There Goes Your PIN

Exploiting Smartphone Sensor Fusion Under Single and Cross User Setting

David Berend[1,2], Bernhard Jungk[3,⋆] and Shivam Bhasin[1]

[1] Physical Analysis and Cryptographic Engineering, Temasek Laboratories
Nanyang Technological University, Singapore
[2] University of Applied Sciences Wiesbaden, Rüsselsheim, Germany
[3] Cyber and Information Security Group, Fraunhofer Singapore
email: david.berend@mine-it.eu, bernhard.jungk@fraunhofer.sg, sbhasin@ntu.edu.sg

Abstract. A range of zero-permission sensors are found in modern smartphones to enhance user experience. These sensors can lead to unintentional leakage of user private data. In this paper, we combine leakage from a pool of zero-permission sensors, to reconstruct user's secret PIN. By harvesting the power of machine learning algorithms, we show a practical attack on the full four-digit PIN space. Able to classify all 10,000 PIN combinations, results show up to 83.7% success within 20 tries in a single user setting. Latest previous work demonstrated 74% success on a reduced space of 50 chosen PINs, where we report 99.5% success with a single try in a similar setting. Moreover, we extend the PIN recovery attack from a single user to a cross-user scenario. Firstly, we show that by training on several users, the PIN recovery success can be boosted, when a target user is part of the training pool. On the other hand, PIN recovery is still possible when training pool is mutually exclusive to the target user, albeit with low success rate.

Keywords: Smartphones, PIN, Sensor, Machine Learning

1 Introduction

Since the introduction of the first generation of iPhones by Apple in 2007, the market for smartphones quickly expanded. Smartphones have drastically changed the way we interact with our mobile devices and range of applications. Evolved from a simple communication device to camera and music player, a modern smartphone has the capabilities to monitor the user's health, movement, personal finances, habits and a lot of other data. Due to the sensitivity of some of the processed data and the widespread usage of smartphones, with over 30.5% of the global population using a smartphone in 2017 [14], the devices have become interesting targets for attackers. Two major operating system, Android and iOS, are dominating the market with a market share of 81.7% and 17.9%

⋆ This research was conducted when author was at Temasek Laboratories.

respectively [13]. Our proof of concept attack is targeting Android devices and hence, our attack applies to a large portion of all smartphones.

Each generation of smartphones implements new innovative features to maintain or increase the customer base. New technologies, such as health trackers, augmented or virtual reality, require more and more computing power and an increasing number and quality of physical sensors, to advance the user experience. However, the security countermeasures and the privacy protections implemented in smartphones are not improved at the same pace.

One problem is that the pool of sensors embedded in smartphones may unintentionally leak sensitive information. For example, sensors that are accessible without user permissions, so-called zero-permission sensors, may be exploited by an attacker, without the knowledge of the user. The scenario is analogous to other widely known side-channel attacks which exploit unintentional (physical) leakage. For example, this can be used to extract cryptographic keys [7]. In our case, the unintentional leakage from sensors reveals privacy-related information about the user, such as personal identification number (PIN) or movement patterns.

The exploitation of zero-permission sensors of smartphones, resulting in security and privacy breaches, has been investigated in many previous publications [1, 10, 11, 5, 18, 15, 4, 3, 2, 12, 8, 9]. These attacks include behavioral profiling, geo localisation, and PIN recovery. The first two applications rely on the relative ease to detect significant movements of the user and the smartphone.

The latter attack – recovering security PINs – is more difficult, because the exploited movements are less pronounced and hence, harder to classify correctly. Previous work on the PIN extraction secnario [17, 3, 2, 12, 8] also deals with the problem of (4-digit) PIN recovery. The two publications [3, 17] try to use a single digit classifier and then to recombine the individual digits to form a PIN. The other publications [2, 12, 8] apply the classification to complete PINs but restrict their training and test data to a set of only 50 PINs. The work of Mehrnezhad et al. [8] is the latest in this line of work and reports the highest success rate to date. It uses machine learning to achieve 74% PIN recovery success from a set of 50 chosen PINs.

In this paper, we propose a novel single digit classification methodology for PIN recovery. Our method fuses information from all the available sensor to boost the PIN recovery success. If applied to the setting of only 50 PINs, we report a PIN recovery success of 99.5%.

Next, we apply the PIN recovery problem in a generalized setting, which can classify all $10,000$ possible combinations. In this setting, we report a recovery success rate up to 83.7% in good measurement settings. The classification algorithms are able to easily weigh the importance of each sensor in PIN recovery and allow high recovery success. Since the methodology works on a single digit, it is scalable to PINs longer than 4-digits.

Moreover, all the previous work were investigated under single user setting, i.e. training and attack were done on the same user. In this work, we extend the PIN recovery to a cross-user setting. The main advantage of the cross-user

setting is, that it allows the training to be done by different users than the attack. This increases the chances of successful attacks in practical scenarios, where a malicious application can learn from several users to recover PIN of a new user. Also, the attacker may be able to train the machine learning algorithm prior to planting the malicious code on the smartphone, reducing the user interaction to a minimum, thus making such attacks much less detectable and hence, more successful.

