseph I. Choi, Vanessa Frost, Christie Raules, Patrick Traynor, Hayawardh Vijayakumar, Lee Harrison, Amir Rahmati, Michael Grace, and Kevin Butler. ATtention Spanned: Comprehensive Vulnerability Analysis of AT Commands Within the Android Ecosystem. In *Proceedings of the 27th USENIX Security Symposium*, 2018.

[59] Dave (Jing) Tian, Nolen Scaife, Adam Bates, Kevin Butler, and Patrick Traynor. Making USB Great Again with USBFILTER. In *Proceedings of the 25th USENIX Security Symposium*, 2016.

[60] J. Tian, N. Scaife, D. Kumar, M. Bailey, A. Bates, and K. Butler. SoK: "Plug Pray" Today – Understanding USB Insecurity in Versions 1 Through C. In *2018 IEEE Symposium on Security and Privacy*, May 2018.

[61] Weitao Xu, Guohao Lan, Qi Lin, Sara Khalifa, Neil Bergmann, Mahbub Hassan, and Wen Hu. Keh-Gait: Towards a Mobile Healthcare User Authentication System by Kinetic Energy Harvesting. In *Proceedings of the 24th Network and Distributed System Security Symposium*, 2017.

[62] Yi Xu, Jared Heinly, Andrew White, Fabian Monrose, and Jan-Michael Frahm. Seeing Double: Reconstructing Obscured Typed Input from Repeated Compromising Reflections. In *Proceedings of the 2013 ACM SIGSAC conference on Computer and communications security*, 2013.

[63] Zhi Xu, Kun Bai, and Sencun Zhu. Taplogger: Inferring User Inputs on Smartphone Touchscreens Using On-Board Motion Sensors. In *Proceedings of the 5th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2012.

[64] Lin Yan, Yao Guo, Xiangqun Chen, and Hong Mei. A Study on Power Side Channels on Mobile Devices. In *Proceedings of the 7th Asia-Pacific Symposium on Internetware*, 2015.

[65] Qing Yang, Paolo Gasti, Gang Zhou, Aydin Farajidavar, and Kiran Balagani. On Inferring Browsing Activity on Smartphones via USB Power Analysis Side-Channel. *IEEE Transactions on Information Forensics and Security*, 2017.

[66] Guixin Ye, Zhanyong Tang, Dingyi Fang, Xiaojiang Chen, Kwang In Kim, Ben Taylor, and Zheng Wang. Cracking Android Pattern Lock in Five Attempts. In *Proceedings of the 24th Network and Distributed System Security Symposium*, 2017.

[67] Qinggang Yue, Zhen Ling, Xinwen Fu, Benyuan Liu, Kui Ren, and Wei Zhao. Blind Recognition of Touched Keys on Mobile Devices. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014.

[68] Lide Zhang, Birjodh Tiwana, Zhiyun Qian, Zhaoguang Wang, Robert Dick, Morley Mao, and Lei Yang. Accurate Online Power Estimation and Automatic Battery Behavior Based Power Model Generation for Smartphones. In *Proceedings of the 8th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, 2010.

[69] Xiaokuan Zhang, Xueqiang Wang, Xiaolong Bai, Yinqian Zhang, and XiaoFeng Wang. OS-level Side Channels without Procfs: Exploring Cross-App Information Leakage on iOS. In *Proceedings of the 25th Network and Distributed System Security Symposium*, 2018.

[70] Xiaokuan Zhang, Yuan Xiao, and Yinqian Zhang. Return-Oriented Flush-Reload Side Channels on ARM and their Implications for Android Devices. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[71] Nan Zheng, Kun Bai, Hai Huang, and Haining Wang. You Are How You Touch: User Verification on Smartphones via Tapping Behaviors. In *Proceedings of the IEEE 22nd International Conference on Network Protocols*, 2014.

[72] Man Zhou, Qian Wang, Jingxiao Yang, Qi Li, Feng Xiao, Zhibo Wang, and Xiaofen Chen. PatternListener: Cracking Android Pattern Lock Using Acoustic Signals. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018.

[73] Li Zhuang, Feng Zhou, and Doug Tygar. Keyboard Acoustic Emanations Revisited. *ACM Transactions on Information and System Security*, 2009.

Appendices

A Additional Figures and Tables

Table 9: Smartphones Used For Evaluation

Phone (Release Year)	OS	Processor	GPU	Screen	
				Resolution	Technology
Motorola G4 (2016)	Android 6.0.1	4 x 1.5 GHz A-53 4 x 1.2 GHz A-53	Adreno 405	1920x1080	LCD
Samsung Galaxy Nexus (2012)	Android 6.0.1	2 x 1.2 GHz A-9	PowerVR SGX540	1280x720	Super AMOLED
Apple iPhone 6+ (2014)	iOS 12.1	2 x 1.4 GHz Typhoon	PowerVR GX6450	1920x1080	LCD
Apple iPhone 8+ (2017)	iOS 12.1.2	2 x 2.3 GHz Monsoon 4 x 1.4 GHz Mistral	Apple GPU	1920x1080	LCD

Table 10: Classification Network Used for iPhone

iPhone Classification Network		
Layer	Operation	Kernel Size
1	Input	100000x1
2	Convolution	50x50
3	MaxPool	5
4	Convolution	50x50
5	MaxPool	5
6	Convolution	50x50
7	MaxPool	5
8	Convolution	50x50
9	GlobalAveragePool	-
10	Dropout	0.5
11	Dense	10

Table 11: Classification Network Used for Android

Android Classification Network		
Layer	Operation	Kernel Size
1	Input	800000x1
2	Convolution	100x100
3	MaxPool	5
4	Convolution	50x75
5	MaxPool	5
6	Convolution	50x75
7	MaxPool	5
8	Convolution	50x75
9	GlobalAveragePool	-
10	Dropout	0.5
11	Dense	10