IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 7 February 2023, accepted 1 March 2023, date of publication 6 March 2023, date of current version 10 March 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3253288

RESEARCH ARTICLE

Machine Learning for PIN Side-Channel Attacks Based on Smartphone Motion Sensors

MATTEO NERINI[1], (Graduate Student Member, IEEE), ELIA FAVARELLI[2], (Member, IEEE), AND MARCO CHIANI[2], (Fellow, IEEE)

[1]Department of Electrical and Electronic Engineering, Imperial College London, SW7 2AZ London, U.K.
[2]DEI/CNIT, University of Bologna, 40136 Bologna, Italy

Corresponding author: Marco Chiani (marco.chiani@unibo.it)

This work was supported in part by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on Telecommunications of the Future (PE00000001—Program "RESTART").

ABSTRACT Motion sensors are integrated into all mobile devices, providing useful information for a variety of purposes. However, these sensor data can be read by any application and website accessed through a browser, without requiring security permissions. In this paper, we show that information about smartphone movements can lead to the identification of a Personal Identification Number (PIN) typed by the user. To reduce the amount of sniffed data, we use an event-driven approach, where motion sensors are sampled only when a key is pressed. The acquired data are used to train a Machine Learning (ML) algorithm for the classification of the keystrokes in a supervised manner. We also consider that users insert the same PIN each time authentication is required, leading to further side-channel information available to the attacker. Numerical results show the feasibility of PIN cyber-attacks based on motion sensors, with no restrictions on the PIN length and on the possible digit combinations. For example, 4-digit PINs are correctly recognized at the first attempt with an accuracy of 37%, and in five attempts with an accuracy of 63%.

INDEX TERMS Cyber security, machine learning (ML), motion sensors, personal identification number (PIN), smartphone PIN attacks.

I. INTRODUCTION

In recent years, we have observed a massive diffusion of mobile electronic devices, capable of gathering information through built-in sensors such as cameras, microphones, motion sensors, and satellite positioning systems. This information is exploited by applications providing services to the users, while rigorous security policies are applied to protect privacy. However, even in modern devices, the motion data (accelerometer and gyroscope measurements) are not considered critical information. For example, in Android OS they are not subject to any kind of protection [1]. This implies that any application installed on the device, and any web page opened through a browser, is able to freely access these data. Recent studies showed that the knowledge about the movements of mobile devices, which reflect human activities, can lead to possible side-channel cyber-attacks [2], [3]. In particular, it has been proved that the user's arm movements can be classified from the motion information captured by wearable devices such as smartwatches [4], [5], [6], [7]. Furthermore, the activity of smart bracelets allows inferring which words are typed on a keyboard [8], [9], or which Personal Identification Number (PIN) is entered at the Automated Teller Machine (ATM) [10], [11]. Besides, accelerometers and gyroscopes have been used to reveal the location of finger taps on touch screen devices [12], [13].

In this work, we focus on smartphones, the most common category of mobile devices, with the aim to investigate possible cyber-attacks based on motion sensor data. In particular, we want to infer which digits are entered in a digital numeric keypad depending on the smartphone movements captured by the motion sensors. There are several practical ways for an attacker to acquire the motion sensors while a PIN is inserted. The first possibility is given by a malicious application that could be installed on the victim's mobile device. This

The associate editor coordinating the review of this manuscript and approving it for publication was Weiping Ding.

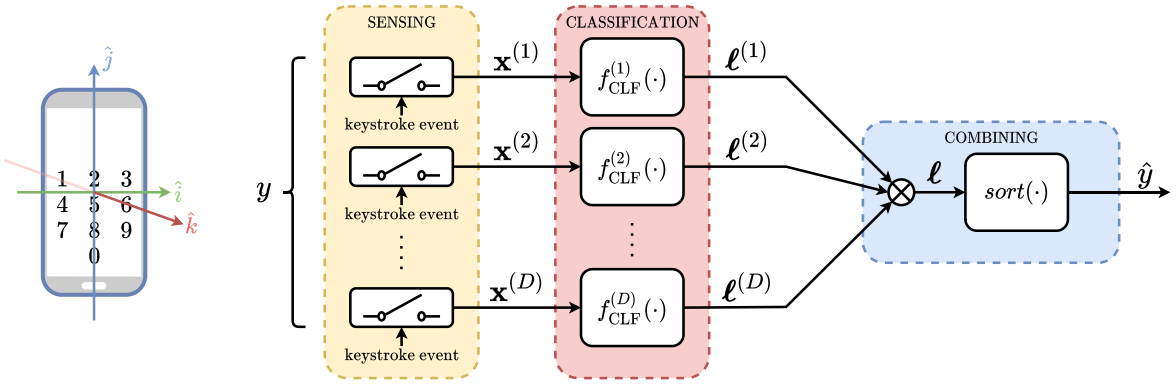


FIGURE 1. PIN recognition block diagram.

application could contain a code able to sniff motion sensor data, even without the victim's consent. In this case, when a PIN is inserted on a web browser or on another application, the malicious software running in the background would be able to sample the motion sensors. However, on smartphones running Android 9 or higher, background applications cannot access sensor events as a security measure [1]. Consequently, in such devices, this first possibility is no longer viable. Nevertheless, a second possible solution to sniff motion data is given by web browsers, as investigated in recent works [14], [15]. In this case, a malicious or compromised web page could receive the motion sensor data while the victim is inserting a PIN on another web page, e.g., a bank website. In [14], four types of attacks based on malicious web pages and online advertisements are presented. Furthermore, in [15], the authors propose a cross-site attack to infer motion-sensor data by using a malicious web page.

Relying on these previous works on sensor data vulnerability, in this study we assume that an attacker has the possibility to acquire motion sensor data from a victim's mobile device. Given the smartphone movements recorded through these sensors while the victim is typing a PIN, our objective is to infer the digits composing the PIN. Motion sensor data are highly user-dependent, and there is no theoretical model able to map this information to the corresponding pressed digit. So, we approach the PIN recognition task as a classification problem solved through supervised learning techniques. In particular, we implement and evaluate four Machine Learning (ML) algorithms: Random Forest (RF), Support Vector Machine (SVM), Neural Network (NN), and k-Nearest Neighbors (KNN). To obtain a realistic estimation of the prediction accuracy, the algorithms are trained on samples generated by multiple users with multiple devices and tested on new unseen examples.

The key novelties of this paper can be summarized as follows:

- We do not pose restrictions on the PIN length and, given an N-digits PIN, we assume that all the 10^N PIN configurations are possible, differently from previous literature.
- We acquire motion information in an event-driven manner: a sample of the considered motion sensors is taken only when a key is pressed by the user. In this way, the amount of sniffed data necessary for the attack is minimized.
- We observe that users insert the same PIN each time they want to access a particular service. Thus, we propose to increase the classification accuracy by exploiting the diversity deriving from multiple observations of the same typed PIN.

To promote reproducible research, the dataset and simulation code is available at https://github.com/matteonerini/pin-side-channel-attacks.

Organization: The rest of the paper is organized as follows. In Section II, we review the related literature. In Section III, we discuss the system model and the data collection. In Section IV, we briefly describe the ML algorithms involved. In Section V, we present our PIN side-channel attack based on motion sensors. In Sections VI and VII, we report the obtained performance and discuss the key observations, respectively. Finally, Section VIII contains the concluding remarks.

Notation: Throughout the paper, capital boldface letters denote matrices and tensors, while lowercase bold letters denote vectors. $(\mathbf{A})^{\mathrm{T}}$, $\|\mathbf{A}\|_2$, and $[\mathbf{A}]_{i,j}$ denote the transpose, the ℓ_2-norm, and the (i,j)-th element of a matrix $\mathbf{A}$.

