PIN Skimming: Exploiting the Ambient-Light Sensor in Mobile Devices

Raphael Spreitzer
Graz University of Technology, Austria
raphael.spreitzer@iaik.tugraz.at

ABSTRACT

The pervasive usage of mobile devices, *i.e.*, smartphones and tablet computers, and their vast amount of sensors represent a plethora of side channels posing a serious threat to the user's privacy and security. In this paper, we propose a new type of side channel which is based on the ambient-light sensor employed in today's mobile devices. While recent advances in this area of research focused on the employed motion sensors and the camera as well as the sound, we investigate a less obvious source of information leakage, namely the ambient light. We successfully demonstrate that minor tilts and turns of mobile devices cause variations of the ambient-light sensor information. Furthermore, we show that these variations leak enough information to infer a user's personal identification number (PIN) input based on a set of known PINs. Our results clearly show that we are able to determine the correct PIN—out of a set of 50 random PINs—within the first ten guesses about 80% of the time. In contrast, the chance of finding the right PIN by randomly guessing ten PINs would be 20%. Since the data required to perform such an attack can be gathered without any specific permissions or privileges, the presented attack seriously jeopardizes the security and privacy of mobile-device owners.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection

Keywords

Mobile security, side-channel attack, sensor-based attack, ambient-light sensor, mobile malware

1. INTRODUCTION

Mobile devices such as smartphones and tablet computers have become a ubiquitous part of our everyday life. Powerful processors and a variety of sensors led to manifold applications being developed on these devices. Besides more general applications that allow users to browse the Internet as well as to take and view pictures, more sophisticated applications like augmented-reality applications and games also exist. However, these devices are not solely employed for entertainment applications but also for handling business applications such as banking transactions and payment services. This in turn leads to sensitive information being processed on these devices, which also attracts the attention of criminals and imposters aiming to steal money from users and to spy on specific people through malicious applications. Hence, the investigation of security and privacy compromising threats is of utmost importance.

One potential source of security and privacy compromising threats is denoted to side channels. Side channels represent unintended information leakage during the operation of a device and potentially allow attackers to recover secret information, as demonstrated by Kocher [33] in 1996. While traditional side-channel attacks against smartcards require expensive hardware, smartphones employ many different types of sensors and features that can be exploited for side-channel attacks. Thus, the "hardware" required to perform side-channel attacks is provided for free.

In other words, besides providing useful information, these sensors also represent a variety of threats to the user's security and privacy. Though Android employs a permission system to prevent malicious access to specific device resources, many of the employed sensors do not require any permission at all. This exacerbates the severity of these vulnerabilities since applications without any specific permissions are able to exploit these side channels, something that has already been demonstrated impressively. For instance, Marquardt *et al.* [35] developed an application intended to run on a smartphone that is capable of recovering inputs of nearby keyboards by exploiting the vibrations recored via the accelerometer sensor. In 2012, Aviv *et al.* [8] demonstrated the possibility of extracting the personal identification number (PIN) and the unlock pattern by exploiting the information provided by the accelerometer sensor in smartphones. In addition, Miluzzo *et al.* [37] have shown how to extract pressed keys from the accelerometer sensor in combination with the gyroscope sensor. In 2013, Simon and Anderson [46] demonstrated how to infer PINs through the camera and the microphone.

The importance of research in this area has been emphasized by Becher *et al.* [13]. In order to raise awareness about such attacks and to develop effective countermeasures, these vulnerabilities must be reported and analyzed extensively. To this end, we show how the ambient light of the user's
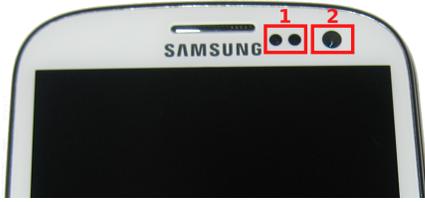
Figure 1: Samsung Galaxy SIII with (1) the proximity and ambient-light sensor as well as (2) the front camera.

environment affects the system's security. This is due to the fact that today's mobile devices are equipped with a light sensor that provides information about the ambient light. The most common usage of this sensor is to adjust the screen brightness depending on the light intensity. However, due to slight tilts and turns during the operation of the device, the information provided by the sensor allows an attacker to infer the input provided by the user. In this paper, we investigate the sensitive information leaked through this sensor.

Contribution. The contributions of this work can be summarized as follows. First, we show that the light sensor employed in today's mobile devices actually represents a new type of side channel that leaks the user's input, *e.g.*, the secret PIN input. Second, we observed that the light sensor of modern smartphones also captures the red, green, blue, and white (RGBW) intensities which leaks even more information and improves the accuracy of the attack. Third, we provide practical insights into the ways in which this side channel can be exploited to gather the secret PIN input of the user by employing machine-learning algorithms. Fourth, we discuss potential mitigation techniques to prevent the exploitation of sensor-based side channels. Finally, we also compare the presented attack to similar sensor-based attacks.

Outline. The remainder of this paper is structured as follows. In Section 2 we start with a basic investigation of the ambient-light sensor and provide an insight into the information leaked through this sensor. Section 3 outlines one potential attack scenario which takes advantage of this side channel to recover the secret PIN input provided by the user. Furthermore, we discuss the security implications of such attacks. Later on, in Section 4, we detail how the leaked information is actually exploited by employing machine-learning algorithms. In Section 5 we extensively analyze the information gathered and provide detailed evaluations regarding the applicability and reliability of this attack. We cover limitations of the presented attack in Section 6 and we provide a brief analysis of countermeasures to prevent such attacks in Section 7. Last but not least, we discuss related work in Section 8 and present a conclusion for this paper in Section 9.

2. AMBIENT-LIGHT SENSOR

The ambient-light sensor employed in many of today's mobile devices provides information about the intensity of the surrounding illumination. Figure 1 illustrates the proximity and ambient-light sensor as well as the front camera on a

Table 1: Sampling rates on the Samsung Galaxy SIII

Rate parameter	Sample rate
SensorManager.SENSOR_DELAY_FASTEST (0)	∼ 750 Hz
SensorManager.SENSOR_DELAY_GAME (1)	∼ 49 Hz
SensorManager.SENSOR_DELAY_UI (2)	∼ 15 Hz
SensorManager.SENSOR_DELAY_NORMAL (3)	∼ 5 Hz
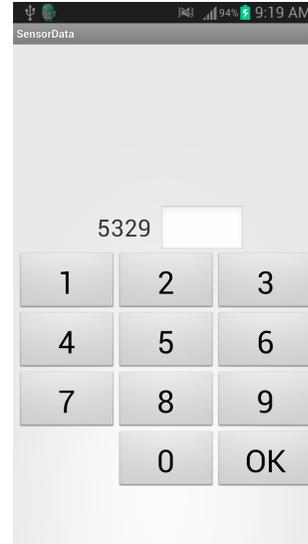
Figure 2: PIN input mask to gather test samples.

Samsung Galaxy SIII, which acts as our test device. The information reported from the ambient-light sensor is given in SI *lux* units, which measures the intensity of illumination of a surface. This information is used to adapt the screen brightness appropriately. For instance, outside in direct sunlight the screen brightness must be increased to remain readable, whereas in darker surroundings the screen is dimmed to reduce eye fatigue [45].

The ambient-light sensor can be accessed via the *Android Sensor API* [5] that allows applications to register listeners which are notified about changes of the sensor values. The method *SensorManager.registerListener(...)* accepts a *rate* parameter which determines how fast the events should be reported. Table 1 shows the resulting sampling frequencies for the different rate parameters according to our observations on the *Samsung Galaxy SIII* smartphone running Android 4.3. We determined the number of reported values over a period of 10 seconds for each of the listed parameters and we observed that about 750 measurement samples can be gathered per second. We also observed a sensor resolution of 1 *lux*, *i.e.*, the smallest detectable change is 1 *lux*.