The rest of the paper is organised as follows. Sec. 2 discusses the attack scenario. Sec. 3 details the proposed methodology and experimental results for PIN recovery in a single-user setting. This is extended to cross-user settings in Sec. 4. Finally, conclusions are drawn in Sec. 5.

2 Scenario

In the presented attack, we extract a PIN entered by the user using the touch-screen interface of a smartphone. Such PINs are often used to protect sensitive applications, such as unlocking screen or an online banking app and therefore, they are assets, which can be interesting for an attacker. Usually, a PIN consists of several digits, where each digit is chosen from the interval $(0, 9)$.

The user enters the PIN using a standard numerical PIN pad keyboard, where each possible number (from the interval $(0, 9)$) is at a distinct location on the touch screen of the smartphone. Therefore, the user has to move his hand and fingers to reach for the number he wants to enter next. For most users, this in turn also causes the smartphone to tilt, to rotate or to move in some other more complex way. The smartphone will also move due to the pressure of the press itself.

The assumption of all attacks exploiting the sensor information is that the movement of both hand and smartphone is specific to each number. Therefore, it might be possible to use the smartphone's sensors to learn and detect which number has been pressed. This information can then be used to reconstruct the PIN successfully with a high probability from the sensor data alone.

With this assumption in mind, the attack scenario proceeds as follows:

1. Injection of malicious code on the victim's smartphone, using a malicious app or using the browser app of the phone with JavaScript. The malicious code then performs the following actions.
2. Sampling of the sensor data during a training period with predetermined PINs.
3. Online or offline profiling of the acquired training data.
4. Sampling of the sensor data during the attack phase with unknown PINs.
5. Classification of the sensor data acquired during the attack phase based on the profile generated in the training phase.
6. Ranking of the results after classification.

For a concrete attack, the attacker has to make a few design decisions. Firstly, he has to decide which sensors to use. Secondly, he has to select appropriate algo-

rithms for the profiling and the classification. Thirdly, he has to decide whether the profiling and classification steps are performed offline or online.

While the first two points are crucial to the attack and determine the success rate of the PIN extraction to a large degree, the third decision influences the practicability of the attack in another major way. In an online setting, the profiling of the attacked user's input is performed on the smartphone directly. Thus, the resulting profile can only be trained for one individual user. One might expect this profile to be a better fit for the exact user, compared to a profile generated from multiple users. However, this is not necessarily true and has to be supported by experimental evidence.

Another potential benefit is, that very little communication over the network link has to be performed and thus, the potential to be detected is reduced. The major drawback is that the user has to enter enough training data to produce a large enough data set to successfully generate a good profile. To this end, the user can be persuaded using different methods, for example using a manipulated casual game [16].

In contrast, an offline attack may collect data from multiple persons and multiple devices. If it is possible to use the machine learning algorithm to benefit from this data, the available training data is much easier to generate and possibly independent of a specific user. When strictly following an online vs. offline approach, the sensor data to be classified during the attack phase is also sent to the server.

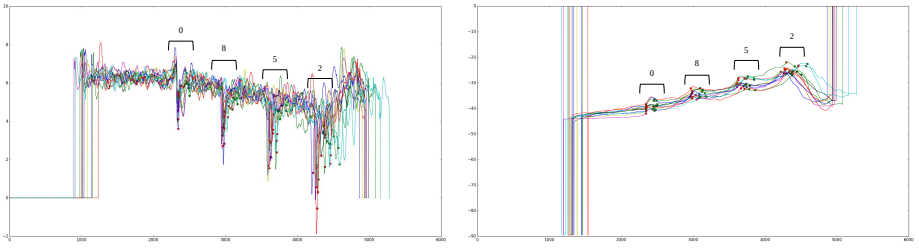
However, also offline-online hybrid attacks are possible. Such attacks do not collect any training data from the same set of users that are to be attacked. Instead, the attacker recruits others to help him with the training and only the final trained classification algorithm is pushed to the attacked user. Only the final classification result after the attack has been performed is then sent back to the attacker.

3 Proposed PIN Recovery Attack Methodology

3.1 Single Digit Classification Approach

For a four-digit PIN classification, two approaches exist. The first classifies a complete PIN consisting of multiple digits. The second identifies individual digits first and then combines the classified digits to complete PINs. For both approaches, the underlying data is a recorded data stream of sensor data over the whole duration of the PIN input (Figure 1).

The first approach is to use the complete data stream and to associate it to the combination of keys during the training phase. This implies, that all 10,000 combinations of a four-digit password have to be trained to be able to later correctly classify all possible combinations, because each class has its own unique characteristic and if the training data does not train for some of the combinations, it will be unable to classify this combination correctly. Previous works using such technique have focused on a subset of 50 classes [2, 8].



(a) Accelerometer (b) Rotation Vector Sensor

Fig. 1. 10 records where the keys 0-8-5-2 were pressed for two different sensors. Red dots signalize the time where the display was touched, while green dots signal the release.

The second approach is to identify each digit in the sensor data individually. For this, we first have to cut out the corresponding part from the relevant sensor data and then classify the digit presses individually. The benefit is, that only ten possible classes exist, each representing a key on the numerical PIN pad.

