Keystroke Inference Using Ambient Light Sensor on Wrist-Wearables: A Feasibility Study

Mohd Sabra, Anindya Maiti
Wichita State University
{masabra, axmaiti}@shockers.wichita.edu

Murtuza Jadliwala
University of Texas at San Antonio
murtuza.jadliwala@utsa.edu

ABSTRACT

Many modern wrist-wearables, such as smartwatches and fitness trackers, are equipped with ambient light sensors that are able to capture the surrounding light levels. While an ambient light sensor is intended to make applications environment-aware, malicious applications can potentially misuse it to infer private information pertaining the wearer. Moreover, such an attack vector is hard to mitigate because the ambient light sensor is a part of the *zero-permission* sensor suite on most wearable platforms, i.e., any on-device application can access these sensors without requiring explicit user-level permissions. In this paper, we study the feasibility of how a malicious smartwatch application can leverage on ambient light sensor data to infer sensitive information about the wearer, specifically keystrokes typed by the wearer on an ATM keypad. While there are multiple previous works that target motion sensor data on wrist-wearables to infer keystrokes, we study the feasibility of how a similar attack can be conducted using an ambient light sensor. The characteristic differences between motion and light data, and how they are impacted during the keystroke activity, implies that existing inference frameworks that rely on motion data cannot be directly employed in this case. As a result, we design a new ambient light based keystroke inference framework which models the varying intensities of light on and around an ATM keypad to infer keystrokes. Our evaluation results indicate that an inference attack on keystrokes is moderately feasible, even with a coarse-grained ambient light sensor found on many low-cost wrist-wearables.

1. INTRODUCTION

The use of modern wrist-wearable devices, such as smartwatches and fitness trackers, is increasingly becoming the norm rather than a luxury. In fact, the number of smartwatch owners have doubled every year in the recent past [4]. These wrist-wearables are worn by users for extended period of time, and are constantly collecting sensory data for context-aware applications such as fitness tracking and health monitoring. Inside these wrist-wearables are a number of different sensors, each sensing a different type of data related to the wearers and their surroundings. A subset of these sensors is referred to as *zero-permission* sensors due to the lack of any access control mechanism imposed on them by the mobile operating system (OS). In other words, any application installed on the wearable device can access these sensors without any user-level or OS-level access control. Examples of zero-permission sensors on Android Wear and Apple watchOS, the two most popular wearable OS, includes accelerometer, gyroscope, magnetometer, barometer, and ambient light sensor.

Unfortunately, applications with malicious intent can exploit these zero-permission sensors as a side-channel to infer sensitive information related to the users, without their knowledge or consent. For example, Michalevsky et al. [15] were able to detect and recover audio speeches by using the gyroscope, a sensor which is generally used to detect changes in device orientation. Maiti et al. [11] used the gyroscope sensor on wrist-wearables to infer combination keys of mechanical padlocks and safes. Another interesting side-channel attack on personal data is targeted on keystrokes typed by users. Keystrokes can include very sensitive information, such as passwords, PIN codes, credit card numbers, etc. The consequences can be grave if malicious applications running on users' wearable device can infer such private information. In the recent past, multiple works demonstrated the possibility of keystroke inference attacks using wrist-wearables [20, 12, 17, 9, 10, 13], primarily using different motion sensors.

In this work, we investigate the feasibility of a new kind of keystroke inference attack with wrist-wearables, using the ambient light sensor (instead of motion data) as a side-channel. Unlike multi-dimensional motion sensors which can be measured very precisely and at a high sampling rate, the ambient light sensor found on modern wrist-wearables such as smartwatches (for example, the Sony SmartWatch 3 and Motorola Moto 360) are relatively coarse-grained and only one-dimensional. As a result, solely employing ambient light sensor data to infer keystrokes on a two-dimensional keypad is challenging. In this paper, we propose a new smart watch ambient light sensor based keystroke inference framework (Section 4.1) and investigate how different light sources around the keypad can be utilized to determine the approximate location of keystrokes. Our initial evaluation results show moderate keystroke inference accuracy (Section 5), which we further improve by adding timing analysis within the framework. We show that by employing such a

WearSys'18, June 10, 2018, Munich, Germany

DOI: https://doi.org/10.1145/3211960.3211973

timing analysis on the keypad layout, it is possible to infer sequences of keystrokes with relatively higher accuracy.