The next step in determining the applicability of the light-sensor information regarding side-channel attacks is to test the sensitivity, *i.e.*, whether operating the smartphone actually results in changes of the information provided by the sensor. Therefore, we developed an Android application that randomly generates a 4-digit PIN and prompts the user to enter this PIN on the provided PIN pad. Figure 2 illustrates a screenshot of this application. During the input of the PIN the application collects the information provided by the ambient-light sensor as well as the corresponding timestamp of the reported value. Furthermore, we store a times-

Figure 3: Light-sensor information for five consecutive PIN inputs (1-5-9-0).

Figure 5: RGB(W)-sensor information for five consecutive PIN inputs (1-7-3-0).

Figure 4: Schematic of information leakage caused by tilts and turns of the smartphone.

tamp when a button is clicked as well as the digit of the clicked button itself. We collected this information for five consecutive PIN inputs and visualized the gathered information in Figure 3. For the sake of clarity we also plotted the corresponding digits of the PIN and, as can be seen in this plot, a recurring pattern for different digits of the PIN (1-5-9-0) can be observed. These differences in the light intensity during the input of the PIN occur inevitably due to slight tilts and turns while operating the smartphone. Figure 4 shows a schematic illustrating the tilts and turns leading to variations of the reported sensor information. For instance, assuming the light bulb being the main source of light, then tilting the device to the left causes a decrease of the reported *lux* value. Although we illustrate a point-like light source, the attack also works for environments that are uniformly lit via tube lights.

Note that the PIN input mask can be aligned on the top or on the bottom of the screen. For instance, on our test device the SIM-unlock PIN mask is aligned on the top while the standard phone pad is aligned on the bottom. However, our observations showed that the alignment does not influence the information leakage.

Accessing the ambient-light sensor does not require special permissions and, hence, any malicious application can gather the required information without raising any suspicion. Contrary to the shutter sound or a LED that indicate an active camera [4], there is not even an audio-visual feedback that signals the user that data is being collected with the sensor. Thus, we conclude that the information provided by the ambient-light sensor can be exploited in side-channel attacks, which means that an attacker is able to determine the corresponding input based on this information.

2.1 RGB(W) Sensor

More recent smartphones, including the *Samsung Galaxy SIII* [43] as well as the *Samsung Galaxy S4* [42], also employ an RGB(W) sensor which is capable of reporting the red, green, blue, and white (RGBW) intensities of the ambient light. Samsung refers to this sensor as an RGB sensor, but since it also reports the white intensity of the ambient light we refer to it as RGB(W) sensor. According to Samsung Electronics [45], the RGB sensor is used to optimize the screen brightness and sharpness to prevent eye fatigue.

The *Android API* does not yet support RGB(W) sensors, not even the latest version (Android 4.4 *KitKat* [3]). Thus, the desired sensor values must be read from the virtual file system directly. On the *Samsung Galaxy SIII* reading */sys/devices/virtual/sensors/light_sensor/raw_data* yields the RGBW values of the sensor. Again, no specific permissions or super-user privileges are required to read these sensor values. We verified the correctness of these values through the Service Menu [26]—accessible by entering *#0*# on the phone pad—which reports the same values.

Figure 5 illustrates the data of the same experiment as mentioned before, but this time we also include the information provided by the RGB(W) sensor. Due to reasons of readability, we plotted only every 10-th value of the RGBW intensities. We observe that all values show a similar curve, but the intensity of the blue light seems to provide a smoother curve than the other four values. Again, we plotted the event of a digit input and labeled each event appropriately. For this specific plot we observe that the "OK" button leads to a rather heavy decrease of the sensor values.

Our observations show that the RGBW information provides additional information that can be exploited, *i.e.*, additional features to be used for the machine-learning algorithm later on.

3. ATTACK SCENARIO

In this section, we outline one possible attack scenario. Based on this scenario, we will later on illustrate the individual steps to reproduce the actual exploitation of the light-sensor information by employing machine-learning techniques. Since the captured light-sensor information reflects the user's ambient-light conditions, a generic attack requires further investigations and is, thus, beyond the scope of this work. Currently, we consider a scenario where the user provides the data to be used to train the machine-learning algorithm in the same environmental setting as the actual data which is to be classified later on. Hence, we consider the following situation as a possible attack scenario.

3.1 Training Phase

A hand-crafted application, *i.e.*, an addicting game, is used to gather the light-sensor information during the user's interaction with the smartphone. The application in question tricks the user into operating the smartphone in a way that is similar to the input of multiple PINs. For example, an application similar to *Math Trainer* [47] could be used, where the user is supposed to solve mathematical puzzles and to enter numbers which can be seen as PINs.

We assume that users play such games in the living room while watching TV, in a waiting room while waiting for an appointment, or during a train ride. Hence, even though mobile devices are carried around all the time, there are still many opportunities to gather the required data from users operating the smartphone while not walking around. Furthermore, a recent study performed by the UK's Office of Communications [49] coined the term *media stacking*, which refers to the fact that about half of UK's adults conduct their smartphone or tablet computer while watching TV. Thus, our assumption seems to be reasonable and the outlined attack scenario is rather realistic.

The game, as outlined above, might only be able to capture a limited number of training samples (PINs) to be used for the classification of the unknown PIN. However, as has been shown in 2012 by Bonneau *et al.* [17] as well as Jakobsson and Liu [30], people tend to choose specific PINs like dates as well as PINs that represent common four-digit words, *e.g.*, *"love"* (5683). Thus, taking research about the users' tendency to choose PINs into consideration, a set of commonly used PINs could be established and might be enough to determine the correct one. The above mentioned game can be used to learn this set of commonly used PINs.

In case computing power for the machine-learning algorithm is required, *i.e.*, a powerful server, the application requires an Internet connection to transmit the gathered data to this powerful intermediary. Nevertheless, we argue that convincing the user that the application requires the *Internet* permission is easy. For instance, to retrieve high scores or information about new updates, new games, etc. Thus, we claim that the proposed attack can be performed without raising the user's suspicion.

3.2 Exploitation Phase

After gathering enough samples, the application tricks the user into restarting the device or starting the desired application, *e.g.*, the banking application, just to capture the light-sensor information during the input of the authentication PIN. If one considers to attack the smartphone's PIN, a service can be implemented to be started on boot time. Af-

terwards the sensor information captured during the game play is used to deduce the unknown PIN input by means of machine learning. Note that the game, *e.g.*, *Math Trainer*, might also trick the user into buying a specific add-on, a "new level", or a "new stage". When the user performs the in-app billing via *Google Wallet*, the game skims the corresponding authentication PIN.

Now one might question whether the revealed PIN is of any value for the attacker. In fact, if the attacker later gains physical access to the mobile device, she might gain access to the mobile device by unlocking the phone, or even worse, might cause financial damage by authenticating herself to the corresponding application using the correct PIN. Furthermore, Aviv *et al.* [8] argued that the learned PIN might be valuable in case the user reuses the PIN, for instance, as the ATM PIN. In addition, Simon and Anderson [46] predict that the number of smartphone applications requiring an authentication PIN will increase over time. Hence, users might be tempted to reuse one PIN across different applications, which exacerbates PIN-skimming vulnerabilities. Overall, we consider the presented attack as a serious threat for today's mobile devices.

3.3 Security Implications

Simon and Anderson [46] state that sensor-based side-channel attacks are capable to overcome strong separation mechanisms like *Samsung KNOX* [44] or *BlackBerry Balance* [15]. These mechanisms try to separate the "private" world from the "business" world in order to provide better protection of corporate data on smartphones. Such mechanisms are of special interest in the *Bring Your Own Device* (BYOD) context where employees integrate their personal devices into the company's infrastructure. However, based on the leaking sensor information, an unprivileged application running in the "private" world of the smartphone might gain knowledge of sensitive information entered in the "business" world. It is important to note that the presented attack is also capable to overcome these separation mechanisms if unprivileged applications running in the "private" world are allowed to access the ambient-light sensor during the operation of the "business" world. Furthermore, since BlackBerry smartphones also support the execution of Android applications and the *BlackBerry Runtime 10.0* [16] also supports the ambient-light sensor, a malicious application developed for Android devices might also harm BlackBerry devices.