The underlying data is identical to the data used in the first approach. However, we first split it into parts corresponding to the individual presses and then train and classify using only parts of the sensor data. This results in a training and classification of only ten digits instead of 10,000 combinations, which is easier to achieve with a high success rate for individual digits. Also, if we can identify individual digits of a PIN with a very high success rate, the overall success rate for a four-digit PIN will also be high.

One of the challenges is to identify the right timing of a key press and then to split the data accordingly. Therefore, we identify distinguished peaks in the data stream, which indicate a press.

Comparing both approaches, ours is much more flexible, achieves a high success rate for any PIN length and also reduces the amount of training data significantly at the same time. Since we train only individual digits, we can apply our attack to any length of the PIN, with higher (shorter PIN) or lower success rate (longer PIN), while the other classification schemes working on complete PINs would not be able to classify any PINs shorter or longer than four digits. For the classification of more PINs and also of different PIN lengths, we do not have to change the amount of training data, nor the nature of it. In contrast, the full-PIN classification schemes from previous work usually have to be trained with each PIN.

3.2 Setup

For our proof of concept attack, we developed an application to sampling the sensor data and a server-side module for aggregation, classification and evaluation. The devices in the experiment should represent the technology of the

current state of smartphone adopters. Therefore, we have chosen an LG Nexus 5 for the common technology standard.

For measuring the sensors during a PIN entry, we developed an android application, which runs in the foreground and stores the records in external storage. The sampling of each zero-permission sensor can be turned on or off individually. The app also provides a numerical keypad, which appears once the measurement process starts. The layout and position of the numerical keypad are identical to the one provided by the Android OS. The layout of the developed app is shown in Fig. 2.

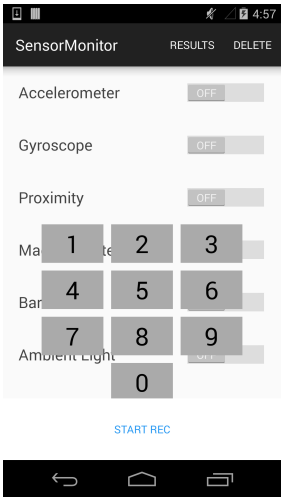


Fig. 2. Layout of the app. In the foreground user can type in a PIN. In the background user set the relevant sensors, before measuring.

A candidate was asked to interact with the application in a consistent natural position. Five times 70 chosen four-digit PINs were entered for training. The combination of the four digits plays a crucial role. To recreate a training base of data, which qualifies to be practically relevant, each key must be entered from all different positions. This is achieved by entering all possible ten keys before and after each key. In addition to the training data, 50 randomly generated pins were entered for validating the algorithm.

3.3 Implementation

Measurement A PIN can be entered in a smartphone in multiple physical ways. Relevant parameters are body position, holding type, speed, left or right hand or both. For this experiment the user sat on a chair, holding the phone in the right hand, typing the passwords with the hand's thumb at a consistent speed. Due to the general experimental limitations on the variation of experimental conditions,

we focused on one behavioral composition for input, as the aggregation and classification is the main contribution of this work.

We have tested the contribution of all available zero-permission sensors for the purpose of PIN recovery. Accelerometer, gyroscope, magnetometer, proximity, barometer, ambient light and rotation vector sensor are available. However, we excluded the barometer and proximity sensor, due to the low sampling frequency (0,5 - 4 Hz), which caused high inaccuracy and hence, were not providing any significant signal. An example measurement for the PIN `0852` (entered 10 times) is shown for accelerometer and gyroscope in Fig. 1. By a simple visual inspection, the digits are shown to have distinguishable features. In the following, we take the help of classifying algorithms to distinguish PINs from different sensor measurements in a systematic manner.

During the measurements, all registered sensors fire an *onSensorChange* event through an Android hardware interface. Each event holds the values of a sensor's dimensions. These are saved into a sensor related array of data entries.

The same happens for key presses. Each time a key is pressed, a *touchevent* triggers a tuple of $(timeoftouch, timeofrelease, keylabel)$ to be saved into the current record. A record holds four presses, one for each key of the PIN. Once a password is successfully entered, the record is saved. The final object consists of all sensor related data entries and presses. The data structure storing all this data is depicted in Fig. 3). It is stored locally as *JSON*-string [6], before the object is sent to the server side.

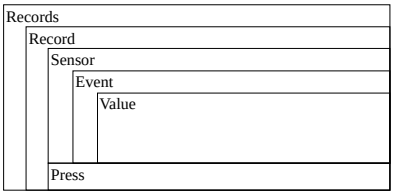


Fig. 3. The data structure of the measured data stored on user's device.

Preprocessing In the measurement step, records are created, where each record r has a timestamp for the start r_{start} and stop r_{stop} time. During recording, activated sensors fire events at their individual frequency. Each sensor holds a list of its events, saved as $(time, value)$ pairs for each dimension. While recording, a candidate performs the PIN input. Each press is saved with key label, as well as the touch and release time of the candidate's thumb on the screen.

Now, the data must be transformed into a comparable state. This means that each record starts at the same time and at the same value. Furthermore, all

sensors require a value at every given point of time. We set the smallest instance of time to a millisecond, to be equal to the timestamp format.

To start off, we calculate the duration of each record r

$$r_{duration} = r_{stop} - r_{start} \tag{1}$$

Then, we replace r_{stop} by $r_{duration}$ and set r_{start} to 0 later in the process.