2. RELATED WORK

Side-Channel Keystroke Inference Attacks. Keystroke inference attacks using side-channels is a vastly researched area. Vuagnoux et al. [19] demonstrated how keyboards (wired or wireless) themselves can be leaking typed information in form of electromagnetic emanations. Asonov et al. [1] and Berger et al. [3] were able to recover keystrokes on a nearby computer keyboard, using the acoustic emanations released by key presses. Similar keystroke inference attacks can be carried out using surface vibration emanations generated during keystrokes [14, 2]. In the recent past, multiple works demonstrated the possibility of keystroke inference attacks using motion sensors on wrist-wearables. These works can be categorized based on the target system, such as physical QWERTY keyboards [20, 9, 10], numeric ATM keypads [20, 9], and touch-screen mobile keypads [12, 13, 17]. For brevity, we target PIN codes typed on a numeric ATM keypad (Figure 1) in this feasibility study.

Ambient Light Sensor as a Side-Channel. The ambient-light sensor has been exploited as a side-channel by a small number of previous works. Spreitzer et al. [18] used the ambient-light sensor on a smartphone to infer PINs enter on the smartphone keypad. This was achieved by analyzing variations in the ambient-light sensor data due to the minor tilts and turns of mobile devices caused by tapping at different locations (keys) on the screen. In contrast, if we use the smartwatch ambient-light sensor for inference, the sensor will be present on the users' wrist rather than on the typing device. As a result, the keystroke inference framework becomes significantly different for smartwatches, compared to Spreitzer et al.'s work. Holmes et al. [7] studied how the light emanations from the computer screen can be used to determine the distance of the wrist from the screen, and hence the keys being pressed. However, Holmes et al. did not present a complete inference framework to recover keystrokes, especially from ambiguous keys with similar light readings. We make advancement in this direction by designing a supervised machine learning-based inference framework to recover the PIN codes typed on an ATM keypad. To further improve the prediction accuracy, we utilize timing analysis based on the distance between keys on an ATM keypad.

3. ADVERSARY MODEL

Our goal in this paper is to analyze the feasibility of inferring keystrokes on ATM keypads by solely exploiting the ambient light sensor data on-board the target users' smartwatch (or any wrist-wearable used by the user that has an on-board ambient light sensor). Our work is motivated by the intuitive observation that the intensity of a light signal, similar to other electromagnetic signals, attenuates with distance. Thus, pressing (or tapping) keys closer to a light source would result in a higher light intensity on the target users' wrist compared to keys away from the light source. We also observe that most enclosed (or partially enclosed) environments containing systems that require human-computer interaction (such as an ATM booth) may contain multiple artificial sources of light in order to give the human user a clear view of the interaction interface

Figure 1: An exemplary setup where a person is typing on an ATM keypad, while wearing a smartwatch on left hand. A similar setup is used in our experiments.

(e.g., keypad). This can result in a non-uniform distribution of light intensities over the target interface (keypad). The question that we want to answer in this work is whether these observations can be exploited to infer user keystrokes on a numeric keypad such as an ATM keypad (Figure 1). We make the following additional assumptions about the adversarial capabilities required for carrying out the inference attack.

To execute the inference attack, the adversary will first have to install an eavesdropping application on target users' wrist-wearable, which exfiltrates ambient light sensors data (measured in luminance or *lux*) to the adversary. This can be achieved by social engineering attacks such as baiting, pretexting, phishing, and etc. [8], or by tricking the user in to installing a Trojan application masqueraded as a legitimate application [5]. In addition to exfiltrating light sensor data, we also assume that the adversarial application detects keystroke events using the on-device microphone, and records their timestamps along with the light sensor data. This is critical in the attack framework, because the adversary must accurately identify when a keystroke event took place before trying to classify it[1]. Although critical, keystroke event detection using audio data is not required in our inference framework; any other technique or attack vector to infer a keystroke event can also be alternatively employed. However, several previous research efforts [3, 14, 10] have employed a similar audio-based keystroke event detection mechanism. Due to the impracticality of an on-screen keyboard on the small smartwatch screens, an adversarial smartwatch application can seek access to the microphone in order to support voice commands or dictation, without raising suspicions, which can be later used for keystroke event detection.