3.4 Observations and Assumptions

Considering a PIN-input mask as illustrated in Figure 2, and a user operating the smartphone with only one hand, using the thumb to enter the digits, we make the following observations. Left-handed persons tilt the smartphone slightly to the left side when entering PIN digits in the middle and right column of the key pad, *i.e.*, 2, 3, 5, 6, 8, 9. In contrast, right-handed persons tilt the smartphone slightly to the right side when entering PIN digits in the left and middle column, *i.e.*, 1, 2, 4, 5, 7, 8. We attribute this observation to the fact that people slightly push the display towards the thumb. These tilts of the device cause the variations in the captured light-sensor information. Similarly, we expect fewer tilts and turns of the mobile device if it is held with one hand and operated with the index finger of the other hand or a stylus pen. Based on these observations we make the following assumptions.

Assumption 1. We assume the user is holding the mobile device in his hands rather than laying it onto a flat surface while operating it. If we would assume the mobile device is lying on a stable surface, *i.e.*, a table, the light-sensor information would not change during the operation of the device, unless the user's hand causes the light-sensor changes.

Assumption 2. Furthermore, we assume that the PIN is entered on a key pad similar to the one illustrated in Figure 2 rather than a QWERTY keyboard with a single row of numbers. Examples of applications that are "protected" with an authentication PIN are, for instance, AppLock [21], Evernote [22], and KeepSafe [31], as well as mobile banking applications, *e.g.*, Barclays [12], and NAB [38], just to name a few of them. Screenshots of these applications—provided by the corresponding developers—show that the authentication PIN is entered on a key pad as illustrated in Figure 2. Hence, this seems to be a rather fair assumption which does not have a negative impact on the attack scenario.

Assumption 3. We also assume that the user operates the mobile device in an environment where the light sensor faces a sufficiently large variance of the ambient light. This is not the case in completely dark environments. However, also rather gloomy environments, *i.e.*, a room in the late afternoon without any artificial light source (rather diffuse light), can be considered for potential attack scenarios, at least in case the *lux* values vary slightly during the operation of the device.

4. ATTACK APPROACH

In this section, we detail the steps for the exploitation of the light-sensor information. As outlined in the scenario above, we perform a matching of sensor values captured during the input of an unknown PIN to the sensor values of known PINs. In terms of machine learning this represents a classification problem, where a so-called feature vector is mapped to a finite number of labels or categories, *i.e.*, PINs in our case. The required steps are as follows: (1) gathering the sensor values under known PINs as well as the sensor values under unknown PINs, and (2) employing machine-learning techniques to determine the unknown PINs based on the set of known PINs.

In order to perform the classification, sensor values and the corresponding PINs are used to train the classifier. This data is referred to as training data. The actual data that is to be classified is known as test data. We stick to the notation of Alpaydin [2] and Bishop [14], *i.e.*, bold letters denote vectors and a superscript T denotes the transpose of a vector. Furthermore, uppercase bold letters denote matrices. Vectors are assumed to be column vectors by default.

4.1 Gathering the Required Data

The gathering of the required training data during game play can be formalized as follows. The malicious application captures a list of tuples (t, L, R, G, B, W), with t being the timestamp and L, R, G, B, and W representing the *lux* information, as well as the red, green, blue, and white intensities of the ambient light. Furthermore, we capture a list of tuples (t_p, d), with t_p being the timestamp of a pressed digit $d \in \{0, \ldots, 9\}$ of the p-th PIN. In our scenario one PIN consists of four consecutive tuples in this list. The two

timestamps t and t_p allow us to align the event of a pressed digit with the data of the ambient-light sensor later on.

For each PIN we extract the sequence of sensor values within the time period defined by the timestamp of the first digit and the timestamp of the last digit of the PIN. In addition, one might consider a time frame of a few milliseconds before and after the input of the first and the last digit of one PIN, which covers additional information. The resulting matrix $\mathbf{M}$ for one PIN is as follows.

$$
\mathbf{M} = \begin{bmatrix} (t, L, R, G, B, W)_1 \\ \vdots \\ (t, L, R, G, B, W)_l \end{bmatrix}
$$

Every column within matrix $\mathbf{M}$ represents the corresponding sensor information during the input of one specific PIN, except the first one which represents the timestamp at which the information was captured. Before the gathered information can be exploited, we normalize the sensor values appropriately. This normalization of the data is done either by dividing each value (L, R, G, B, W) in one column by the norm of the corresponding column vector, or by rescaling the values via, *e.g.*, $L_i = (L_i - \min(L)) / (\max(L) - \min(L))$.

The data of matrix $\mathbf{M}$ is then used to derive the actual feature vectors. While Owusu *et al.* [39] consider a large number of features and employ special feature selectors to determine the actual set of features to be used, we keep the feature space simple and stick to a limited number of features for our investigation. These features are outlined briefly within the following paragraphs.

Lux Values Only. The first set of feature vectors we consider are the exact *lux* values at the input of each specific digit of the PIN. Therefore, we use the timestamp of each input event during one PIN input to look-up the corresponding values in matrix $\mathbf{M}$. We represent these values as a vector $\mathbf{x} = [L_1, L_2, L_3, L_4]^{\mathrm{T}}$, where the subscript refers to a specific digit of the PIN.

Lux Values Including RGBW Values. The second set of feature vectors we consider are the exact *lux* values including the red, green, blue, and white (RGBW) intensities at the input of each digit for one specific PIN, represented as $\mathbf{x} = [L_1, R_1, G_1, B_1, W_1, \ldots, L_4, R_4, G_4, B_4, W_4]^{\mathrm{T}}$. In the following we refer to the feature vector comprised of these five features as $LRGBW$.

Polynomial of Degree 3. The third possibility we consider is fitting a polynomial of degree 3 through the second column of $\mathbf{M}$. More formally, through all *lux* values during one PIN input. The coefficients of this polynomial $f(x) = ax^3 + bx^2 + cx + d$ are then considered as the features of a specific PIN, *i.e.*, $\mathbf{x} = [a, b, c, d]^{\mathrm{T}}$. The coefficients for the red, green, blue, and white intensities—the remaining columns of $\mathbf{M}$—are obtained in the same manner and appended to $\mathbf{x}$.

After gathering the data as outlined above, any set of the outlined feature vectors is combined into a matrix $\mathbf{F}_n$ with n rows, *i.e.*, one row for each PIN. Furthermore, a class vector $\mathbf{c}_n = [(d_1, d_2, d_3, d_4)_1, \ldots, (d_1, d_2, d_3, d_4)_n]$ of tuples

corresponding to the PIN-digits can be derived.

$$\mathbf{F}_n = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{bmatrix}, \ \mathbf{c}_n = \begin{bmatrix} (d_1, d_2, d_3, d_4)_1 \\ \vdots \\ (d_1, d_2, d_3, d_4)_n \end{bmatrix}$$

The matrix $\mathbf{F}_n$ as well as the label vector $\mathbf{c}_n$ are then used to train the classification algorithm.

4.2 Detecting PIN Inputs

The above outlined approach of gathering the feature vectors can be performed in a real attack because the data for training the machine-learning algorithm can be captured during game play, in which situation all of this information is available, including the timestamp of a pressed digit. However, during the input of the unknown PIN the timestamp is not known and hence we need a mechanism to determine the PIN input on a sequence of sensor values. Therefore, the approach of Miluzzo *et al.* [37] might be used. Their idea is to move "windows" of a fixed length (the length of an average PIN input time) over the sensor data and try to detect the input. Another approach by Simon and Anderson [46] suggests to use the microphone to capture the vibrations of the haptic feedback to determine when a button on the touchscreen has been pressed. While the former approach does not require any specific permission, the latter approach requires a permission to access the microphone. However, we consider the detection of PIN inputs on a sequence of sensor values as solved and beyond the scope of this work.