Currently, each sensor has a list of $(time, value)$ pairs for each dimension. The goal however, is to have a value for each millisecond. Therefore, we transform the $(time, value)$ pairs into an array, which holds a value for each millisecond. First, we drop the beginning four data pairs, to eliminate initial sensor inaccuracy. Then, an array with the size of $r_{duration}$ is instantiated for each dimension. For each millisecond (index of array) it is checked if the dimension's $(time, value)$ pairs have an entry e to address the value to the current index of $e_{currentTime} - r_{start}$. If the sensor didn't fire at that point in time, the previous value is taken.

The final step is to set the first value (denoted by v_{init}) of all time arrays to zero and readjust all other values relative to the adjustment of the first one. Sensors have shown, that noise is very high when activating measurement. Therefore, the first data point would be an insufficient choice as initial orientation v_{init}. Instead, we could improve accuracy, calculating the average value over the first 300 milliseconds of values v_i, where $n\#v_i$ and $v_i \neq 0$.

$$v_{init} = \frac{1}{n} * \sum_{i=0}^{300} v_i \tag{2}$$

v_{init} is then used to calculate the absolute value of the difference to other values v_i of the array.

$$v_{i\ normalized} = |v_i - v_{init}| \tag{3}$$

At this point, the records are comparable.

Classification Classification requires the data to be preprocessed and to be in a comparable state. This allows to fully classify a four-digit password, where each press is classified individually. Therefore, the goal is to characterize a key by the information from a known press so the key can be recognized by the raw motion data afterward.

Due to the relevance of the key being known, a training phase must take place. The data for the training phase consists of many press objects, which holds information about each sensor and its dimensions at every millisecond, during the press duration. To characterize the keys, features are extracted. These are statistical attributes, such as min, max, mean, median, standard deviation and skewness. The attributes are then associated with a known key, which was recorded during measurement. As each dimension of a sensor holds a data stream of values, the stream serves as extraction source. Therefore, the more sensors are taken into consideration, the more features exist. All features are extracted in

the same order of each press and then associated to the key by index in another array, called targets. Finally, the training concludes with a two-dimensional array X, where each row holds an array of all attributes a of each dimension d of each sensor s during a press. For each press, the originally pressed key is stored in a second array y, at the identical index as the array of press attributes in X. The data structure is presented in Table 1. Samples and targets are then fitted onto the classification algorithm.

Table 1. The data structure of classification data. A two-dimensional sample array X and a corresponding target array y, where $k \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

	X	**y**
press	$s_1d_1a_1$ $s_1d_1a_2$ ... $s_1d_2a_1$ $s_1d_2a_2$ ... $s_2d_1a_1$ $s_2d_1a_2$ ...	*key*
p_1	[...]	k
p_2	[...]	k
...	[...]	...

To test the algorithm, a record consisting of four presses serves as input. The four presses undergo the same feature extraction process, to match the same data format, the algorithm has learned. Each press is classified in adhering order. The algorithm returns for each one of them a probability for each key. Finally, all key combinations are permuted, while multiplying their associated probability at each position. The combination with the highest product is the first guess, the one with the second highest, the second guess, and so on.

The primarily used classification algorithm is a multi-layer perceptron (MLP) neural network. The network possesses two middle layers. The first includes as many nodes as input features exist. The second as much nodes as classes. Through this architecture, the features become weighted more accurate and enter a pooling layer afterwards, to granulate the information. Backpropagation optimizes with a limited memory Broyden Fletcher Goldfarb Shanno (L-BFGS) algorithm, which can handle relatively small amounts of data well.

For a comprehensive analysis, of the state of the art classification algorithms, we examine how they perform and if they qualify for scalability. 500 random training samples of the training data of three candidates were added to a training set. Then randomly generated testing data ($30 - 50$ four-digit combinations entered by each candidate) was classified by the fitted algorithm. Afterward, another 500 training samples were added to the training set and fitted onto the algorithm again. The same testing took place. This process was repeated until the training set reached a total size of 2500 samples. For the classification, four algorithms were tested, namely MLP neural network, Random Forest, k - Nearest Neighbour and Gaussian naive Bayes classifier. At the first iteration with only 500 training samples, Random Forest and MLP neural network achieved the highest classification success, while Gaussian Naive Bayes classifier and k - Nearest Neighbour performed 10% and 15% worse (figure 4). Over the course of

all five iterations, MLP neural networks outperformed Random Forest by 4% and began to stagnate, as the number of samples for testing was fixed and assessed relevant information almost maximized. K - Nearest Neighbour and Gaussian Naive Bayes Classifier performed almost identical at a steady linear growth rate of 2% - 5%, although k - Nearest Neighbour tend to perform better when more samples would had existed. However, both algorithms performed 25% - 28% worse than MLP neural networks.

Concluding the results, MLP neural network classification was able to achieve the most success and showed the highest growth rate. Therefore, we use it as classification algorithm to be a valuable option for scalability. If a fast classification is required, Random Forest does not perform much worse. In extremely small training data scenario, it might even surpass MLP neural network due to its simplicity.