The exfiltration of ambient light sensor data and keystroke timestamps can be done using a covert-channel, such as embedded between other legitimate Internet traffic, so as to avoid detection. In this feasibility study, we also assume that the adversary can collect training data in the target environment, which can be used to create generalized prediction models (as outlined in detail in the later sections). We also assume that an adversary can set up additional ambient light sources to improve the accuracy of the inference attack. These two assumptions are practical as an environ-

[1]The acoustic data is not used for keystroke inference, but rather just to identify keystroke events.

ment such as an ATM machine are generally open and physically accessible to everyone, including the attacker. Some ATM keypads are equipped with a cover to protect against visual eavesdropping attacks. Our attack is not preventable by such designs due to the fact that a smartwatch (and light sensor on it) is worn on the wrist, which usually remains outside the cover.

4. THE ATTACK FRAMEWORK

In this section, we present our attack framework for inferring keystrokes from raw ambient light sensor data, and how it can be improved with supplementary timing analysis based on the distance between keys on a standard ATM keypad layout.

4.1 Keystroke Inference Using Ambient Light

Our keystroke inference framework consists of a learning phase, followed by an attack phase. In the learning phase, the adversary learns the light intensities observed during typing, and applies them in an attack phase to infer a target user's keystrokes.

4.1.1 Learning Phase

This phase focuses on the construction of a predictive model that would be used during the attack phase. The model comprises of three major steps: (i) *data collection*, (ii) *feature extraction* and (iii) *supervised learning*. Before starting the learning phase, the adversary must determine the experimental parameters related to a target user. This includes determining the hand the smartwatch is worn (either by observing the target user or having prior knowledge), and the ATM used.

Data Collection: In this part of the learning phase, the adversary collects labeled and time-stamped light intensities (*lux* values) observed for all ten numeric keys of the ATM keypad. To remove any sequential bias, this training data is collected in a random typing order. In our experiments, the training data was collected using a custom Android Wear application, and one of our authors (acting as the adversary) typed 1200 random digits which uniformly covered all the ten numeric keys of the keypad. As the test users were unknown during our training data collection, our evaluation results represent the worst-case scenario. If the adversary has additional prior information about the target user (such as height and typing style), a more personalized training data can be collected, and utilized in the attack phase.

Feature Extraction: An ideal lighting scenario occurs when every key on the keypad has a unique *lux* value, in which case the *lux* values can be used as the only distinguishing feature. However, such an ideal scenario is hard to achieve in a real-world setting. As a result, we extracted the following features for training and testing our prediction model: *lux* value after a keystroke, *lux* value before a keystroke, maximum *lux* value between consecutive keystrokes, average *lux* value between consecutive keystrokes, and median *lux* value between consecutive keystrokes.

Supervised Learning: We model the ambient light keystroke inference problem as a multi-class classification problem. In the learning phase, labeled feature vectors are used to train five different classifiers appropriate for our problem (based on the properties of our features): Support Vector Machines, Decision Trees, Random Forests, Navies Bayes, and k-Nearest Neighbors.

4.1.2 Attack Phase

Data Collection: This is similar to the data collection during the learning phase, however here the *lux* values and keystroke timestamps are collected from the target users when they are entering their PIN code on the ATM. In our experiment, the ground-truth was also recorded from the ATM keypad. In a real attack, the ground-truth would not be accessible, but we collect ground-truth in order to evaluate the accuracy of our inference framework.

Feature Extraction: The same set of features, as used in the learning phase, are extracted from the raw ambient light sensor data that is recorded by the malicious application during the attack phase.

Keystroke Classification: This is the final phase of the inference framework, where the adversary uses the trained classifiers from the training phase, in order to predict the unknown target keystrokes, and thus infer the PIN code entered.

4.2 Reducing Search Space

Prediction accuracy using ambient light sensor alone may not be sufficient to accurately infer a sequence of keys (such as a PIN code). Therefore we propose the use of timing analysis to significantly reduce the search space within which individual keystrokes of a PIN code are classified, thus improving the overall accuracy of inferring the entire PIN code.