4.3 Determining the Unknown PIN

After we gathered the required light-sensor information for all the PINs ($\mathbf{F}_n$ and $\mathbf{c}_n$) to be used for the training phase of the machine-learning algorithm, we start the actual training phase. Therefore, we employ Matlab's Statistics Toolbox [36] with its extensive features and machine-learning algorithms. We perform the outlined attack by employing a *supervised learning* algorithm, which tries to learn a function and its parameters based on labeled training data. This function is later on used to determine the label of unseen data. More formally, given a set of tuples $(\mathbf{x}_i, c_i)$, with $\mathbf{x}_i \in \mathbb{R}^n$ being a feature vector and c_i the corresponding label of the observation, the algorithm tries to infer a function $f : X \rightarrow C$, where $X \in \mathbb{R}^n$ represents the feature vector of an observation and C the inferred label.

In this work, we investigate three different classification algorithms—which have also been used in related work [8, 37]—and compare their results afterwards. We briefly outline the chosen classification algorithms in the following paragraphs. For further information about these algorithms we refer the reader to [2, 14].

Multiclass Logistic Regression. The classifier tries to learn parameters $\mathbf{w}_k$ for every class label $k \in C$, such that a vector $\mathbf{x}$ is assigned to label k in case $p(k|\mathbf{x}) > p(j|\mathbf{x})$ for all $j \neq k$, with p defined as below.

$$p(k|\mathbf{x}) = \frac{\exp\left(\mathbf{w}_k^{\mathrm{T}} \cdot \mathbf{x}\right)}{\sum_i \exp(\mathbf{w}_i^{\mathrm{T}} \cdot \mathbf{x})}$$

Discriminant Analysis. The classifier tries to learn parameters $\mathbf{w}_k$ for every class label $k \in C$, such that a vector $\mathbf{x}$ is assigned to label k in case $f_k(\mathbf{x}) > f_j(\mathbf{x})$ for all $j \neq k$,

with f defined as below.

$$f_k(\mathbf{x}) = \mathbf{w}_k^{\mathrm{T}} \cdot \mathbf{x} + w_{k0}$$

When talking about *discriminant analysis* we refer to *linear discriminant analysis*, where classes are separated linearly.

K-Nearest Neighbor Algorithm. The algorithm assigns the input vector $\mathbf{x}$ a label which is determined by the majority of the K nearest neighbors. In our case, we used $K = 5$. Usually, the Euclidean distance is used to determine the distance between two vectors.

5. EVALUATION AND RESULTS

We engaged a total of ten users to acquire the necessary data for the evaluation of this information leakage. Each user was asked to enter at least one set of random PINs with cardinality $k \in \{15, 30, 50\}$, each PIN for a specific number of times $N \in \{3, \dots, 10\}$. If a PIN was entered incorrectly, we ignore the corresponding input and prompt the user to enter the PIN again. Unless explicitly stated, we always use the largest set of 50 PINs within the following analysis. We basically follow the approach of Aviv *et al.* [8] who also measured the performance of their attack on a set of 50 PINs. In total we use the data of 29 test runs gathered by a total of ten users.

Evaluation Methodology. Since the performance of a learned classifier could be distorted (either positively or negatively) when determined solely with the gathered test data, we apply the concept of k-fold cross validation. The purpose of this concept is to estimate the average success rate of classifying unknown data into one of the learned categories. Compared to the performance based solely on specific test data, the concept of cross validation provides more reliable estimations regarding the performance of a classifier [14].

Setup. We performed the tests in rather unconstrained environments regarding the lighting conditions. This means that experiments were performed in office rooms that were uniformly illuminated via tube lights, in a living room with a standard ceiling lamp, and in rooms where the only light source was a window. We even considered different daytimes, *e.g.*, during the day and in the late afternoon which means that this attack was also tested for diffuse light conditions without a direct source of light rays. However, we asked the users not to walk around while entering the presented PINs, which is compliant with the above outlined attack scenario. Furthermore, we did not insist on a specific input method, but only that the users hold the mobile device during the operation. Allowing users to choose their desired input method freely yields more generic results because the users operate the mobile device in their usual manner.

We watched the users during the gathering of the measurement samples and observed the following input methods:

1. Holding the smartphone in one hand and entering the digits with the thumb of the same hand.

2. Holding the smartphone in one hand and entering the digits with the thumb of the other hand.

3. Holding the smartphone in one hand and entering the digits with the index finger of the other hand.
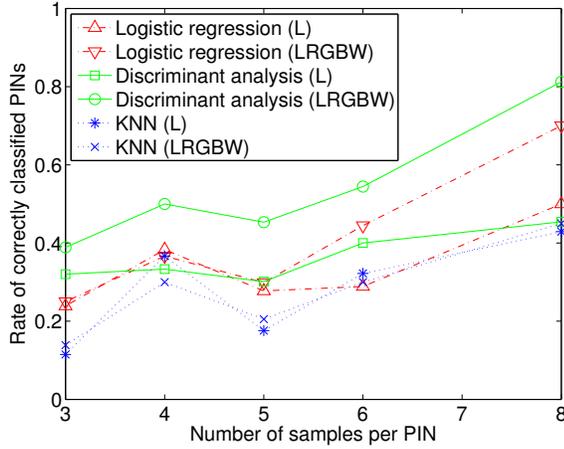
Figure 6: Average rate of correctly classified PINs over multiple runs with a set of 15 PINs each.

Figure 7: Average rate of correctly classified PINs over multiple runs with a set of 50 PINs each.

Within the following paragraphs we analyze the gathered data with the above mentioned machine-learning algorithms and the proposed feature vectors in order to determine whether the secret PIN input can be recovered based on a set of known PINs.

5.1 Comparison of Classification Algorithms

Our first intention is to determine the overall classification rate based on the different classification mechanisms: (1) *multiclass logistic regression* (logistic regression), (2) *linear discriminant analysis* (discriminant analysis), and (3) *k-nearest neighbor classification* (KNN). These classifiers are fed with the feature vectors: (a) the *lux* values only (L), and (b) the *lux* values including the RGBW values (LRGBW).

We applied a 10-fold cross validation on all three classifiers and evaluated the performance of the suggested features for different numbers of samples (repetitions) per PIN. Figure 6 illustrates the average rate of correctly classified PINs out of a set of 15 known PINs for the different classifiers and the proposed feature vectors. The y-axis represents the average rate of correctly classified PINs, and the x-axis illustrates the number of gathered samples (repetitions) per PIN. Given this plot, we observe that the feature vector comprised of the LRGBW values outperforms the *lux* value only feature vector most of the time for all classifiers. Thus, we conclude that the additional information leaked through the RGB(W) sensor leads to a better attack performance. We also observe that the *linear discriminant analysis* provides better results than the other two classifiers. Furthermore, the average rate of correctly classified PINs increases with the number of samples per PIN. For instance, if we perform a *linear discriminant analysis* with a training set of 15 PINs, each repeated 8 times, then we are able to classify more than 80% of the PINs correctly. Note that the chance of correctly guessing the right PIN from a set of 15 PINs randomly is $\frac{1}{15} = 6.7\%$.