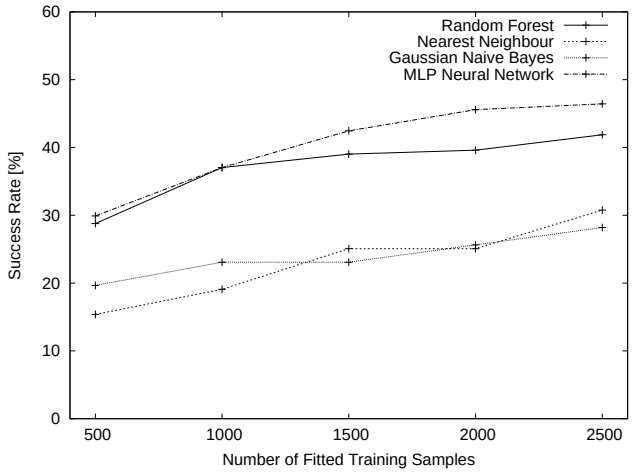


Fig. 4. Success of classification algorithms fitted with 500 to 2500 training samples and tested with 116 test samples in each iteration.

3.4 Evaluation

In this section, we present the results for applying the attack on the data of the best performing candidate. First, the success of individual sensors is examined. Next, further optimizations are investigated in order to maximise the PIN retrieval success. The tested optimizations are sensor fusion and thumb's movement towards and away from a key. As previously shown that MLP performs best when dealing with the full PIN search space, i.e. 10,000 for a four-digit PIN, we use it for the further evaluation. For the rest of the paper, we limit to 20 tries, unless specified.

Individual Sensor Success During a key press, all sensors were measured at their individual highest possible frequency. To evaluate which sensor data contains the most unique and distinguishable key related information, the classification procedure was performed for each sensor. In figure 5 the individual sensor success is presented. To establish the success rate, the individual training data of the most accurate candidate was fitted onto the MLP neural networks and validated with the candidate's testing data. Overall, the accelerometer performed the best with a success rate of 63%, followed by the rotation vector sensor, magnetometer and gyroscope (28%, 20% and 5% respectively). The ambient light sensor provided almost no information. This shows, that sensors with high frequency, perform better. The proximity sensor, as well as the barometer are not listed, as the frequency is too low (on average one data point per second).

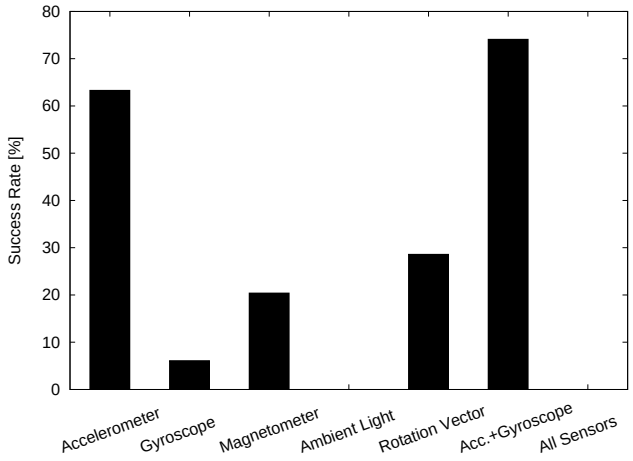


Fig. 5. Sensor individual success, success of the best fusion option & success of all sensors combined together.

Sensor Fusion After investigating the individual sensors, we now explore the advantage of combining information from different sensors. In general, classification algorithms perform better, when more features exist. For this experiment six statistical attributes (section 3.3) are extracted from the data stream of each dimension of each sensor. When all sensors are fused together, 72 features exist (6 attributes for each three dimensions of accelerometer, gyroscope, magnetometer and rotation vector, one dimension of ambient light). However, directly combining all sensors does not give the best results. As shown in the last column of figure 5, the PIN retrieval success is null. To improve the results, all different

combinations were tested. The best fusion resulted in 74.1% by the accelerometer fused with the gyroscope, presented in the sixth column of Fig. 5.

Fewer features achieve better results, as too many features with not enough data cause more noise than relevant information. Additionally, some compositions of sensors work exceptionally well, while others don't. Therefore, the nature of the sensors' dimensions must be examined. The accelerometer measures the phone's acceleration in space. The same space is defined by the magnetometer and the ambient light in their specific manner. The gyroscope measures the acceleration of the phone's movement around itself and the rotation vector sensor aggregates that movement to angles along the same dimensions. Therefore, the sensors can be grouped into space-related and angle-related. When sensors of the same group are used for fusion, redundancies occur and adding them can also cause noise. From the two groups, the best individual sensor performances are from the accelerometer and the rotation vector sensor. However, the fusion of the accelerometer with the gyroscope performed 3-5% better. This is because the gyroscope performed more consistently and fitted better with the accelerometer, as both represent the acceleration along their dimensions.