The ATM keypad layout is publicly known, and as the distance between buttons is fixed, the adversary can calculate relative distances between the keys. In the attack phase, the adversary can leverage on the fact that users have to press the "Enter" key after typing the PIN code. By backtracking the distances traveled from the "Enter" key to the first keystroke, the adversary can significantly reduce the search space and simplify the prediction/classification stage of the inference framework. However, the adversary cannot infer the distances traveled between keys just from the ambient light sensor. But the adversary can utilize the timestamps already collected for individual keystroke, to indirectly calculate the distance between keystrokes. In our experiments, we found out that almost all participants typed with a close to constant speed (keys traveled per unit time, not keys pressed per unit time). Using the formula $Distance = Time \times Speed$, where speed is assumed to be constant and *relative distances* between numbers are known (Figure 3), the adversary can deduce a relationship between distance between keys and time. To understand how the distance-time relation can be useful, let us assume a 3-digit numeric PIN code "$a - b - c$". If the user takes double the time between keystrokes $a \rightarrow b$ compared to $b \rightarrow c$, it implies that a and b are twice as far apart as b and c are. As a result, the adversary can infer distance traveled between keys

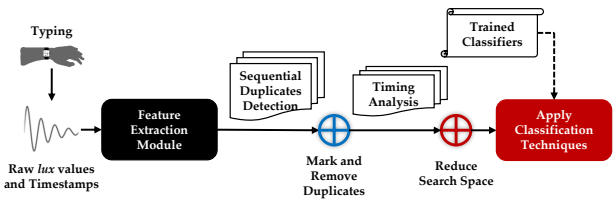

Figure 2: An overview of the proposed keystroke inference framework.

	1	2	3	4	5	6	7	8	9	0	Enter
1	0	1	2	1	1.4	2.25	2	2.25	2.83	3.1	4.25
2		0	1	1.4	1	1.5	2.25	2	2.25	3	3.5
3			0	2.25	1.4	1	2.83	2.25	2	3.1	3.1
4				0	1	2	1	1.4	1.4	2.25	3.5
5					0	1	1.4	1	1.4	2	2.83
6						0	2.25	1.4	1	2.25	2.83
7							0	1	2	1.4	3.1
8								0	1	1	2.25
9									0	1.4	1.4
0										0	2

Figure 3: Euclidean distances between any two key pairs (numeric and the "Enter" keys). Measured in number of keys traveled.

	1	1.4	2	2.25	2.82	3	3.1	3.5	4.25
1	1	1.4	2	2.25	2.82	3	3.1	3.5	4.25
1.4		1	1.428571	1.607143	2.014286	2.142857	2.214286	2.5	3.035714
2			1	1.125	1.41	1.5	1.55	1.75	2.125
2.25				1	1.253333	1.333333	1.377778	1.555556	1.888889
2.82					1	1.06383	1.099291	1.241135	1.507092
3						1	1.033333	1.166667	1.416667
3.1							1	1.129032	1.370968

Figure 4: Proportions of Euclidean distances between any two key pairs (numeric and the "Enter" keys). Same colored regions are so close to each other that we assume any key pairs that fall inside that range would be hard to distinguish. White regions are very distinctive, allowing an adversary to uniquely identify the underlying key pair.

based on timestamps, and use a backtracking algorithm (Algorithm 2) to significantly reduce the search space. While some users type faster than others (and vice versa), it does not affect the deduction of how many keys were traveled between consecutive keystrokes of a PIN (+Enter). The reason behind this is that the distances are calculated based on relative time between keystrokes observed within a PIN code, typed by a single user at a constant typing speed.

The distance between any two keys of the targeted ATM keypad can be calculated using the Euclidean geometry [6]. The unit of distance can be represented as keys traveled, with the minimum travel distance of one key (for example, two adjacent keys). In Figure 3, we see that there are 7 distinct distances possible between any two keys (1, 1.4, 2, 2.25, 2.82, 3, and 3.1) and 9 distinct distances (1, 1.4, 2, 2.25, 2.82, 3, 3.1, 3.5, and 4.25) between the final PIN code key and the "Enter" key. With this information, in Figure 4 we list all the possible proportions between any two consecutive pair of keystrokes ($distance(a \rightarrow b)/distance(b \rightarrow c) = 2$ in the previous example). Algorithm 1 uses time periods between all keystrokes of a PIN (+Enter) to deduce a set of all possible combination of distances. After the adversary calculates all the possible combination of distances, the adversary can backtrack starting from the "Enter" button to