II. RELATED WORK

In this section, we review the literature on the role of motion sensors and behavioral biometrics in smartphone security. Motion sensors are able to capture behavioral biometrics, i.e., recurrent patterns typical of each person [16], [17], [18]. For this reason, some studies proposed authentication methods completely based on behavioral biometrics, utilizing wrist-worn devices. In [19], [20], and [21], these devices are used to collect wrist movements while the user performs a specific gesture. Thus, the identity of the person wearing the device is verified from these motion data. In [6] and [22] , wrist movements are analyzed jointly with mouse and keystroke activities. The correlation between these activities

is used to provide Continuous Authentication (CA). CA can be provided also on the basis of the behavioral traits contained in the movements recorded by smartphones [23], [24]. Here, the user is authenticated while performing daily life activities such as sitting, walking, and running. In [25], CA on smartphones is obtained by using app-usage information. However, authentication methods solely based on behavioral biometrics might be not enough accurate due to the irregular nature of the movements [26].

Motion sensors and behavioral biometrics have been also used to make a PIN, or an alphanumeric password, more robust against cyber-attacks. In some works, the PIN is entered using touch gestures substituting the keystrokes, making it more secure against shoulder surfer and side-channel attacks [27], [28]. In [15], the effectiveness of side-channel attacks is reduced by adding Gaussian noise to the motion sensor data. However, this data perturbation also affects the accuracy of the sensors and their utility. In [29] and [30], the user is requested to draw the digits of the PIN or predefined patterns on the touchscreen. In this way, the user's drawing traits are used as a further authentication measure, beyond the secrecy of the PIN. In [31], the PIN strength is improved by verifying that the smartphone movements during the PIN insertion are typical of the smartphone owner. In this way, the PIN security is improved with no further actions required from the user. Finally, keystroke dynamics information, describing the person's typing rhythm, can be used to enhance the security of alphanumeric passwords [32], [33], or to provide free-text authentication [34].

The information provided by smartphone sensors has been also used to compromise smartphone security. In [35], the sounds produced while typing the PIN have been used to infer the PIN secret combination. In [36], smartphone speakers have been used to produce ultrasound signals. Then, the PIN has been inferred based on the echos recorded by the smartphone microphone. Supervised learning techniques have been used to identify the PIN from motion data [37], [38], [39], [40], [41], [42], [43], [44], [45], [46]. However, three problems can be identified in the related works. First, the set of possible PINs configurations has been significantly reduced in other works, with consequent simplification of the problem. In particular, several works restricted the search space to a subset of a few tens of possible 4-digits PINs [37], [38], [39], [40], [41]. To select the most user-chosen PINs, dedicated studies have been conducted [42]. Second, another important issue regards motion data acquisition. A common strategy is to sample the motion sensors with the maximum allowed sampling frequency, ranging typically from 50 Hz to 100 Hz [37], [40], [43], [44], [45]. This approach produces a huge amount of data to sniff from the device. Third, in some works, the generalization capability of the PIN cyber-attack has not been fully tested. In [43], a subset of the training set has been used to test the classifier accuracy. As a consequence, the recognition capabilities cannot be generalized to unseen patterns. In [46], training and testing are carried out on a dataset built with a single user typing on a single smartphone model, which is a strong assumption, and hard to implement in a practical attack.

III. SYSTEM MODEL

In this section, we describe how a real-world dataset has been constructed and which relevant features deriving from motion sensor data have been selected. To collect the training and test data, we developed a smartphone application able to sample motion sensors when the user enters a digit in a digital numeric keypad. Note that a web page could be equivalently used to capture the data through a browser. To capture the data through a web page, a dedicated website could be built to sample the sensor data while the user is interacting with a keypad, as investigated in [14] and [15]. Through our application, when a digit is pressed, sensor data are recorded and labeled with the respective digit. Since the sensor sampling operation is triggered only in correspondence with each keystroke, the amount of sniffed data is significantly low.

A. SENSORS OF INTEREST

Android OS supports three categories of sensors: motion sensors, environmental sensors, and position sensors [1]. For our purposes, we consider only motion sensors, since their access is not protected. In Android Studio, their values are provided according to a coordinate system defined with respect to the device screen. More precisely, the $\hat{i}$ axis is horizontal and points to the right, the $\hat{j}$ axis is vertical and points upward, the $\hat{k}$ axis is perpendicular to the screen of the device (see Fig. 1). We selected six relevant sensors for the PIN recognition, which are described below:

- The *Accelerometer* measures the total acceleration in m/s^2, experienced on the three axes $\hat{i}, \hat{j}, \hat{k}$.
- The *Gravity* sensor gives the components of the gravitational acceleration g in m/s^2 along each direction $\hat{i}, \hat{j}, \hat{k}$.
- The *Gyroscope* measures the angular speed, in rad/s, around the three device axes $\hat{i}, \hat{j}, \hat{k}$. The rotation is positive in the counter-clockwise direction.
- The *Linear Acceleration* indicates the acceleration experienced along each device axis $\hat{i}, \hat{j}, \hat{k}$, without the gravity contribution.
- The *Rotation Vector* is composed of three dimensionless components and it represents the rotations of the device $\hat{i}, \hat{j}, \hat{k}$ axes with respect to the East, the geomagnetic North, and the Zenith, respectively.
- The *Orientation* sensor returns an array of three angles: *azimut*, i.e. is the angle between the geomagnetic North direction and the device $\hat{j}$ axis; *pitch*, i.e., is the angle of rotation around the $\hat{i}$ axis; and *roll*, which is the angle of rotation around the $\hat{j}$ axis.

Among the 18 total values provided, three of them can be discarded because not relevant to our scope. In particular, we did not consider the components of the *Rotation Vector* taken with reference to the North and East directions, and we discarded the *azimut* value of the *Orientation* sensor. These

three features are linked to the geographic direction pointed by the smartphone, which is independent of the pressed digit. In addition, we appended a further value M to the features set representing the total inclination with respect to the Zenith, defined as $M = \sqrt{pitch^2 + roll^2}$. In conclusion, a total of 16 features have been used to represent the smartphone movement.

B. DATA COLLECTION

For data acquisition, we developed a smartphone application able to sample the motion sensors when a user enters a digit in a numeric keypad. The application, developed with the Android Studio environment, has been designed to collect both the training set, to instruct the classification algorithms, and the test set, to evaluate their classification performance. The user interface consists of a single screen exhibiting a numeric keypad, as represented in Fig. 1. The recorded data is organized by the application into a table stored in a Comma-Separated Values (CSV) file. When a digit in this keypad is pressed, the considered motion sensors are recorded together with the value of the pressed digit, and a row is added to the CSV file. Finally, each row contains 17 numerical values: one value representing the pressed digit, and 16 features sampled by the considered motion sensors.

We recruited 12 volunteer students, expert smartphone users, who installed the dedicated application on their smartphones. All devices were Android, running an updated version of the operative system (7 Nougat or higher). Each student was asked to type a list of 500 digits randomly generated (50 samples of each digit per student) in the numeric keypad shown in the application, while naturally holding their smartphone with one hand. At the end of the typing session, not all students typed exactly 500 digits, and the resulting dataset consisted of 5400 digits. The set was randomly split into training and test subsets. The training set, including 90% of the dataset, was used to choose the hyper-parameters of the ML models via 5-fold cross-validation, and for the final training. The value 5 in the cross-validation has been chosen such that each training subset of data samples is large enough to be statistically representative of the whole dataset. The test set, including the remaining 10% of the data, was considered to assess the performance of the models. Each set contained the same percentage of samples of each target class as the complete dataset.

IV. THE ADOPTED CLASSIFICATION ALGORITHMS

We consider first a single-digit classification problem, and we extend it to a general N-digits PIN recognition afterward. The single-digit case is formalized as a classification problem with 10 classes, solved in a supervised manner [47], [48], [49]. In this study, we evaluate the performance of four ML techniques commonly used for supervised classification tasks. In this section, we briefly recall the algorithms and the chosen hyper-parameters which have been tuned through 5-fold cross-validation [47].