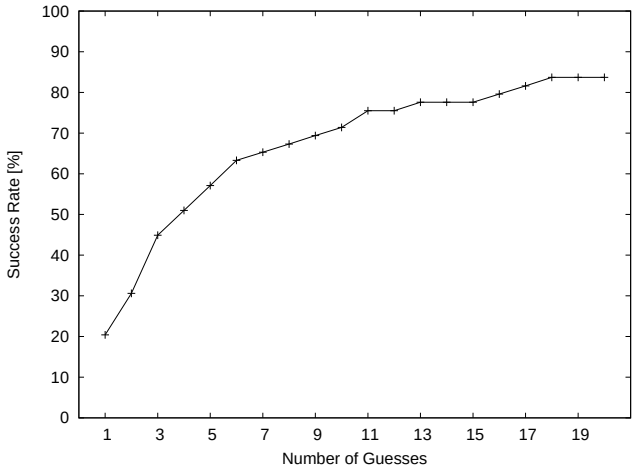


Fig. 6. Best Case Success of getting the password right after 1-20 guesses.

Movement towards and away from a key The fundamental object of which features for classification are extracted is the data stream of sensor data during the candidate's press. The press is cut out from the measurement based on the timestamps of touch and release, provided by the *onTouchEvent* by Android OS. To examine whether additional unique key information lies in the movement towards and from a key press, the time interval is enhanced at both sides by

25 and 50 milliseconds. We use the best performing sensor combination i.e. accelerometer and gyroscope. The data with and without the finger movement information is fed to the MLP for training and validated with the testing data. The results, presented in table 2, show that success reached its maximum of 83.7%, when the press interval was enlarged by $50ms$ at the beginning and $0ms$ at the end, thus improving success by another 12.9%. The evolution of the success from 1^{st} to 20^{th} guesses is shown in Fig. 6. In the present experiment, the results show that the success generally increases when adding information of the movement towards a key, while success decreases when information of the movements from a key is put into the classification data. Moving the thumb towards a key seems therefore to be more key unique, as the same coordinate is aimed at. In the other direction, the movement can go in each direction, which causes success to decrease.

As a general conclusion, we report a PIN retrieval success of 83.7% after combining data from relevant sensors and exploiting information of thumb movement. This, to our knowledge, is the best-achieved figures in the context of PIN classification. The achieved success is reported when applied to complete PIN search space of 10,000 PINs. Moreover, with the underlying agile methodology, the technique can be easily scaled up to longer PINs. Previous works reported success up to 74% when applied to a small subset of chosen 50 PINs. To make a fair comparison, in the following, we scale down our method to 50 chosen PINs and compare the retrieval success.

Table 2. Best sensor fusion (accelerometer and gyroscope), where the press' data stream is enhanced by $0ms$, $25ms$ and $50ms$ at the beginning (rows) and the end (columns)

Towards \ Away	*0ms*	*25ms*	*50ms*
0ms	74.1%	68.7%	71.4%
25ms	78.2%	70.1%	72.1%
50ms	83.7%	76.9%	75.5%

Comparison With Previous Work Classifying each key individually, rather than the full data record including all four digits of the combination resulted in a high success rate. While previous work [8] applied the method on 50 chosen PINs, our previous test was on a complete set of 10,000 PIN. We now apply our method on a small subset of 50 PIN to draw a comparison. The results are MLP-neural network classifier is trained with 50 four-digit-PINs with equally distributed keys. Each press is then classified individually, which results in four arrays of probabilities for a key being pressed. The combinations of all ten keys are then permuted, while the product of the combination related probability is calculated. The combinations, which were not trained in the beginning are then excluded.

The results were worse than the method of Mernezhad et al. [8]. While [8] reports 74% success on first try, our method reports only 60%. It reaches a 100% success after 6 presses. Further analysis of the results indicated that the dataset of 50 PINs is too small for training our MLP neural network, resulting in poor results. To further improve the results, we fall back to Random Forest, which performed close to MLP neural networks in previous experiments. Owing to the simplicity of Random Forest, it performs well with limited training complexity. Repeating the same attack with random forest, we were able to classify the PIN correctly with a 99.5% success within the first guess (refer Fig. 7). Thus, our agile classification method performs 25.5% better than the previous best result of Mernezhad et al. [8].

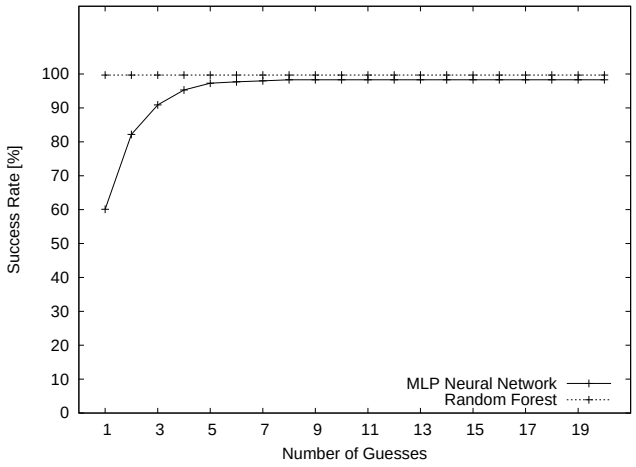


Fig. 7. Success of MLP neural network and Random Forest classifier classifying 50 combinations with multiple guesses.