Algorithm 1 Algorithm to find all possible distances based on time between keystrokes (N digits followed by "Enter").

```
TBSK[N] #Array of times between N + 1 sequential keystrokes
TimeProportions = ComputeAllTimeProportions(TBSK)
ShadedSeqRegions =
TimeProportions2Shades(TimeProportions)
possibleDistances[][N] = {∅} #Possible N sequential distances
possibleDistances = CheckPossibilties(ShadedSeqRegions)
return(possibleDistances)

function ComputeAllTimeProportions(TBSK)
    pair = [(, )]
    proportion = []
    for i = 0 to TBSK.size() do
        for j = i + 1 to TBSK.size() do
            pair.append(make_pair(TBSK[i], TBSK[j]))
            proportion.append(TBSK[i]/TBSK[j])
        end for
    end for
    return(pair, proportion)
end function
function TimeProportions2Shades(tp)
    shades[tp.proportions.size()] = []
    values[] #Unique values in Figure 4
    correspondingShades[] #Corresponding shades in Figure 4
    for prop in tp.proportions do
        nearestValue = find_nearest(values, prop)
        index = values.find(nearestValue)
        shades.append(correspondingShades(index))
    end for
    return(tp.pair, shades)
end function
function CheckPossibilties(SSR)
    #Try all possible distance combinations. Add combination as
a possible distance set, if shaded relationship remains valid for
the entire N digits. Return all possible distance sets.
end function
```

create a set of combinations that could be the PIN code, using Figure 4 and Algorithm 2. This process does not attempt to predict the PIN, but significantly reduces the PIN code search space.

5. EVALUATION

In this section, we empirically evaluate the accuracy of our inference framework under different light settings and using different machine learning algorithms. We first evaluate the prediction accuracy of individual keystrokes, followed by an evaluation on prediction accuracy of entire 4-digit PIN codes.

5.1 Experimental Setup

We recruited 14 participants[2] who wore a Sony Smartwatch 3 on their left wrist and typed 40 4-digit test PIN codes on a virtual ATM keypad replicated based on the dimensions of a real ATM keypad (Wincor Nixdorf EPP v5[3]). The 40 test PIN codes were generated randomly to prevent any bias. An Android Wear data collection application was developed to record ambient light sensor data from the Sony Smartwatch 3 (Android Wear 1.5) at a sampling rate of 10-25 Hz. We collected the ambient light sensor reading (lux values) during the experiment, and transferred them to a remote server, which has the required computing power to process the results. To train and test our classifiers, we used SciKit v0.19.1 [16]. Participants were also asked to practice typing their assigned PIN code 3 times immediately before

[2]IRB approval from the authors' institutes obtained before performing the experiments involving human subjects.
[3]https://www.dieboldnixdorfag.com/

Algorithm 2 Algorithm to find all possible PINs based on relative distances between keystrokes.

```
SearchSpace = {∅} #Possible N-digit PINs
for element in possibleDistances do #From Algorithm 1
    DistancesTraveled[N] = element
    ReverseArray(DistancesTraveled[]) # Start from end
    locations[] = {∅}
    locations.append(10) # Start at "Enter" represented by 10
    for distance in DistancesTraveled do
        PossibleSequentialDigits.previous() =
        GetKeysBasedOnDistance(distance, locations)
        locations = PossibleSequentialDigits.LastLocations()
    end for
    SearchSpace.append(PossibleSequentialDigits.
    getPossiblePINCodes())
end for
return(SearchSpace)

function GetKeysBasedOnDistance(dist,loc)
    distanceFromLoc[][] #Values from Figure 3
    newLocations[] = {∅}
    for l in loc do
        newLocations.append(distanceFromLoc[l].find(dist))
    end for
    return(newLocations)
end function
```

starting the data collection. Every participant typed the test PIN codes under the following scenarios using the setup shown in Figure 1:

1. Having two light sources (Lights 2 and 3) active with light intensity of 80% and then 60%.
2. Having all three light sources active with light intensity of 80% and then 60%.

We experimented between different light setups to understand which setup is more suitable for our inference framework. The two primary factors affecting the observed *lux* value on an smartwatch are the distance from the light source(s) and the incident angle. As both of these factors are variable while typing, by using a single light source it is difficult to determine how much impact both of the factors have on observed changes in *lux* value. With multiple lights, the impact of incident angle is significantly reduced, and the impact of distance becomes the major factor affecting the observed *lux* value. With asymmetrically positioned light sources around a keypad, it is possible to achieve unique *lux* values for each key, with minimal impact from changing wrist angle while typing. Therefore we test our inference framework with two and three light sources. The light sources used in our experiment were Phillips Hue[4] smart bulbs rated at 840 lumens; dimmable using the Hue smartphone application.