A. RANDOM FOREST (RF)

A RF is an ensemble technique that fits a certain number of decision trees, N_T, on different sub-samples of the whole dataset [50]. The sub-samples size is the same as the original dataset size, where the samples are taken with replacement using the so-called bootstrapping technique. In this way, the overfitting is controlled by averaging the decision of every individual tree [50]. In each decision tree, nodes are split with respect to the feature that minimizes the Gini impurity, defined as:

$$I_G = \sum_{i=1}^{N_\mathrm{c}} p_i(1 - p_i), \tag{1}$$

where N_c is the total number of classes and p_i the fraction of items labeled with class i. In our RF implementation, we found through 5-fold cross-validation that the choice of $N_\mathrm{T} = 150$ trees, gives a suitable compromise between complexity and performance. Furthermore, to smooth the model, a node is split if it contains at least four samples, and the split is considered valid if there are at least two samples in each branch.

B. SUPPORT VECTOR MACHINE (SVM)

A SVM constructs a set of hyper-planes in a multi-dimensional space that can be used for classification or regression tasks [47]. Let us consider a training set composed by N_o samples $\mathbf{x}_n$ in the feature space $\mathbb{R}^s$, where $s = 16$ is the number of features involved. A label $y_n \in \mathbb{R}$ is associated to each training example $\mathbf{x}_n$. In a binary classification problem, where $y_n \in \{1, -1\}$, the estimated label $\hat{y}_n$ is given by $\hat{y}_n = sign\left(\mathbf{w}^T \phi(\mathbf{x}_n) + b\right)$, where $\phi(\cdot)$ denotes a fixed feature space transformation. Thus, the goal of a SVM is to learn $\mathbf{w} \in \mathbb{R}^s$ and $b \in \mathbb{R}$ by minimizing the total margin violation $\sum_{n=1}^{N_\mathrm{o}} \zeta_n$ as follows:

$$\min_{\mathbf{w},b,\zeta_n} C \sum_{n=1}^{N_\mathrm{o}} \zeta_n + \frac{1}{2} \|\mathbf{w}\|_2^2 \tag{2}$$

$$\text{s.t. } y_n \left(\mathbf{w}^T \phi(\mathbf{x}_n) + b\right) \geq 1 - \zeta_n, \quad \forall n, \tag{3}$$

$$\zeta_n \geq 0, \quad \forall n, \tag{4}$$

where ζ_n is the margin violation for the sample $\mathbf{x}_n$. In the objective function to be minimized, the hyper-parameter C controls the strength of the regularization term $\frac{1}{2} \|\mathbf{w}\|_2^2$.

This optimization problem can be solved through its dual, defined as:

$$\min_{\alpha_n} \frac{1}{2} \sum_{m=1}^{N_\mathrm{o}} \sum_{n=1}^{N_\mathrm{o}} \alpha_m \alpha_n y_m y_n K(\mathbf{x}_m, \mathbf{x}_n) - \sum_{n=1}^{N_\mathrm{o}} \alpha_n \tag{5}$$

$$\text{s.t. } \mathbf{y}^T \boldsymbol{\alpha} = 0, \tag{6}$$

$$0 \leq \alpha_n \leq C, \quad \forall n, \tag{7}$$

where α_n are such that $\mathbf{w} = \sum_{n=1}^{N_\mathrm{o}} \alpha_n y_n \mathbf{x}_n$ and $K(\mathbf{x}_m, \mathbf{x}_n) = \phi(\mathbf{x}_m)^T \phi(\mathbf{x}_n)$ is the kernel [47], [48]. In our implementation,

we use a Radial Basis Function kernel defined by [51]:

$$K(\mathbf{x}_m, \mathbf{x}_n) = \exp\left(-\gamma \|\mathbf{x}_m - \mathbf{x}_n\|_2^2\right), \tag{8}$$

where $\mathbf{x}_m$ and $\mathbf{x}_n$ are two generic dataset points. Furthermore, $\gamma = 1/16$, and $C = 3$ have been chosen based on 5-fold cross-validation. Since our problem is a 10-class classification, the so-called one-versus-one approach is implemented: in total, $10(10-1)/2 = 45$ binary classifiers are constructed.

C. NEURAL NETWORK (NN)

A feed-forward NN can be employed to set the boundaries between the classes representing the different digits [48]. In our architecture, all the layers are fully connected, with activation functions being Rectified Linear Unit (ReLU) for the hidden layers and softmax for the output layer. The network is trained with all the training set points for a maximum of $N_{\mathrm{E}} = 1000$ epochs through the Adam stochastic gradient-based optimizer algorithm, with learning rate $\lambda = 5 \times 10^{-3}$. The loss function to be minimized is given by the cross-entropy [48]. Convergence of the optimization is considered to be reached if the classification accuracy on the validation set does not decrease by at least 10^{-4} in 25 consecutive epochs. The NN input layer has 16 neurons (as the number of features), while the output layer has 10 neurons since there are 10 possible digits (i.e., the classes in our classification problem). Additionally, two hidden layers have been used, with 100 and 50 neurons, respectively. To avoid overfitting, we add an ℓ_2 regularization term with strength 10^{-4} to the loss function.

D. K-NEAREST NEIGHBORS (KNN)

In KNN a set of N_{o} pairs $\{(\mathbf{x}_n, y_n)\}_{n=1}^{N_{\mathrm{o}}}$ is given as training set, where $\mathbf{x}_n$ takes values in the feature space $\mathbb{R}^s$ upon which is defined the metric $d(\mathbf{x}_n, \mathbf{x}_m)$. Here, this metric is taken as the Euclidean distance. When $k = 1$, given a test point $(\mathbf{x}_m, y_m)$, the estimate of y_m is given by the nearest neighbor training point with respect to the test point as

$$\hat{y}_m = \left\{ y_l : \mathbf{x}_l = \arg\min_{\mathbf{x}_n} d(\mathbf{x}_n, \mathbf{x}_m) \right\}, \tag{9}$$

so that $\mathbf{x}_m$ is assigned to the same class of the nearest point $\mathbf{x}_l$. In general, instead of considering only the closest neighbor, the nearest k neighbors are taken into account for the classification, according to the majority rule [52]. In our KNN algorithm, the 5-fold cross-validation suggested $k = 8$.

V. PIN SIDE-CHANNEL ATTACK

To recognize an N-digit PIN, we first analyze the classification of a single digit taking values from 0 to 9. This is formalized as a multi-class classification problem with 10 classes. Then, we solve N instances of this problem by feeding the classifier with a new input for each digit that composes the PIN. In this way, the identification is extended from a single digit to the whole PIN. To classify a single digit, we implement and evaluate four ML algorithms: RF, SVM, NN, and KNN [47]. These ML techniques receive in input the 16 values obtained from the motion sensors, as discussed in Section III, and return the predicted class of the pressed key. For the considered algorithms, we are interested in the likelihood for all 10 classes, since every authentication system allows multiple attempts to enter the correct numerical password. To this end, for each classification algorithm considered, the likelihoods of the output classes have been defined as follows:

- For each tree of the RF, given a specific input $\mathbf{x}_n$, we define the likelihood of each class as the fraction of training samples of the class in the leaf where $\mathbf{x}_n$ is collocated. Then, these values are averaged over all trees to obtain the likelihood estimates.
- For each binary SVM constructed, the output score for a given sample $\mathbf{x}_n$ is given by the distance between $\mathbf{x}_n$ and the separating hyper-plane. Pairwise coupling has been used to calculate the likelihood estimate for each class, by combining the 45 binary score values [47].
- In the NN, the likelihood estimates for each class are given by the values of the softmax activation function applied to the output layer, when the sample $\mathbf{x}_n$ is provided in input.
- In the KNN classifier, given a sample $\mathbf{x}_n$, the likelihood estimate of a class is the fraction of the k neighbors of $\mathbf{x}_n$ belonging to that class.