4 Cross-user Exploitation

A PIN recovery attack on a single user was demonstrated in the previous section. Such analysis, along with other previous works [2, 8], is often limited to training and testing on a single user. This assumption is understandable because every user has a unique way of handling the phone, thus also motivating research for tracking user behaviour [11, 15, 4]. However, we argue that it is possible to identify certain application-specific traits, which can be common across users. In the following, we extend the previous experiments in a cross-user setting. By cross-user setting, we mean that the proposed methodology is first trained with labeled data from a pool of users, and then tested for a target user. Two

scenarios are tested, i.e. when the target user is (a) included in the training pool and (b) excluded from the training pool. This, to our knowledge, is the first work to report cross-user exploits. Due to practical limitation, our proof of concept experiments are limited to three distinct users.

4.1 Attack Scenario & Setup

The attack scenario majorly follows the previous settings. For the training phase, two different approaches can be adopted. The first approach, that we call *inclusive*, PIN entry data from a pool of users is collected in a common training set, and exploited to enhance the attack success rate of any selected user from the pool. However, this would require persuading several users to enter valid training data (on a malicious casual game for example) and to send the data over the network link to the smartphone. In the second approach, which we call *exclusive*, the attacker colludes with other people to generate the training data without direct interaction with the target users. This completely separates the training and testing user set and has practical advantages, since the training can be done without any additional user interaction and also without using the user's network link. If the algorithm is trained sufficiently on the server side, it could be deployed just for attack phase on the target user's device. As a result, the classification takes place in the smartphone and the only data sent to the attacker would be the PIN itself.

Three candidates were asked to interact with the application in a consistent natural position. Each candidate entered five times 70 chosen four-digit password for training, and 30 to 50 randomly generated pins were entered by each candidate for testing. User A (same as in Sec. 3) entered in ideal settings, i.e. sitting on a chair, holding the phone in right hand and typing at consistent speed. User B and C did hold the phone in right hand and entered PIN with a thumb, they were allowed to move and typing speed restrictions were not enforced. In other words, there was a lot of variance in intra-user measurements for user B and C. Since our user set was limited to three, the extra movements allowed variance in quality of training data and testing impact of noise. The following set of experiments were done on the full PIN space of $10,000$ and thus MLP neural network was used for classification and testing.

Individual Benchmarking Before testing the cross-user setting, the users were self-benchmarked. This means, the training and testing phase as proposed in Sec. 3, was repeated for each user individually. Without any information about the movement towards or away from the key, the reported success rate for users A, B and C were 74.1%, 8.9% and 26.3% respectively. The wide variance in success rate depicts the importance of good measurement. Next, we included the information on the movement towards or away from the key. The information on the movement increased the success rate in general for all three users. After applying this preprocessing, the best case success rate increased to 83.7%, 27.8% and 39.5% for users A, B and C respectively. Although, movement information

(towards and away) improved the attack, the best settings vary from user to user. While user A achieved maximum at 50/0 (movement towards/after), user B and achieved it at 50/50 and 25/50 respectively. Thus, the information on the movement is an individual trait. Since movement information generally improved the results, we continue the following cross-user experiments with average case, i.e. 25/25 milliseconds.

4.2 Inclusive Setting

We now investigate the PIN recovery success rate in an inclusive scenario. In this setting, we trained the classification algorithm with data from a pool of users and tested with one user from the pool. To test, we first train our MLP neural network with all the training data from A, B and C (denoted ABC). Next, we test individual users against the combined training set (ABC) and compare the success rate figures against the ones achieved in Sec. 4.1.

Table 3. Success rate for inclusive cross-user exploitation

Testing \ Training	*A*	*B*	*C*	*ABC*
A	70.1%			79.6%
B		16.7%		30.0%
C			17.9%	20.5%

The results are summarised in Tab. 3. It can be observed that the success rate always increases when trained with all the users rather than an individual. For some users, B for example, the success rate increased as much as 80% due to the extra information brought by other users. Although the PIN was entered differently and from different users, the MLP neural network is able to extract relevant features which are not user specific to boost the success rate. The results indicate that, with more users training the system, the success rate of the methodology is expected to scale up.

4.3 Exclusive Setting

In this part, we test a stronger attack scenario. This setting excludes the attacked user from the training set. For example, if user C is attacked, we train the classification algorithm with training data from user A and user B only (denoted AB). In general, we expect the success rate to decline as compared to previous cases, since some personally identifiable traits of the PIN typing style will be missing. However, it gives much power to the attacker, as he can train and attack mutually exclusive users, thus easing the end to end execution of the attack. Note that for a random PIN guess, the success rate is 0.01% for 1^{st} try and 0.2% with 20 tries. In this experiment, our aim is to perform better than random 20 tries.

Table 4. Success rate for exclusive cross-user exploitation

Testing \ Training	*AB*	*BC*	*CA*
A		6.1%	
B			6.7%
C	5.3%		

Tab. 4 shows the experimental results. The results are worse than previous experiments but that is expected as the target user did not train the system. Thus, the individual user traits are not part of the training data. The system tries to guess user agnostic features to identify the entered PIN. Although worse than previous cases, the success rate is around 6%, which is still much better than random guessing. On average, we have a success rate 30× higher. The impact of this result is very serious as it implies that even without the target user training a system, the PIN activity can be guessed.