5.2 Individual Keystroke Classification

Figure 5 shows the prediction accuracy (percentage of correctly classified keystrokes in the testing data) of individual keystrokes under multiple light settings (and intensities), broken down for different classifiers. The first observation we make is that although the prediction accuracies are moderately high for certain classifiers ($> 60\%$ in best case), it is not sufficiently high to accurately infer a 4-digit PIN code. This makes our timing analysis necessary, as it can significantly reduce the search space for individual keystrokes that make up a PIN code, thereby improving the overall classification accuracy. We also observed that for most classifiers,

[4]https://www2.meethue.com/

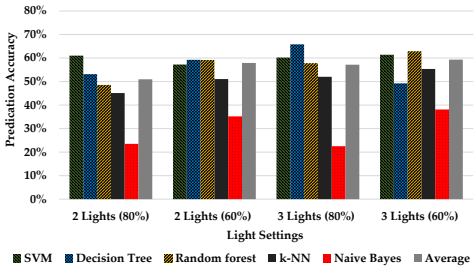

Figure 5: Prediction accuracy of individual keystrokes under multiple light settings (and intensities), using different classifiers.

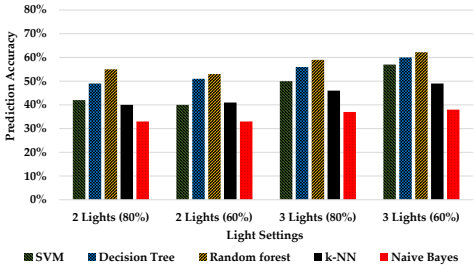

Figure 6: Prediction accuracy of 4-digit PIN codes under multiple light settings (and intensities), with timing analysis applied before predicting individual constituent keys.

the predication accuracy improved when the number of light sources was increased from 2 to 3. Also, the average predication accuracy improved when the intensity of the lights were decreased from 80% to 60% (of full brightness).

After we found that the best prediction accuracy was achieved with three light sources at 60% brightness, we empirically investigated the reasoning behind it. We found that the primary factor affecting individual keystroke prediction is the distinguishability in *lux* value between all keys of the keypad. Using two light sources, it is harder to achieve a light placement where all keys have a unique *lux* value. However, we can achieve uniqueness in *lux* values by placing three bulbs in non-symmetrical positions around the keypad (as shown in Figure 1). In our setup, we found that it is best to place the three lights in such a way that the smartwatch observes *lux* values between 72 and 512 units, which gives enough room for having a unique range of *lux* values for each of the 10 numeric keys. Also, because the light's intensity on the keypad and its distance from the keypad are inversely propositional, bulbs further away can be compensated by making them brighter (and vice versa).

5.3 Timing Analysis and PIN Code Recovery

After applying the timing analysis on 4-digit test PIN codes, the ground-truth PIN code was observed within the reduced search space for 98.3% of test cases. In the best case, time analysis reduced the search space from 10,000 to only 16 (99.84% reduction of search space). We also observed that the search space of PIN codes with a variety of different distances between keys can be more easily reduced, compared to those with a fewer variety of distances.

5.3.1 *Special Case: Sequential Duplicates*

An exception to the above observation are PIN codes with

multiple sequential duplicates, such as $1-1-1-1$ or $5-5-5-5$. Sequential duplicate keystrokes could be found by detecting no *lux* value change between two keystrokes. We had a 100% detection accuracy on duplicate digits typed under 60% light intensity, and a 99% detection accuracy under 80% light intensity. In case a target user has a PIN code of 4 sequential duplicate digits (such as $1-1-1-1$ or $5-5-5-5$), the search space would effectively be reduced by 99.9% (from 10,000 to 10 PIN codes). Moreover, among all 4-digit PIN codes, 27.1% contain at least 2 digits that are in duplicate sequential order (repeated two or more times). Using this knowledge, we can further divide the search space based on whether a test PIN code has a sequential duplicate or not. As a result, the search space of 4-digit PIN codes can be reduced either by 72.9% (if sequential duplicates are detected), or by 27.1% (no sequential duplicates in PIN code).

5.3.2 4 -Digit PIN Code Recovery

After applying the sequential duplicate detection and then reducing the search space with timing analysis, we used the classifiers that were trained previously for individual keystrokes, to predict the most likely PIN code for each of the test PIN code. We achieved more than 62% accuracy in recovering entire 4-digit PIN codes (Figure 6), using the Random Forest classifier.