For all these classification algorithms, the final output consists of a list of 10 likelihoods.

Furthermore, a user inserts always the same PIN when authenticating to access a particular service. Thus, we assume that several samples corresponding to the same pressed digits can be sniffed by the attacker, who exploits this diversity to increase recognition accuracy. Based on this observation, considering a diversity order D, we feed the classifier $f_{CLF}(\cdot)$ with D samples $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(D)}\}$, taken from the test set, corresponding to the same pressed digit y. Then, for each output class, the obtained estimates $\{\boldsymbol{\ell}^{(1)}, \boldsymbol{\ell}^{(2)}, \ldots, \boldsymbol{\ell}^{(D)}\}$ are combined to obtain a more reliable one $\boldsymbol{\ell}$, as shown in Fig. 1. Finally, the first choice is the class with the highest combined likelihood. Three combining methods have been implemented and tested, in which the resulting likelihoods have been designed as:

i) the maximum among the D likelihoods

$$\ell_k = \max\left\{\ell_k^{(1)}, \ell_k^{(2)}, \ldots, \ell_k^{(D)}\right\}, \quad k = 1, \ldots, 10; \tag{10}$$

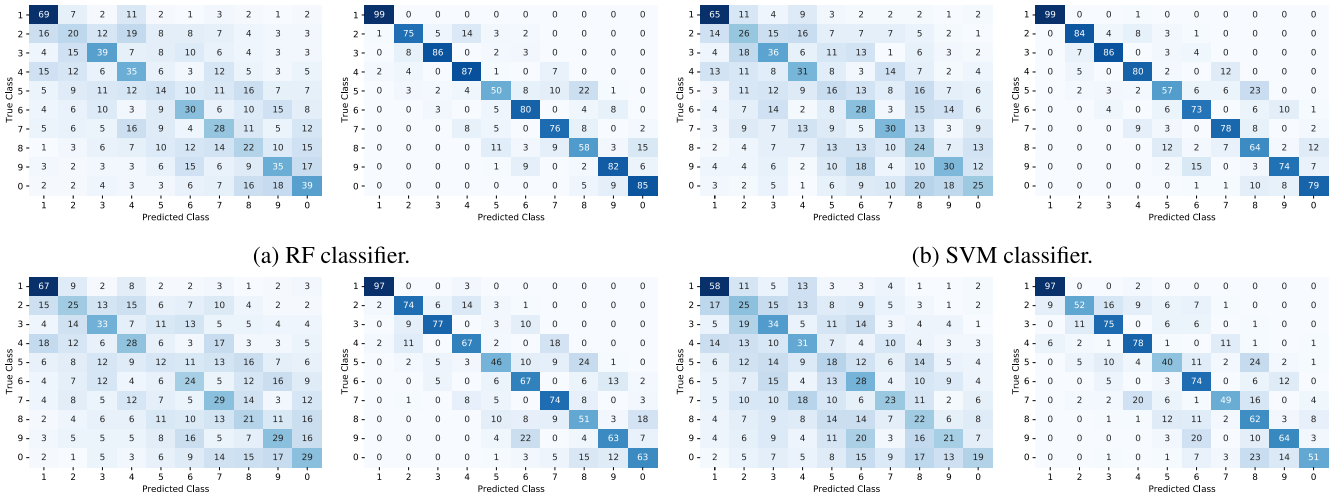
ii) the arithmetic mean of the D likelihoods

$$\boldsymbol{\ell} = \frac{1}{D}\sum_{d=1}^{D} \boldsymbol{\ell}^{(d)}; \tag{11}$$

iii) the geometric mean of the D likelihoods

$$\boldsymbol{\ell} = \left(\prod_{d=1}^{D} \boldsymbol{\ell}^{(d)}\right)^{1/D}; \tag{12}$$

where the product is to be intended as element-wise. For the sake of comparison, we analyze the performance of all



(a) RF classifier.

(b) SVM classifier.

(c) NN classifier.

(d) KNN classifier.

FIGURE 2. Confusion matrix, in percentage, of the four classifiers without diversity ($D = 1$), on the left, and with diversity $D = 10$, on the right.

the four aforementioned classifiers used as $f_{CLF}(\cdot)$. Thus, we implement the function $f_{CLF}(\cdot)$ independently with the four considered classifiers: RF, SVM, NN, and KNN. Experimentally, for the KNN classifier, the best-performing combiner has been proved to be the arithmetic mean. For the other algorithms, we found that the geometric mean leads to a more accurate recognition. Indeed, the product is able to penalize the estimates which are close to zero, hence excluding those digits which are very unlikely in at least one output over the D available. Thus, in the following, the presented results have been obtained with these combining techniques. The simulations have been conducted on Google Colab servers, running a Central Processing Unit (CPU) Intel Xeon, with a clock frequency of 2.20 GHz [53]. The considered classification algorithms have been implemented in Python through the library Scikit-learn [54].

VI. NUMERICAL RESULTS

In this section, we measure the performance of our attack in terms of accuracy (or success rate) A_1, and F-score, defined as follows. We define the accuracy A_1 as the proportion of correct predictions among the total number of tested samples. Thus, given a multi-class classifier with confusion matrix $\mathbf{C}$, the accuracy writes as

$$A_1 = \frac{\sum_{k=1}^{C} [\mathbf{C}]_{k,k}}{\sum_{i=1}^{C} \sum_{j=1}^{C} [\mathbf{C}]_{i,j}}, \tag{13}$$

where C is the number of classes. Furthermore, in a multi-class classification problem, for the k-th class, we denote as TP_k (resp. FP_k) the number of true (resp. false) positives, i.e., the number of test samples correctly (resp. incorrectly) classified as belonging to that class. Similarly, for the k-th class, we denote as TN_k (resp. FN_k) the number of true (resp. false) negatives, i.e., the number of test samples correctly (resp. incorrectly) classified as not belonging to that class. For the k-th class, the F-score F_k is given by the harmonic mean of precision and recall. Thus, it can be written as

$$F_k = \frac{2TP_k}{2TP_k + FP_k + FN_k}. \tag{14}$$

Given a confusion matrix $\mathbf{C}$, the F score for the k-th class (14) can be computed as

$$F_k = \frac{2\,[\mathbf{C}]_{k,k}}{2\,[\mathbf{C}]_{k,k} + \sum_{i \neq k} [\mathbf{C}]_{i,k} + \sum_{j \neq k} [\mathbf{C}]_{k,j}}. \tag{15}$$

Finally, the overall F-score F is computed as the mean over the 10 possible classes $F = \frac{1}{10} \sum_{k=1}^{10} F_k$.

A. SINGLE-DIGIT RECOGNITION

We start by analyzing the performance on the test set of the four considered classification algorithms for the single-digit classification problem. To obtain robust assessments, independent of the split of the dataset into training and test sets, we repeated our experiments by dividing 20 times the dataset with different seeds, according to the Monte Carlo method. Fig. 2 shows the confusion matrices, in percentage, for digit recognition performed with the RF, SVM, NN, and KNN classifier. Here, we consider no diversity, on the left, and diversity $D = 10$, on the right. In general, the digits which are more difficult to recognize are "2", "5", and "8". These are the internal keys in the numeric keypad, and, as expected, are the most difficult to classify. On the other hand, the digits occupying corner positions in the keypad are detected with the highest accuracy.