5 Conclusions

This paper explores the side-channel vulnerability of zero-permission sensors for privacy concerns, through case study of PIN recovery attacks. In this work, we develop a single-digit classification methodology to recover PIN from maliciously captured sensor data. The sensor data from all available sensors are used to train MLP neural network. The best performing sensor is the accelerometer, followed by the rotation vector sensor and the gyroscope. Accelerometer with gyroscope provided the best fusion. Preprocessing with information on movement towards and away from the key improved classifiction, as well. Combining these approaches, we are able to apply our attack on the full space of 10,000 PINs. We report up to 83.7% success in ideal measurement setting. Compared to previous works which worked on 50 chosen PINs, we report a 99.5% success rate (using random forest) against previous 74% [8].

Next, we explore cross-user exploitation. To our knowledge, all previous works, trained and tested on same user. We show that it is possible to boost the success rate of the attack, by training with pool of users and attacking one of them, rather than just training on the target user. The attack even works when target user is not part of the training set, which has much stronger implications.

Limiting the maximum operating frequency of the sensors can reduce the attack feasibility. Alternatively, disabling sensors while sensitive operations like PIN entry can also prevent such attacks. However, these are just temporary fixes, and sensors access in smartphones must be rethought, in general. Finally, implications of zero-permission sensors is much beyond PIN recovery. It can be easily applied to learn user behavior, location etc, completely compromising user privacy.

Acknowledgements

We would like to thank Prof. Adrian Ulges and Prof. Ulrich Schott from Hochschule RheinMain University of Applied Sciences Wiesbaden Rüsselsheim for their valuable advice on classification techniques. We would also like to thank Prof. Steffen Reith from Hochschule RheinMain University of Applied Sciences Wiesbaden Rüsselsheim and Aly Madhavji from INSEAD Business School for their strategic guidance during the development. We thank Prof. Theodoros Evgeniou from INSEAD Business School for the interesting discussions on the topic. Finally, we thank the voluntary participants, who contributed to our data collection.

References

1. Ahmed Al-Haiqi, Mahamod Ismail, and Rosdiadee Nordin. On the best sensor for keystrokes inference attack on android. *Procedia Technology*, 11:989 – 995, 2013.
2. Adam J. Aviv and Jonathan M. Smith. *Side Channels Enabled by Smartphone Interaction*. PhD thesis, Univ. of Pennsylvania, Pennsylvania, 2012.
3. Liang Cai and Hao Chen. Touchlogger: Inferring keystrokes on touch screen from smartphone motion. In *Proc. of the 6th USENIX Conference on Hot Topics in Security*, HotSec'11, pages 9–9, Berkeley, CA, USA, 2011. USENIX Association.
4. Katherine Ellis. *Classifying Human Behaviors, Activities and Contexts from Mobile Sensor Data*. PhD thesis, UC San Diego, San Diego, 2010.
5. Cristiano Giuffrida, Kamil Majdanik, Mauro Conti, and Herbert Bos. *I Sensed It Was You: Authenticating Mobile Users with Sensor-Enhanced Keystroke Dynamics*, pages 92–111. Springer International Publishing, Cham, 2014.
6. Ecma Internaitonal. Json data interchange format, standard ecma-404, October 2013.
7. Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Advances in cryptologyCRYPTO99*, pages 789–789. Springer, 1999.
8. Maryam Mehrnezhad, Ehsan Toreini, Siamak F Shahandashti, and Feng Hao. Stealing pins via mobile sensors: actual risk versus user perception. *International Journal of Information Security*, pages 1–23, 2016.
9. Yan Michalevsky, Gabi Nakibly, Gunaa Arumugam Veerapandian, Dan Boneh, and Gabi Nakibly. Powerspy: Location tracking using mobile device power analysis. In *Proceedings of the 24th USENIX Conference on Security Symposium*, SEC'15, pages 785–800, Berkeley, CA, USA, 2015. USENIX Association.
10. Emmanuel Owusu, Jun Han, Sauvik Das, Adrian Perrig, and Joy Zhang. Accessory: Password inference using accelerometers on smartphones. In *Proc. of HotMobile 2012*, San Diego, USA, February 28–29, 2012.
11. Charissa Ann Ronao and Sung-Bae Cho. Human activity recognition with smartphone sensors using deep learning neural networks. *Expert Systems with Applications*, 59:235 – 244, 2016.
12. Raphael Spreitzer. Pin skimming: Exploiting the ambient-light sensor in mobile devices. In *Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices*, SPSM '14, pages 51–62, New York, NY, USA, 2014. ACM.
13. Statista. Global smartphone sales by operting system since 2009, September 2008.
14. Statista. Number of smartphone users worldwide 2014 - 2020, January 2015.

15. X. Su, H. Tong, and P. Ji. Activity recognition with smartphone sensors. *Tsinghua Science and Technology*, 19(3):235–249, June 2014.
16. Vanja Svajcer. Malicious cloned games attack Google Android Market. *Naked Security: http://nakedsecurity. sophos. com/2011/12/12/malicious-cloned-games-attack-google-android-market*, 2011.
17. Zhi Xu, Kun Bai, and Sencun Zhu. Taplogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors. In *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks*, pages 113–124. ACM, 2012.
18. N. Zheng, K. Bai, H. Huang, and H. Wang. You are how you touch: User verification on smartphones via tapping behaviors. In *2014 IEEE 22nd International Conference on Network Protocols*, pages 221–232, Oct 2014.