5.4 Future Work

Our evaluation in this work was mostly a feasibility analysis, which opened possibilities for a more comprehensive work. In future, we plan to address some of the unanswered questions in this work. For example, we will evaluate how our inference framework scales across left and right handed users. We will also evaluate if the trained classifiers can be applied in a cross-device setting, which can be beneficial to an adversary who does not know or possess the same smartwatch as their target.

6. CONCLUSION

We investigated the feasibility of a keystroke inference attack using the ambient light sensor data of wrist-wearables. We proposed a new inference framework that can predict keystrokes based on different ambient luminance around the wrist. We also proposed timing analysis algorithms to significantly reduce the keystroke search space for 4-digit PIN codes. Our evaluation results demonstrated that ambient light sensors on wrist-wearables can in fact be used as a side-channel, with individual keystroke inference accuracy much higher than random guessing. 4-digit PIN code inference was also made more accurate ($> 60\%$) with the help of timing analysis. Our work shows that under appropriate lighting conditions, ambient light sensor on wrist-wearables can be effectively used to infer sensitive private information.

Acknowledgment

This work was supported by the Division of Computer and Network Systems (CNS) of the National Science Foundation (NSF) under award number 1828071 (originally 1523960).

7. REFERENCES

[1] D. Asonov and R. Agrawal. Keyboard Acoustic Emanations. In *IEEE S&P*, 2004.

[2] A. Barisani and D. Bianco. Sniffing Keystrokes with Lasers/Voltmeters. *Black Hat USA*, 2009.

[3] Y. Berger, A. Wool, and A. Yeredor. Dictionary Attacks using Keyboard Acoustic Emanations. In *ACM CCS*, 2006.

[4] Consumer Technology Association. Smartwatch unit sales worldwide from 2014 to 2018 (in millions), 2018.

[5] A. P. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner. A survey of mobile malware in the wild. In *ACM SPSM*, 2011.

[6] H. G. Forder. *The foundations of Euclidean geometry*, volume 10. Dover New York, 1958.

[7] A. Holmes, S. Desai, and A. Nahapetian. Luxleak: capturing computing activity using smart device ambient light sensors. In *ACM Smart Objects*, 2016.

[8] K. Krombholz, H. Hobel, M. Huber, and E. Weippl. Advanced social engineering attacks. *Journal of Information Security and applications*, 22, 2015.

[9] X. Liu, Z. Zhou, W. Diao, Z. Li, and K. Zhang. When good becomes evil: Keystroke inference with smartwatch. In *ACM CCS*, 2015.

[10] A. Maiti, O. Armbruster, M. Jadliwala, and J. He. Smartwatch-based keystroke inference attacks and context-aware protection mechanism. In *ACM AsiaCCS*, 2016.

[11] A. Maiti, R. Heard, M. Sabra, and M. Jadliwala. Towards Inferring Mechanical Lock Combinations using Wrist-Wearables as a Side-Channel. In *ACM WiSec*, 2018.

[12] A. Maiti, M. Jadliwala, J. He, and I. Bilogrevic. (Smart)Watch Your Taps: Side-channel Keystroke Inference Attacks Using Smartwatches. In *ACM ISWC*, 2015.

[13] A. Maiti, M. Jadliwala, J. He, and I. Bilogrevic. Side-Channel Inference Attacks on Mobile Keypads using Smartwatches. *IEEE Transactions of Mobile Computing*, 2016.

[14] P. Marquardt, A. Verma, H. Carter, and P. Traynor. (sp) iphone: decoding vibrations from nearby keyboards using mobile phone accelerometers. In *ACM CCS*, 2011.

[15] Y. Michalevsky, D. Boneh, and G. Nakibly. Gyrophone: Recognizing Speech from Gyroscope Signals. In *USENIX Security*, 2014.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct), 2011.

[17] A. Sarkisyan, R. Debbiny, and A. Nahapetian. Wristsnoop: Smartphone pins prediction using smartwatch motion sensors. In *IEEE WIFS*, 2015.

[18] R. Spreitzer. Pin skimming: Exploiting the ambient-light sensor in mobile devices. In *ACM SPSM*, 2014.

[19] M. Vuagnoux and S. Pasini. Compromising Electromagnetic Emanations of Wired and Wireless Keyboards. In *USENIX Security*, 2009.

[20] H. Wang, T. T.-T. Lai, and R. Roy Choudhury. Mole: Motion leaks through smartwatch sensors. In *ACM MobiCom*, 2015.