Within the test set, composed of samples never used during the training phase, each of the four considered algorithms correctly classifies the pressed digit with an accuracy A_1. This means that the correct classification of a digit occurs at the first attempt with a rate A_1. In addition, we denote with A_2

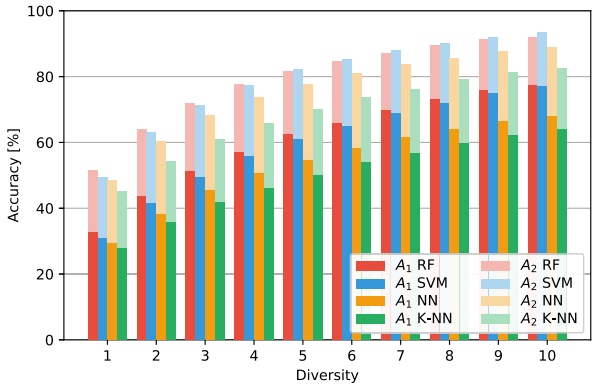


FIGURE 3. Accuracy in recognizing the pressed digit at the first and the second attempt, A_1 and A_2, for each classifier, and with different diversity orders D.

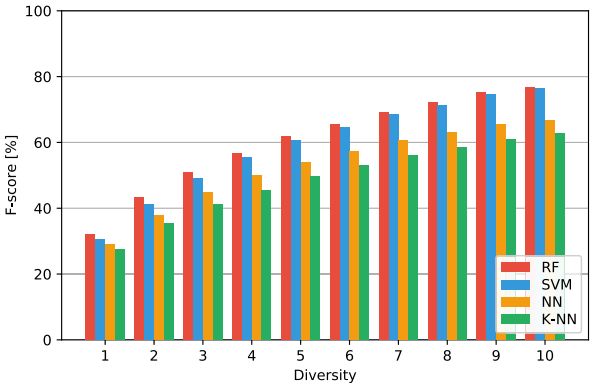


FIGURE 4. F-score for each classifier, and with different diversity orders D.

the accuracy of the same algorithms when the second most likely class is chosen as the prediction. Thus, we have that the correct classification takes place on the second attempt with a rate A_2. In Fig. 3, the accuracies A_1 and A_2 are reported for different values of diversity D, while in Fig. 4, the F-score is reported.

As expected, the accuracy of all four implemented classifiers increases monotonically with the diversity degree. Among them, the RF gives the best classification performance. When diversity is not considered, the RF recognizes a digit with a success rate of 33% and 19% at the first and second attempt, respectively. In the presence of diversity, the accuracy improves significantly. For example, when $D = 10$, i.e. the attacker can obtain 10 different samples associated with the same inserted digit, the correct recognition takes place with a probability of 78% and 14% at the first and second attempt, respectively. Finally, we notice that the first-attempt accuracy A_1 reflects approximately the F-score of the classifiers for every value of diversity.

B. N-DIGIT PIN RECOGNITION

Now, the identification is extended from a single digit to a PIN composed of N digits. In this case, it is not possible to provide reliable experimental results given the high number of classes in this multi-class classification problem, i.e., 10^N. For this reason, we use a probabilistic approach assuming that an N-digit PIN classification is composed of N independent single-digit classifications. Given a probability confusion matrix $\mathbf{C} \in \mathbb{R}^{10\times 10}$ of a single-digit classifier, its entries are conditional probabilities defined as

$$[\mathbf{C}]_{m+1,n+1} = P(n|m), \tag{16}$$

where $P(n|m)$ denotes the probability to detect the digit n given the true digit m. Similarly, the (i,j)-th entry of the confusion matrix of a N-digit classifier $\mathbf{C}^{(N)} \in \mathbb{R}^{10^N\times 10^N}$ is given by the conditional probability

$$\left[\mathbf{C}^{(N)}\right]_{i+1,j+1} = P((j_1,\ldots,j_N)\,|\,(i_1,\ldots,i_N)), \tag{17}$$

where i_n and j_n are defined such that $i = \sum_{n=1}^{N} i_n 10^{N-n}$ and $j = \sum_{n=1}^{N} j_n 10^{N-n}$. Assuming that the classifications of the N digits are independent, (17) can be reformulated as

$$\left[\mathbf{C}^{(N)}\right]_{i+1,j+1} = \prod_{n=1}^{N} P(j_n|i_n). \tag{18}$$

Thus, plugging (16) into (18), we obtain

$$\left[\mathbf{C}^{(N)}\right]_{i+1,j+1} = \prod_{n=1}^{N} [\mathbf{C}]_{i_n+1,j_n+1}, \tag{19}$$

providing the expression of the N-digit PIN classification confusion matrix $\mathbf{C}^{(N)}$ as a function of the single-digit classification confusion matrix $\mathbf{C}$.

Given the obtained $\mathbf{C}^{(N)}$, an N-digit PIN is correctly entirely recognized at the first attempt with an accuracy

$$A_1^{(N)} = \frac{\sum_{k=1}^{10^N} \left[\mathbf{C}^{(N)}\right]_{k,k}}{\sum_{i=1}^{10^N}\sum_{j=1}^{10^N} \left[\mathbf{C}^{(N)}\right]_{i,j}}, \tag{20}$$

according to the definition of accuracy. First, the numerator of (20) can be rewritten as

$$\sum_{k=1}^{10^N} \left[\mathbf{C}^{(N)}\right]_{k,k} = \sum_{k=1}^{10^N}\prod_{n=1}^{N} [\mathbf{C}]_{k_n,k_n} \tag{21}$$

$$= \sum_{k_1=1}^{10}\cdots\sum_{k_N=1}^{10}\prod_{n=1}^{N} [\mathbf{C}]_{k_n,k_n} \tag{22}$$

$$= \sum_{k_1=1}^{10} [\mathbf{C}]_{k_1,k_1}\cdots\sum_{k_N=1}^{10} [\mathbf{C}]_{k_N,k_N} \tag{23}$$

$$= \left(\sum_{k=1}^{10} [\mathbf{C}]_{k,k}\right)^N, \tag{24}$$

by applying (19). Since $A_1 = \sum_{k=1}^{10} [\mathbf{C}]_{k,k}/10$, we have

$$\sum_{k=1}^{10^N} \left[\mathbf{C}^{(N)}\right]_{k,k} = (10A_1)^N. \tag{25}$$

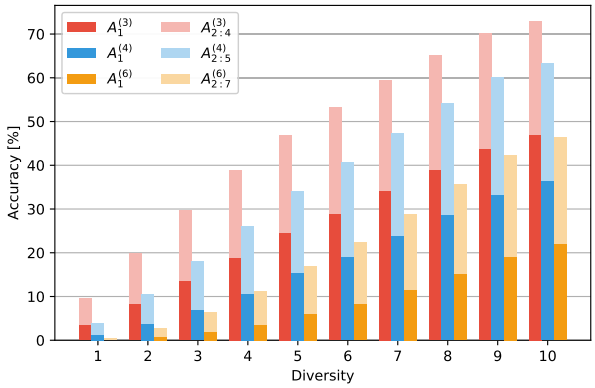


FIGURE 5. *N*-digit PIN recognition accuracy, where $N \in \{3, 4, 6\}$, using the RF classifier, with different diversity orders D.

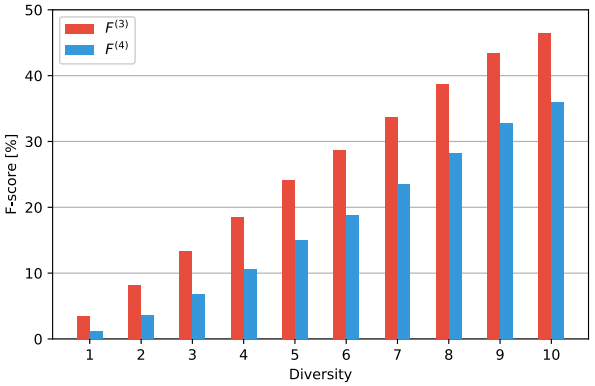


FIGURE 6. *N*-digit PIN recognition *F*-score, where $N \in \{3, 4\}$, using the RF classifier, with different diversity orders D.