Figure 7 illustrates the average rate of correctly classified PINs out of a set of 50 PINs. Again, the *linear discriminant analysis* outperforms the other two classifiers and the additional information from the RGB(W) sensor increases the performance compared to the *lux* value only. At first glance, an average rate of correctly classified PINs of 40 to
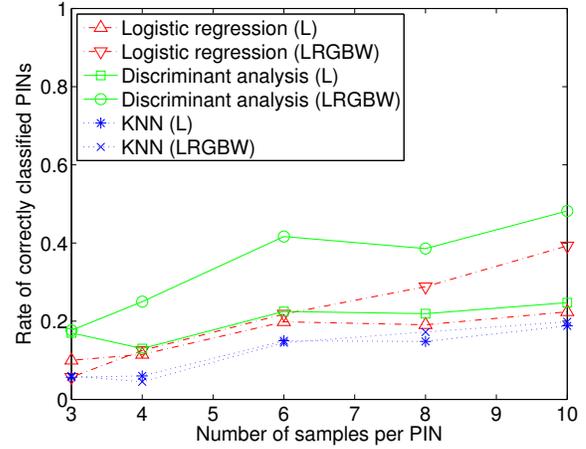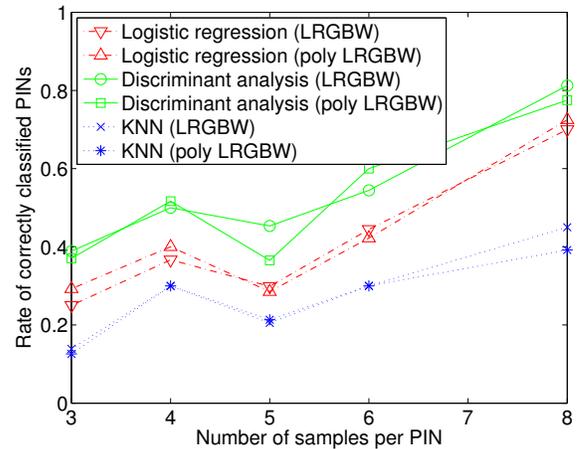
Figure 8: Average rate of correctly classified PINs over multiple runs with a set of 15 PINs each.

50% seems to be quite moderate. However, the chance of correctly guessing the right PIN from a set of 50 PINs is $\frac{1}{50} = 2\%$. Thus, our attack outperforms random guessing by a factor of 20 to 25.

5.2 Comparison of Feature Vectors

Since we introduced a feature vector which is comprised of degree 3 polynomials fitted through all the sensor values (*e.g.*, lux and RGBW values) during one PIN input, we need to compare the performance of the plain sensor values and the approximated sensor values.

Figure 8 illustrates the average rate of correctly classified PINs for these feature vectors based on a 10-fold cross validation over multiple runs with a set of 15 PINs. We observe that both feature vectors, *i.e.*, the plain sensor values as well as the approximated sensor values, yield a similar performance for the different classification algorithms. Again, the *linear discriminant analysis* performs better than the other two classifiers. Based on this observation we only focus on the *linear discriminant analysis* within the following investigations.
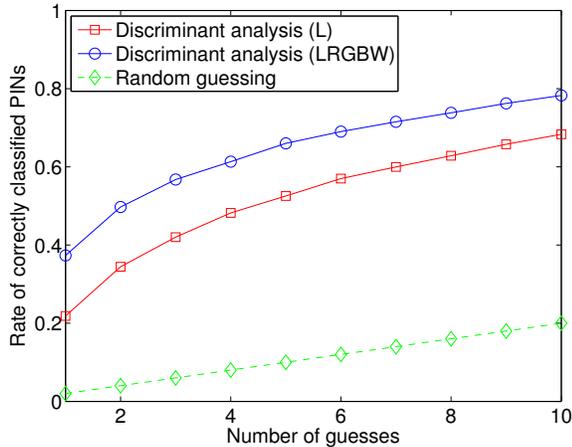
Figure 9: **Average rate of correctly classified PINs for multiple guesses on a set of 50 PINs.**

Figure 10: **Average rate of correctly classified PINs for different input methods on a set of 15 PINs.**

5.3 Guessing PINs Based on their Probability

An interesting approach is to consider the fact that an adversary is able to enter PINs for a specific number of times, *i.e.*, to guess possible PINs according to their probability for being the correct one. In this case the probability of finding the correct PIN increases with every tested PIN. Thus, we instruct the classifier to return a probabilistic ranking of the inferred PINs. Afterwards, we sort the potential PINs according to their probability for being the correct one and illustrate the rate of correctly classified PINs after a specific number of guesses.

Figure 9 shows the average rate of correctly guessing a PIN out of a set of 50 random PINs for a specific number of guesses. The additional information provided by the RGB(W) sensor increases the success rate by about 10 percentage points compared to the *lux* value only feature vector. We also illustrate the success rate if one were trying to guess PINs randomly, which clearly shows the advantage of our attack compared to random guessing.

When comparing our results to the results of Aviv *et al.* [8], we observe that the ambient-light sensor provides results at least as good as those achieved by exploiting the accelerometer sensor. Comparing our results to the work of Simon and Anderson [46], we observe that the ambient-light sensor provides even better results than the approach of exploiting the camera to infer PINs. For instance, Simon and Anderson claim to infer more than 30% of the PINs after two guesses and more than 50% of the PINs after five guesses. In contrast, the ambient-light sensor allows us to infer about 50% of the PINs after two guesses and about 65% of the PINs after five guesses.

The presented results indicate that guessing PINs according to their probability for being the correct one provides an effective means of finding the correct one. On average we are able to determine the correct PIN with a probability of 80% when considering the ten most probable PINs. In contrast, guessing PINs randomly from a set of 50 PINs would result in a success rate of 20% after ten guesses.

5.4 Impact of Different Input Methods

Since users employ different input methods, we also investigate the impact of an input method on the classification
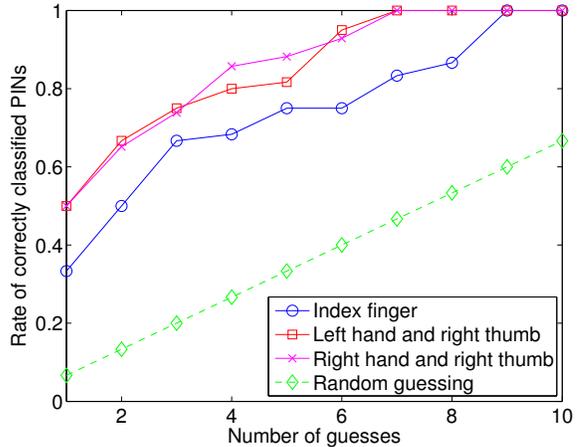
Table 2: **Input methods of three users**

User	Input method
User 1	Left hand and index finger
User 2	Right hand and right thumb
User 3	Left hand and index finger

rate. To this end, we compare the three input methods observed during our experiments:

1. Holding the device in one hand and using the thumb of the same hand to operate it.

2. Holding the device in one hand and operating it with the thumb of the other hand.

3. Holding the device in one hand and using the index finger of the other hand to operate it.

Figure 10 illustrates the average rate of correctly classified PINs for the three different input methods after guessing a specific number of the most probable PINs. The plot is based on a 10-fold cross validation considering a *linear discriminant analysis* on the LRGBW values. Each of the underlying sets of PINs had a cardinality of 15. According to this plot the two input methods involving the thumb, *i.e.*, left hand and right thumb as well as right hand and right thumb, seem to be more vulnerable to this attack than the one with the index finger. This is due to the fact that the mobile device usually undergoes only minor movements when the index finger is used, because one hand is solely used to hold the mobile device. However, this is not entirely correct because also for the input method involving the left hand and the right thumb one hand is solely used to hold the mobile device.

To gain further insight into factors affecting the rate of correctly classified PIN inputs, we compare the provided data of three different users. The corresponding input methods of these users are illustrated in Table 2. All three users entered 30×3 PINs, *i.e.*, each of the 30 PINs was entered 3 times, within the same room under the same environmental conditions regarding the ambient light.
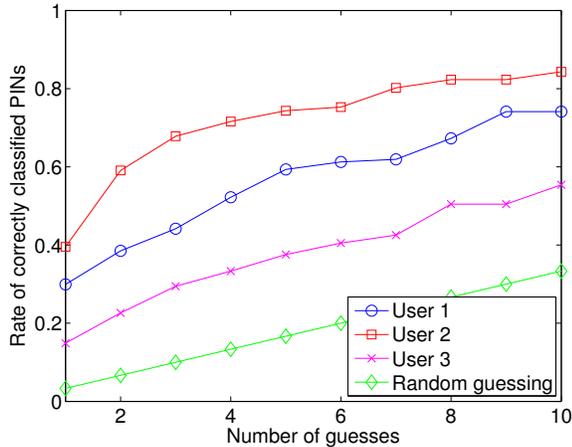
Figure 11: Average rate of correctly classified PINs for three specific users on a set of 30 PINs.