Second, according to (17), the denominator of (20) is given by

$$\sum_{i=1}^{10^N}\sum_{j=1}^{10^N}\left[\mathbf{C}^{(N)}\right]_{i,j} = 10^N. \tag{26}$$

Thus, plugging (25) and (26) into (20), we obtain

$$A_1^{(N)} = {A_1}^N. \tag{27}$$

The intuition behind (27) is that an N-digit PIN is correctly identified at the first attempt when all the N digits are correctly identified. In addition, there are N combinations in which the i-th position is occupied by the key recognized at the second attempt, with $i \in \{1, 2, \ldots, N\}$, while the $N-1$ remaining digits are immediately correctly identified at the first attempt. The probability $A_{2:N+1}^{(N)}$ that the PIN under attack is in the set of these N is:

$$A_{2:N+1}^{(N)} = N{A_1}^{N-1}A_2. \tag{28}$$

Finally, by adding (27) and (28), we obtain the probability $A_{1:N+1}^{(N)}$ to identify an N-digit PIN within $N+1$ attempts:

$$A_{1:N+1}^{(N)} = A_1^{(N)} + A_{2:N+1}^{(N)} = {A_1}^N + N{A_1}^{N-1}A_2. \tag{29}$$

Such accuracies are plotted in Fig. 5 for three different PIN lengths: $N=3$, $N=4$, and $N=6$. We selected these values since $N=3$ is the length of the Card Verification Value (CVV), a code commonly used to perform online payments, while $N=4$ and $N=6$ are the most popular PIN lengths. In Fig. 5, the solid colored bars represent the accuracy with which the PIN is correctly identified at the first attempt, while the faded colors depict the accuracies $A_{2:N+1}^{(N)}$. Thus, the total height of the bars indicates the accuracy with which an N-digit PIN is recognized in $N+1$ trials. In particular, provided that $D=10$ different observations of the PIN have been acquired by the attacker, a 3-digit PIN is identified in four attempts with accuracy 72%, a 4-digit PIN is identified in five attempts with accuracy 63%, and a 6-digit PIN is identified in seven attempts with accuracy 47%. If at most five attempts are allowed to insert the PIN, the accuracy is 74% and 38% for 3-digit and 6-digit PIN, respectively. With respect to previous literature, each digit is here correctly recognized at the first attempt on average with an accuracy of 78%, to be compared with accuracy 70% in [46] and [55]. Moreover, the proposed event-driven attack requires significantly less sniffed data, as analyzed in Section VII.

Finally, we assess the N-digit PIN recognition in terms of F-score. Given an N-digit PIN classifier with confusion matrix $\mathbf{C}^{(N)}$, the F score for the k-th class (14) can be computed as

$$F_k^{(N)} = \frac{2\left[\mathbf{C}^{(N)}\right]_{k,k}}{2\left[\mathbf{C}^{(N)}\right]_{k,k} + \sum_{i\neq k}\left[\mathbf{C}^{(N)}\right]_{i,k} + \sum_{j\neq k}\left[\mathbf{C}^{(N)}\right]_{k,j}}. \tag{30}$$

Besides, the overall F-score $F^{(N)}$ is computed as the mean over the 10^N possible classes $F^{(N)} = \frac{1}{10^N}\sum_{k=1}^{10^N} F_k^{(N)}$. This F-score is reported in Fig. 6 for two different PIN lengths: $N=3$ and $N=4$. We observe that the F-score is similar to the accuracy $A_1^{(N)}$ shown in Fig. 5, as in the single-digit recognition problem.

VII. DISCUSSION

In the case of the NN classifier, a common problem affecting the classification performance on the test set is overfitting. When overfitting occurs, the classifier has high accuracy on the training set by low accuracy on unseen samples. To verify that our NN is not overfitting, we report the observed learning curves in Fig. 7. Here, the training loss is the cross-entropy loss on the training set, while the validation accuracy is the accuracy on the validation set over iterations. The curves have been averaged over 20 training runs initialized with random values. As we can observe from the validation accuracy curve, overfitting is successfully avoided thanks to two properties of our NN. First, we add an ℓ_2 regularization term to the loss function. Second, we implement the so-called "early-stopping" technique to interrupt the training process as soon as the classification accuracy stops improving over iterations.

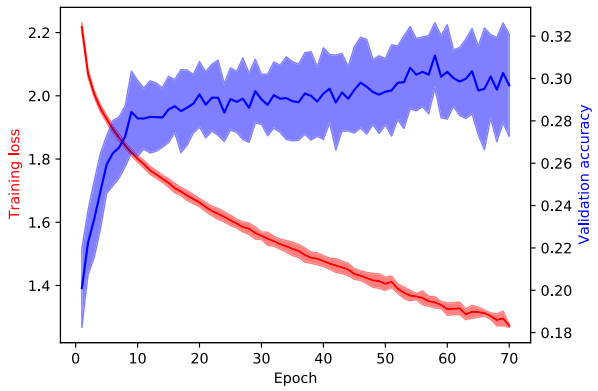


FIGURE 7. Learning curves of the NN classifier.

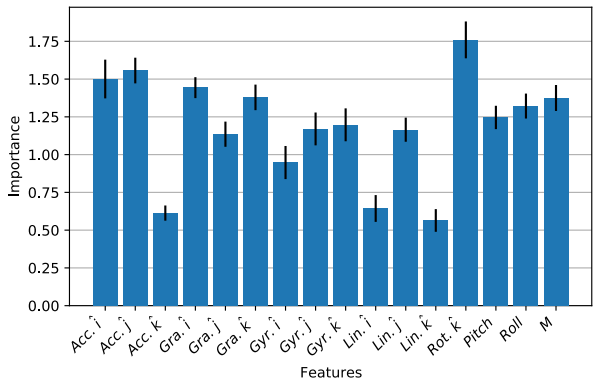


FIGURE 8. Feature importance on the single-digit classification task using the RF, together with the 95% confidence interval.

In our experiments, all the relevant motion sensors available in Android smartphones have been considered. To understand the contribution of each sensor in the construction of the RF, i.e., the best-performing classifier, we introduce the so-called feature importance. For any feature, this metric is the increase in the classification error when the values of that feature are randomly permuted over the out-of-bag observations, i.e., the samples ignored during the construction of each tree. The feature importance is evaluated for every tree, then averaged over the entire forest and divided by the standard deviation over the ensemble. The importance of the 16 features described in Section III-A is reported in Fig. 8. Here, the values are averaged over 20 RF instances, initialized with different seeds. In these different instances, we do not observe large variations, as we can see from the narrow confidence intervals. Since each bar in the chart is strictly positive, we can conclude that all the sensors considered are informative, even if some components significantly contributed more than others. The *Rotation Vector* component, which accounts for the rotation of the device $\hat{k}$ axis with respect to the Zenith direction, resulted to be the most important feature, followed by the acceleration along the $\hat{\imath}$ and $\hat{\jmath}$ axes.

In the related literature, the motion sensors are sampled with the maximum allowed sampling frequency f_s, ranging typically from 50 Hz to 100 Hz, as in [37], [40], [43], [44], and [45]. Conversely, in our method, the sensors are sampled only in correspondence with a keystroke. Thus, the sampling frequency is given by the digits per second (DPS) the user inserts. Experimentally, we notice that the DPS ranges approximately from 1 Hz (when the PIN digits are inserted very slowly) to 5 Hz (when the PIN digits are inserted very quickly). We present the reduction in sniffed data compared with the literature in Tab. 1. From Tab. 1, we notice that the maximum reduction is obtained when the user inserts 1 digit per second. Compared with a sampling frequency $f_s = 100$ Hz, we sniff 100 times fewer data. Conversely, the minimum reduction is experienced when the user inserts 5 digits per second. In this case, compared with a sampling frequency $f_s = 50$ Hz, we sniff 10 times fewer data. In conclusion, our method requires 10 to 100 times less sniffed data compared to the related literature.