Figure 12: Average rate of correctly classified PINs for different sampling rates based on a set of 15 PINs.

Figure 11 illustrates the result of the 10-fold cross validation of the three data sets provided by *User 1*, *User 2*, and *User 3*, respectively. The input method of *User 2* seems to leak the most information. This appears to be due to the fact that she rested her upper arm against her upper body in a relaxed manner and operated the smartphone in a very comfortable way. *User 1* placed her elbows on her knees and also operated the device in a very relaxed way. In contrast, *User 3* tightly pressed her upper arm against the upper body and held the device very firmly in her hand while entering the digits with the index finger. Based on the investigations of these three users, we observe that though the input method seems to have an impact on the classification rate, a general statement regarding the security or insecurity of a specific input method is difficult to make. Nevertheless, we claim that the tighter and more firmly one holds the device, the less information is leaked. To put it more generally, the more movements the mobile device undergoes during the operation, the more information is leaked.

5.5 Impact of the Sampling Frequency

As outlined in Section 2, the ambient-light sensor can be configured to operate with different sampling frequencies. With a sampling frequency of 750 Hz the *Samsung Galaxy SIII* provides an immense number of measurement samples per second, far more than is necessary for a successful attack. In fact, most of our attacks were performed with a sampling frequency between 5 and 50 Hz, though we performed successful attacks with all possible sampling frequencies.

Figure 12 indicates that the lowest sampling frequency supported by our device (5 Hz) is actually enough to perform the presented attack. The plot illustrates that the performance does not even decrease when sampling with the lowest frequency of 5 Hz.

Demonstrating that an attack can be performed with a low sampling frequency is more important than demonstrating its success on a high sampling frequency. This is due to the fact that many devices are already equipped with such a sensor and upcoming mobile devices are most likely equipped with even more powerful RGB(W) sensors, *i.e.*, supporting even higher sampling frequencies and resolutions.

Though we did not observe any problems with the lowest
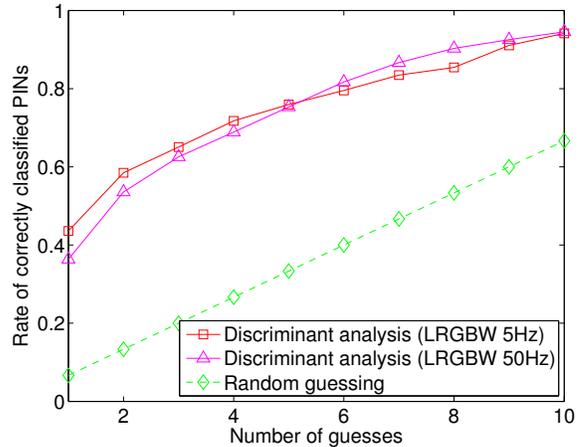
sampling frequency on the *Samsung Galaxy SIII*, we note that even lower sampling frequencies potentially might prevent this attack. This is due to the fact that for sampling frequencies below 5 Hz too few measurement samples might be gathered if one enters the PIN too fast.

6. LIMITATIONS

The presented attack also has some limitations. First, our model does not consider mistyped PINs. If a user deletes an incorrect digit and enters the correct digit afterwards, we are not able to infer the correct PIN anymore. Nevertheless, related attacks also do not cover this case. Second, we did not evaluate our attack outside under sunny, foggy or cloudy light conditions. However, we evaluated the attack in a room—where the only light source was a window— during different daytimes. Furthermore, the data used to train the machine-learning algorithm is currently gathered in the same environment as the actual data that should be classified. Nevertheless, our attack scenario is based on the assumption that people use their smartphone while watching TV, while waiting in a waiting room, or during a train ride, which seems to be a reasonable assumption. Future work, however, might investigate whether a more general model can be established in order to decouple the training phase from the actual attack phase. For instance, a "calibration" phase might be used to determine the overall light conditions in the user's environment to speed-up the training phase. Third, due to the fact that the presented attack is based on the ambient-light sensor, it does not work in case the user operates the mobile device in a completely dark environment. Though, a completely dark environment also prevents an attack that exploits the camera [46].

7. ANALYSIS OF COUNTERMEASURES

In this section, we discuss potential mitigation techniques to prevent the exploitation of sensor information.

7.1 UI and API Modifications

Aviv *et al.* [8] argue that an effective security mechanism would be to prevent untrusted applications from accessing

motion sensors, at least during the input of sensitive information. However, the crucial question is: *When is an input considered as being sensitive?* Clearly, the input of an authentication PIN or a password represents a sensitive input. But what about the input while writing an e-mail or the data entered in forms on websites? Perhaps the sensors should be disabled as soon as the virtual keyboard is displayed? However, this renders applications that rely on these sensors completely useless as is also stated in [8].

Owusu *et al.* [39] suggest to vary the keyboard layout by rearranging the buttons on the virtual keyboard prior to every sensitive input. The drawback of such countermeasures is the dramatic decrease of usability. While this might be applicable for a PIN pad, it would definitely undermine the usability of the QWERTY keyboard layout.

Limiting the resolution and the sampling frequency of the sensor might be another possible countermeasure. For instance, the ambient-light sensor is currently used to adapt the screen brightness. For such an application a more coarse resolution as well as a lower sampling frequency should be sufficient. We are not aware of any scenario that requires a sampling frequency of $50\,Hz$ or even $750\,Hz$. As we have shown in this paper, even the lowest sampling frequency of $5\,Hz$ on the *Samsung Galaxy SIII* does not prevent the presented attack. However, reducing the sampling frequency to 1-$2\,Hz$ should suffice for the task of adapting the screen brightness and to prevent the presented attack. Furthermore, since only the OS performs the task of adapting the screen brightness, access to this sensor might be restricted to the OS exclusively. While these countermeasures might be quite effective, they limit the functionality of specific applications, *e.g.*, games that heavily rely on the usage of sensors.

7.2 Rethinking the Permission Model

A quite sophisticated countermeasure might be a fine-grained permission system in mobile operating systems[1]. Felt *et al.* [24] evaluated different permission-granting mechanisms, including automatic granting, trusted UIs (cf. Roesner *et al.* [41]), runtime-consent dialogs, and install-time warnings. After considering their model we conclude that an effective countermeasure would be an install-time warning, *i.e.*, to pause the installation process and to explicitly inform or warn the user about the requested permission.

Specific risks that might arise from permissions must be communicated to the user, as has also been reported by Felt *et al.* [25], especially since the manifold permissions confuse many users [25, 32]. However, excessive warnings lose effectiveness and might cause users to ignore these warnings again. In order to overcome this problem, Peng *et al.* [40] suggest to rank applications according to their risks rather than using a binary decision for the classification of vulnerable applications. This ranking decision is based on the requested permissions of applications that are known to be malware. Based on such a ranking users might make more deliberate decisions regarding the installation of applications. However, such rankings are only applicable if the motion sensors as well as the ambient-light sensor are considered within the permission system of the OS, which calls for security-specific actions of operating-system developers.

---

[1]Recently, Google took the opposite approach and "simplified the permission system" [28]. Now users are informed about rather coarse-grained permission groups during the installation of applications.

7.3 Application Analysis

A similar approach might be achieved by extending *App-Guard* [11]—an application to enforce security policies—to support the detection of possibly unwanted sensor accesses by malicious applications. *AppGuard* could scan applications during the installation and inform the user about sensor accesses that potentially leak sensitive information. Other malware-analysis applications such as static analyzers, *e.g.*, *Stowaway* [23] or *AndroidLeaks* [27], or applications like *VirusTotal* [50] could also be extended to check applications for malicious sensor accesses.

7.4 User Behavior

Another possible countermeasure might be to enter sensitive data only in environments without any light source and with the index finger or a stylus pen. However, in this case other sensors, *e.g.*, motion sensors, might still be exploitable. So, for really sensitive data, the user might cover the ambient-light sensor as well as the camera, *e.g.*, with her finger, and place the mobile device on a flat surface while providing the input.

Encouraging users to choose longer PINs and passwords might also increase the security [46]. However, some applications do not even allow PINs with more than 4–5 digits. Again, the drawback of such countermeasures is the decreasing usability. Last but not least, awareness must be raised amongst users. Applications should not be able to gather information without knowledge of the user and users must be encouraged to be wary when installing applications, which is why studies like this one are essential.

8. RELATED WORK

Side-channel attacks on input devices in general have been shown to occur both directly [6, 34, 48] as well as indirectly, *e.g.*, through oily residues on the touchscreen [7], or through reflections of monitors [9, 10]. More specific investigations of threats represented by sensor-based side channels on mobile devices include, for instance, the work of Cai *et al.* [20]. They raised the awareness regarding the camera, the microphone, and the GPS signal in modern smartphones. In 2011, Cai and Chen [18] claimed to be the first to show the privacy risk of motion sensors utilized in smartphones. Han *et al.* [29] used the accelerometer sensor to infer the location of the device owner, even with the location-based services deactivated. In 2012, Owusu *et al.* [39] employed the accelerometer sensor to infer passwords entered on touchscreens. Miluzzo *et al.* [37] and Xu *et al.* [51] made use of the accelerometer and motion sensors in general to infer the locations of taps on touchscreens. Recently, attacks have been presented to infer unknown PIN inputs based on a set of learned PINs. For instance, Aviv *et al.* [8] demonstrated such an attack by exploiting the accelerometer and Simon and Anderson [46] demonstrated such an attack by exploiting the camera and the microphone. We briefly compare related attacks to our attack within the following paragraphs.

Comparison. Compared to attacks based on motion sensors, a major advantage of the ambient-light sensor is the power consumption. The method *Sensor.getPower()* returns the power in mA used by the corresponding sensor while in use [5]. On our *Samsung Galaxy SIII* this method returns $0.2\,mA$ for the ambient-light sensor, $0.23\,mA$ for the

Table 3: Comparison of related work targeting a set of 50 PINs

	Aviv *et al.* [8]	Simon and Anderson [46]	Ours
Sensor	Accelerometer	Camera	Ambient-light sensor
Permissions	Internet	Camera, Internet	Internet
Training	Independent of user/location	For each user individually	For each user/environment individually
Drawbacks	-	LED and shutter sound on non-rooted devices	Does not work in completely dark environments
Input method	No constraints	Thumb of holding hand	No constraints
Accuracy	43% within 5 guesses	50% within 5 guesses	65% within 5 guesses

accelerometer, and $6.1\,mA$ for the gyroscope. Hence, the ambient-light sensor consumes a factor of 30 less power than the gyroscope, which means that our attack is less prone to gain the user's attention through battery drainage.

Table 3 provides a comprehensive comparison of related work that is similar to ours, *i.e.*, attacks targeting a specific set of PINs. All attacks assume to have Internet access in order to transfer the gathered data to a powerful server that performs the classification. However, as argued in Section 3, the *Internet* permission can be gained rather easily without raising the user's suspicion.

The main drawback of our attack is that users are not allowed to walk around while entering the PINs because currently the data is only exploitable for one specific environment. Hence, the data cannot be reused for multiple attacks as in case of Aviv *et al.* [8]. Nevertheless, their results indicate a rather low success rate of 20% within 5 guesses when inferring PINs that were entered while walking around. Furthermore, their attack also works in completely dark environments which does not hold for the ambient-light sensor. However, our results indicate a slightly better accuracy of 65% within 5 guesses when inferring unknown PINs.

The work of Simon and Anderson [46] additionally requires the *Camera* permission which potentially gains the user's suspicion. In addition, their attack must deal with the problem of audio-visual feedback, *e.g.*, the shutter sound or the LED, while capturing the required data. Compared to their work we do not restrict our study to specific input methods as long as the user is holding the device while operating it. Furthermore, they need to transfer image data to the server, which cannot be represented as compact as sensor values. While this first investigation showed that the ambient-light sensor might not be superior to existing sensor-based attacks, we have shown that this sensor leaks sensitive information that can be exploited effectively.

9. CONCLUSION AND FUTURE WORK

In this paper, we investigated a new type of side channel which is based on the ambient-light sensor employed in today's mobile devices. To the best of our knowledge, we are the first to show that the ambient-light sensor indeed leaks sensitive information about the user's input on the touchscreen. We developed a proof-of-concept application that allows us to infer unknown PINs, when given a set of already known PINs. This application clearly demonstrates that the leaked information represents a viable side channel for compromising the user's privacy and security. Since no specific permission is required to access this sensor, an adversary is able collect sensitive inputs from mobile-device owners without raising any suspicion and, thus, remain unrecognizable.

Any mobile device equipped with an ambient-light sensor that provides a sufficiently high sampling rate and resolution

can be attacked. Our investigation showed that state-of-the-art smartphones—from all major device manufacturers—include an ambient-light sensor and might be at risk.

There are many different dimensions among the information leakage of sensors which can be investigated. Examples of factors affecting the applicability and performance of sensor-based side-channel attacks are, for instance, the sensor hardware itself, the screen dimension, the device orientation, the keyboard layout, the user's behavior and typing style, the different classification algorithms and the employed feature vectors, the actual environment (*e.g.*, indoor and outdoor), etc. These examples demonstrate that further research is necessary to evaluate the performance of all available sensors under different settings in order to determine the best sensor for a specific attack scenario. For instance, Cai and Chen [19] as well as Al-Haiqi *et al.* [1] performed a comparison of the gyroscope and the accelerometer for multiple users on different devices in different settings to determine the best motion-based sensor. It remains an open question whether a combination of motion sensors with the ambient-light sensor leads to better performances than reported so far. However, the intention of this work was to provide a first feasibility study. By comparing three classification algorithms, different feature vectors, different input methods, different environments, and the impact of different sampling frequencies, we showed that the ambient-light sensor provides a viable side channel.

Related work on the leakage of motion sensors claimed that access to these sensors must be limited with a fine-grained permission system. As shown in this work, access to the ambient-light sensor must also be protected through such a permission system and, thus, operating-system developers need to deal with this problem. Probably even more important is the fact that users need to be aware of such threats and be wary when installing applications that require permissions to access sensors. This is actually why studies like this one are essential.

Acknowledgment

References

[1] A. Al-Haiqi, M. Ismail, and R. Nordin. Keystrokes Inference Attack on Android: A Comparative Evaluation of Sensors and Their Fusion. *Journal of ICT Research and Applications*, 7(2):117–136, 2013.

[2] E. Alpaydin. *Introduction to Machine Learning (Adaptive Computation and Machine Learning*. The MIT Press, 2nd edition, 2010.

[3] Android Developers. Android KitKat. `https://developer.android.com/about/versions/kitkat.html`.

[4] Android Developers. Camera. `http://developer.android.com/reference/android/hardware/Camera.html`.

[5] Android Developers. Sensors Overview. `http://developer.android.com/guide/topics/sensors/sensors_overview.html`.

[6] D. Asonov and R. Agrawal. Keyboard Acoustic Emanations. In *IEEE Symposium on Security and Privacy*, pages 3–11, 2004.

[7] A. J. Aviv, K. Gibson, E. Mossop, M. Blaze, and J. M. Smith. Smudge Attacks on Smartphone Touch Screens. In *USENIX Conference on Offensive Technologies (WOOT)*, pages 1–7, 2010.