TABLE 1. Reduction factor for different sampling frequencies f_s and DPS values.

	$DPS = 1$	$DPS = 2$	$DPS = 5$
$f_s = 50$ Hz	50	25	10
$f_s = 100$ Hz	100	50	20

VIII. CONCLUSION

In this study, we have shown the feasibility of inferring numerical passwords inserted on smartphones just by reading motion sensor data, which are freely accessible. To this end, we proposed a ML-based approach able to learn a mapping between smartphone movements and the pressed digits. We minimized the amount of needed data with an event-driven sampling of the considered sensors. Assuming a smartphone user inserts the PIN with a rate of 2 digits per second, our method requires 50 times less sniffed data than related studies employing a sampling frequency of 100 Hz. In addition, we considered a diversity-based method to significantly improve recognition performance. Our approach, trained and validated with different smartphone models and with several users, requires significantly less sniffed data with respect to previous methods, achieving better accuracy at the same time. 3-digit PINs are correctly recognized at most four attempts with an accuracy higher than 70%. To prevent such side-channel attacks, security restrictions should be implemented also for data apparently not sensitive.

We identify two different future research directions. First, possible attacks based on smartphone motion sensors should be investigated also for other authentication methods. For instance, several authentication methods require the user to draw patterns on the touch screen. These patterns could be identified by an attacker accessing smartphone movements. Second, techniques to counteract such motion sensors-based attacks should be developed. To this end, a possible strategy involves the use of behavioral biometrics to strengthen the security of authentication methods such as PINs or patterns. To effectively process behavioral biometrics, data-driven techniques such as ML could be employed.

REFERENCES

[1] Google. *Android Developers*. Accessed: Mar. 1, 2023. [Online]. Available: https://developer.android.com/guide/topics/sensors/

[2] A. Nahapetian, "Side-channel attacks on mobile and wearable systems," in *Proc. 13th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2016, pp. 243–247.

[3] R. Spreitzer, V. Moonsamy, T. Korak, and S. Mangard, "Systematic classification of side-channel attacks: A case study for mobile devices," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 465–488, 1st Quart., 2017.

[4] S. Shen, H. Wang, and R. Roy Choudhury, "I am a smartwatch and i can track my user's arm," in *Proc. 14th Annu. Int. Conf. Mobile Syst., Appl., Services*, Jun. 2016, pp. 85–96.

[5] M. Guerar, M. Migliardi, F. Palmieri, L. Verderame, and A. Merlo, "Securing PIN-based authentication in smartwatches with just two gestures," *Concurrency Comput., Pract. Exper.*, vol. 32, no. 18, p. e5549, Sep. 2020.

[6] B. Li, W. Wang, Y. Gao, V. V. Phoha, and Z. Jin, "Wrist in motion: A seamless context-aware continuous authentication framework using your clickings and typings," *IEEE Trans. Biometrics, Behav., Identity Sci.*, vol. 2, no. 3, pp. 294–307, Jul. 2020.

[7] L. Pucci, E. Testi, E. Favarelli, and A. Giorgetti, "Human activities classification using biaxial seismic sensors," *IEEE Sensors Lett.*, vol. 4, no. 10, pp. 1–4, Oct. 2020.

[8] A. Maiti, O. Armbruster, M. Jadliwala, and J. He, "Smartwatch-based keystroke inference attacks and context-aware protection mechanisms," in *Proc. 11th ACM Asia Conf. Comput. Commun. Secur.*, 2016, pp. 795–806.

[9] H. Wang, T. T.-T. Lai, and R. Roy Choudhury, "MoLe: Motion leaks through smartwatch sensors," in *Proc. 21st Annu. Int. Conf. Mobile Comput. Netw.*, Sep. 2015, pp. 155–166.

[10] X. Liu, Z. Zhou, W. Diao, Z. Li, and K. Zhang, "When good becomes evil: Keystroke inference with smartwatch," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2015, pp. 1273–1285.

[11] C. Wang, X. Guo, Y. Wang, Y. Chen, and B. Liu, "Friend or foe? Your wearable devices reveal your personal PIN," in *Proc. 11th ACM Asia Conf. Comput. Commun. Secur.*, May 2016, pp. 189–200.

[12] E. Miluzzo, A. Varshavsky, S. Balakrishnan, and R. R. Choudhury, "Tapprints: Your finger taps have fingerprints," in *Proc. 10th Int. Conf. Mobile Syst., Appl., Services*, Jun. 2012, pp. 323–336.

[13] E. Owusu, J. Han, S. Das, A. Perrig, and J. Zhang, "Accessory: Password inference using accelerometers on smartphones," in *Proc. 12th Workshop Mobile Comput. Syst. Appl.*, 2012, pp. 1–6.

[14] C. Yue, "Sensor-based mobile web fingerprinting and cross-site input inference attacks," in *Proc. IEEE Secur. Privacy Workshops (SPW)*, May 2016, pp. 241–244.

[15] R. Zhao, C. Yue, and Q. Han, "Sensor-based mobile web cross-site input inference attacks and defenses," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 1, pp. 75–89, Jan. 2019.

[16] A. Alzubaidi and J. Kalita, "Authentication of smartphone users using behavioral biometrics," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 1998–2026, 3rd Quart., 2016.

[17] G. Li and P. Bours, "A novel mobilephone application authentication approach based on accelerometer and gyroscope data," in *Proc. Int. Conf. Biometrics Special Interest Group (BIOSIG)*, Sep. 2018, pp. 1–4.

[18] C. Bo, L. Zhang, X.-Y. Li, Q. Huang, and Y. Wang, "SilentSense: Silent user identification via touch and movement behavioral biometrics," in *Proc. 19th Annu. Int. Conf. Mobile Comput. Netw.*, New York, NY, USA: Association for Computing Machinery, 2013, pp. 187–190, doi: 10.1145/2500423.2504572.

[19] J. Yang, Y. Li, and M. Xie, "MotionAuth: Motion-based authentication for wrist Worn smart devices," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops*, Mar. 2015, pp. 550–555.

[20] A. Lewis, Y. Li, and M. Xie, "Real time motion-based authentication for smartwatch," in *Proc. IEEE Conf. Commun. Netw. Secur. (CNS)*, Oct. 2016, pp. 380–381.

[21] M. A. S. Mondol, I. A. Emi, S. M. Preum, and J. A. Stankovic, "User authentication using wrist mounted inertial sensors," in *Proc. 16th ACM/IEEE Int. Conf. Inf. Process. Sensor Netw.*, Apr. 2017, pp. 309–310.

[22] S. Mare, A. M. Markham, C. Cornelius, R. Peterson, and D. Kotz, "ZEBRA: Zeuiro-effort bilateral recurring authentication," in *Proc. IEEE Symp. Secur. Privacy*, May 2014, pp. 705–720.

[23] M. Ehatisham-Ul-Haq, J. Loo, K. Shuang, S. Islam, U. Naeem, and Y. Amin, "Authentication of smartphone users based on activity recognition and mobile sensing," *Sensors*, vol. 17, no. 9, p. 2043, Sep. 2017.

[24] S. Yao, S. Hu, Y. Zhao, A. Zhang, and T. Abdelzaher, "DeepSense: A unified deep learning framework for time-series mobile sensing data processing," in *Proc. 26th Int. Conf. World Wide Web*, Apr. 2017, pp. 351–360.

[25] U. Mahbub, J. Komulainen, D. Ferreira, and R. Chellappa, "Continuous authentication of smartphones based on application usage," *IEEE Trans. Biometrics, Behav., Identity Sci.*, vol. 1, no. 3, pp. 165–180, Jul. 2019.

[26] J. Kolberg, M. Grimmer, M. Gomez-Barrero, and C. Busch, "Anomaly detection with convolutional autoencoders for fingerprint presentation attack detection," *IEEE Trans. Biometrics, Behav., Identity Sci.*, vol. 3, no. 2, pp. 190–202, Apr. 2021.