[8] A. J. Aviv, B. Sapp, M. Blaze, and J. M. Smith. Practicality of Accelerometer Side Channels on Smartphones. In *Annual Computer Security Applications Conference (ACSAC)*, pages 41–50, 2012.

[9] M. Backes, T. Chen, M. Dürmuth, H. P. A. Lensch, and M. Welk. Tempest in a Teapot: Compromising Reflections Revisited. In *IEEE Symposium on Security and Privacy*, pages 315–327, 2009.

[10] M. Backes, M. Dürmuth, and D. Unruh. Compromising Reflections-or-How to Read LCD Monitors around the Corner. In *IEEE Symposium on Security and Privacy*, pages 158–169, 2008.

[11] M. Backes, S. Gerling, C. Hammer, M. Maffei, and P. von Styp-Rekowsky. AppGuard - Enforcing User Requirements on Android Apps. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 543–548, 2013.

[12] Barclays PLC. Mobile Banking Services. `http://www.barclays.co.uk/Mobile/BarclaysPingit/P1242603570446`.

[13] M. Becher, F. C. Freiling, J. Hoffmann, T. Holz, S. Uellenbeck, and C. Wolf. Mobile Security Catching Up? Revealing the Nuts and Bolts of the Security of Mobile Devices. In *IEEE Symposium on Security and Privacy*, pages 96–111, 2011.

[14] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[15] BlackBerry. BlackBerry Balance. `http://us.blackberry.com/business/software/blackberry-balance.html`.

[16] BlackBerry. BlackBerry Runtime 10.0. `http://developer.blackberry.com/android/`.

[17] J. Bonneau, S. Preibusch, and R. Anderson. A Birthday Present Every Eleven Wallets? The Security of Customer-Chosen Banking PINs. In *Financial Cryptography (FC)*, pages 25–40, 2012.

[18] L. Cai and H. Chen. TouchLogger: Inferring Keystrokes on Touch Screen from Smartphone Motion. In *USENIX Conference on Hot Topics in Security (HotSec)*, 2011.

[19] L. Cai and H. Chen. On the Practicality of Motion Based Keystroke Inference Attack. In *Trust and Trustworthy Computing (TRUST)*, pages 273–290, 2012.

[20] L. Cai, S. Machiraju, and H. Chen. Defending Against Sensor-Sniffing Attacks on Mobile Phones. In *ACM SIGCOMM Workshop on Networking, Systems, and Applications for Mobile Handhelds (MobiHeld)*, pages 31–36, 2009.

[21] DoMobile. AppLock. `https://play.google.com/store/apps/details?id=com.domobile.applock`.

[22] Evernote Corporation. Evernote. `https://play.google.com/store/apps/details?id=com.evernote`.

[23] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner. Android Permissions Demystified. In *ACM Conference on Computer and Communications Security (CCS)*, pages 627–638, 2011.

[24] A. P. Felt, S. Egelman, M. Finifter, D. Akhawe, and D. Wagner. How to Ask for Permission. In *USENIX Conference on Hot Topics in Security (HotSec)*, 2012.

[25] A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner. Android Permissions: User Attention, Comprehension, and Behavior. In *Symposium On Usable Privacy and Security (SOUPS)*, page 3, 2012.

[26] Gary Mazo. How to uncover and use the hidden Service menu on the Galaxy S3. `http://www.androidcentral.com/how-uncover-and-use-hidden-service-menu-galaxy-s3`.

[27] C. Gibler, J. Crussell, J. Erickson, and H. Chen. AndroidLeaks: Automatically Detecting Potential Privacy Leaks in Android Applications on a Large Scale. In *Trust and Trustworthy Computing (TRUST)*, pages 291–307, 2012.

[28] Google. Simplified permissions on Google Play. `https://support.google.com/googleplay/answer/6014972`, 2014.

[29] J. Han, E. Owusu, L. T. Nguyen, A. Perrig, and J. Zhang. ACComplice: Location Inference using Accelerometers on Smartphones. In *Communication Systems and Networks (COMSNETS)*, pages 1–9, 2012.

[30] M. Jakobsson and D. Liu. Your Password is Your New PIN. In *Mobile Authentication*, SpringerBriefs in Computer Science, pages 25–36. Springer New York, 2013.

[31] KeepSafe. KeepSafe. `https://play.google.com/store/apps/details?id=com.kii.safe`.

[32] P. G. Kelley, S. Consolvo, L. F. Cranor, J. Jung, N. M. Sadeh, and D. Wetherall. A Conundrum of Permissions: Installing Applications on an Android Smartphone. In *Financial Cryptography Workshops*, pages 68–79, 2012.

[33] P. C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Advances in Cryptology (CRYPTO)*, pages 104–113, 1996.

[34] D. F. Kune and Y. Kim. Timing Attacks on PIN Input Devices. In *ACM Conference on Computer and Communications Security (CCS)*, pages 678–680, 2010.

[35] P. Marquardt, A. Verma, H. Carter, and P. Traynor. (sp)iPhone: Decoding Vibrations From Nearby Keyboards Using Mobile Phone Accelerometers. In *ACM Conference on Computer and Communications Security (CCS)*, pages 551–562, 2011.

[36] MathWorks. Statistics Toolbox. `http://www.mathworks.com/products/statistics/`.

[37] E. Miluzzo, A. Varshavsky, S. Balakrishnan, and R. R. Choudhury. Tapprints: Your Finger Taps Have Fingerprints. In *Mobile Systems, Applications, and Services (MobiSys)*, pages 323–336, 2012.

[38] NAB. NAB. `https://play.google.com/store/apps/details?id=au.com.nab.mobile`.

[39] E. Owusu, J. Han, S. Das, A. Perrig, and J. Zhang. ACCessory: Password Inference using Accelerometers on Smartphones. In *Mobile Computing Systems and Applications (HotMobile)*, page 9, 2012.

[40] H. Peng, C. S. Gates, B. P. Sarma, N. Li, Y. Qi, R. Potharaju, C. Nita-Rotaru, and I. Molloy. Using Probabilistic Generative Models for Ranking Risks of Android Apps. In *ACM Conference on Computer and Communications Security (CCS)*, pages 241–252, 2012.

[41] F. Roesner, T. Kohno, A. Moshchuk, B. Parno, H. J. Wang, and C. Cowan. User-Driven Access Control: Rethinking Permission Granting in Modern Operating Systems. In *IEEE Symposium on Security and Privacy*, pages 224–238, 2012.

[42] Samsung. Samsung Galaxy S4 Specifications. `http://www.samsung.com/global/microsite/galaxys4/`.

[43] Samsung. Samsung Galaxy SIII Specifications. `http://www.samsung.com/global/galaxys3/specifications.html`.

[44] Samsung. Samsung KNOX. `http://www.samsung.com/global/business/mobile/solution/security/samsung-knox`.

[45] Samsung. What You May Not Know About GALAXY S4. `http://global.samsungtomorrow.com/?p=23610`.

[46] L. Simon and R. Anderson. PIN Skimmer: Inferring PINs Through The Camera and Microphone. In *ACM Workshop on Security and Privacy in Smartphones & Mobile Devices (SPSM)*, pages 67–78, 2013.

[47] Solirify. Math Trainer. `https://play.google.com/store/apps/details?id=com.solirify.mathgame`.

[48] D. X. Song, D. Wagner, and X. Tian. Timing Analysis of Keystrokes and Timing Attacks on SSH. In *USENIX Security Symposium*, 2001.

[49] UK Office of Communications. Communications Market Report 2013. `http://media.ofcom.org.uk/2013/08/01/the-reinvention-of-the-1950s-living-room-2/`, 2013.

[50] VirusTotal. VirusTotal. `https://play.google.com/store/apps/details?id=com.virustotal`.

[51] Z. Xu, K. Bai, and S. Zhu. TapLogger: Inferring User Inputs On Smartphone Touchscreens Using On-Board Motion Sensors. In *Conference on Security and Privacy in Wireless and Mobile Networks (WISEC)*, pages 113–124. ACM, 2012.