[27] E. Von Zezschwitz, A. De Luca, B. Brunkow, and H. Hussmann, "SwiPIN: Fast and secure PIN-entry on smartphones," in *Proc. 33rd Annu. ACM Conf. Hum. Factors Comput. Syst.*, Apr. 2015, pp. 1403–1406.

[28] M. Guerar, L. Verderame, M. Migliardi, and A. Merlo, "2GesturePIN: Securing PIN-based authentication on smartwatches," in *Proc. IEEE 28th Int. Conf. Enabling Technol., Infrastruct. Collaborative Enterprises (WETICE)*, Jun. 2019, pp. 327–333.

[29] T. Van Nguyen, N. Sae-Bae, and N. Memon, "DRAW-A-PIN: Authentication using finger-drawn PIN on touch devices," *Comput. Secur.*, vol. 66, pp. 115–128, May 2017.

[30] J. Kim and P. Kang, "Draw-a-deep pattern: Drawing pattern-based smartphone user authentication based on temporal convolutional neural network," *Appl. Sci.*, vol. 12, no. 15, p. 7590, Jul. 2022.

[31] M. Nerini, E. Favarelli, and M. Chiani, "Augmented PIN authentication through behavioral biometrics," *Sensors*, vol. 22, no. 13, p. 4857, Jun. 2022.

[32] P. Bours and E. Masoudian, "Applying keystroke dynamics on one-time pin codes," in *Proc. 2nd Int. Workshop Biometrics Forensics*, Mar. 2014, pp. 1–6.

[33] E. Ivannikova, G. David, and T. Hamalainen, "Anomaly detection approach to keystroke dynamics based user authentication," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jul. 2017, pp. 885–889.

[34] B. Ayotte, M. Banavar, D. Hou, and S. Schuckers, "Fast free-text authentication via instance-based keystroke dynamics," *IEEE Trans. Biometrics, Behav., Identity Sci.*, vol. 2, no. 4, pp. 377–387, Jun. 2020.

[35] S. Panda, Y. Liu, G. P. Hancke, and U. M. Qureshi, "Behavioral acoustic emanations: Attack and verification of PIN entry using keypress sounds," *Sensors*, vol. 20, no. 11, p. 3015, May 2020.

[36] X. Liu, Y. Li, and R. H. Deng, "UltraPIN: Inferring PIN entries via ultrasound," in *Proc. ACM Asia Conf. Comput. Commun. Secur.* New York, NY, USA: Association for Computing Machinery, May 2021, pp. 944–957, doi: 10.1145/3433210.3453075.

[37] A. J. Aviv, B. Sapp, M. Blaze, and J. M. Smith, "Practicality of accelerometer side channels on smartphones," in *Proc. 28th Annu. Comput. Secur. Appl. Conf.* New York, NY, USA: Association for Computing Machinery, Dec. 2012, pp. 41–50.

[38] M. Mehrnezhad, E. Toreini, S. F. Shahandashti, and F. Hao, "TouchSignatures: Identification of user touch actions and PINs based on mobile sensor data via Javascript," *J. Inf. Secur. Appl.*, vol. 26, pp. 23–38, Feb. 2016.

[39] M. Mehrnezhad, E. Toreini, S. F. Shahandashti, and F. Hao, "Stealing PINs via mobile sensors: Actual risk versus user perception," *Int. J. Inf. Secur.*, vol. 17, no. 3, pp. 291–313, Jun. 2018.

[40] C. Shen, Y. Chen, Y. Liu, and X. Guan, "Adaptive human–machine interactive behavior analysis with wrist-Worn devices for password inference," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 12, pp. 6292–6302, Dec. 2018.

[41] S. Naval, A. Pandey, S. Gupta, G. Singal, V. Vinoba, and N. Kumar, "PIN inference attack: A threat to mobile security and smartphone-controlled robots," *IEEE Sensors J.*, vol. 22, no. 18, pp. 17475–17482, Sep. 2022.

[42] P. Markert, D. V. Bailey, M. Golla, M. Durmuth, and A. J. Aviv, "On the security of smartphone unlock PINs," *ACM Trans. Privacy Secur.*, vol. 24, no. 4, pp. 1–36, Sep. 2021, doi: 10.1145/3473040.

[43] A. Sarkisyan, R. Debbiny, and A. Nahapetian, "WristSnoop: Smartphone PINs prediction using smartwatch motion sensors," in *Proc. IEEE Int. Workshop Inf. Forensics Secur. (WIFS)*, Nov. 2015, pp. 1–6.

[44] Z. Xu, K. Bai, and S. Zhu, "TapLogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors," in *Proc. 5th ACM Conf. Secur. Privacy Wireless Mobile Netw.*, New York, NY, USA, Apr. 2012, pp. 113–124.

[45] C. Shen, S. Pei, Z. Yang, and X. Guan, "Input extraction via motion-sensor behavior analysis on smartphones," *Comput. Secur.*, vol. 53, pp. 143–155, Sep. 2015.

[46] L. Cai and H. Chen, "TouchLogger: Inferring keystrokes on touch screen from smartphone motion," in *Proc. 6th USENIX Conf. Hot Topics Secur.*, 2011, p. 9.
[47] C. M. Bishop, *Pattern Recognition and Machine Learning*. Berlin, Germany: Springer, Aug. 2006.
[48] J. Watt, R. Borhani, and A. K. Katsaggelos, *Machine Learning Refined*. Cambridge, U.K.: Cambridge Univ. Press, 2016.
[49] E. Testi, E. Favarelli, and A. Giorgetti, "Machine learning for user traffic classification in wireless systems," in *Proc. 26th Eur. Signal Process. Conf. (EUSIPCO)*, Rome, Italy, Sep. 2018, pp. 2040–2044.
[50] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[51] S. Amari and S. Wu, "Improving support vector machine classifiers by modifying kernel functions," *Neural Netw.*, vol. 12, no. 6, pp. 783–789, 1999.
[52] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 1, pp. 21–27, Jan. 1967.
[53] Google. *Google Colab*. Accessed: Mar. 1, 2023. [Online]. Available: https://colab.research.google.com/
[54] F. Pedregosa, S. Varoquaux, A. Gramfort, V. Michel, and B. Thirion, "SciKit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Dec. 2011.
[55] R. Song, Y. Song, Q. Dong, A. Hu, and S. Gao, "WebLogger: Stealing your personal PINs via mobile web application," in *Proc. 9th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Oct. 2017, pp. 1–6.

MATTEO NERINI (Graduate Student Member, IEEE) received the B.Sc. degree in electronics and telecommunications engineering and the M.Sc. degree in telecommunications engineering from the University of Bologna (Unibo), Italy, in 2018 and 2020, respectively, and the M.Sc. degree in communication technology from the Norwegian University of Science and Technology (NTNU), Norway, in 2020. He is currently pursuing the Ph.D. degree with the Wireless Communications and Signal Processing Laboratory, Imperial College London, London, U.K.

His research interests include wireless channel feedback, reconfigurable intelligent surfaces, and deep learning for wireless communications.

ELIA FAVARELLI (Member, IEEE) received the M.S. degree (magna cum laude) in electronics and telecommunications engineering for energy and the Ph.D. degree in structural and environmental health monitoring and management from the University of Bologna, Italy, in 2018 and 2021, respectively.

He is currently a Research Fellow with the Department of Electrical, Electronic, and Information Engineering "Guglielmo Marconi" (DEI). His research interests include telecommunications, machine learning (ML), deep learning (DL), anomaly detection techniques for structural health monitoring (SHM), and tracking. His current research focuses on ML algorithms applied to anomaly detection, structural health monitoring, and tracking.

MARCO CHIANI (Fellow, IEEE) received the Dr.-Ing. degree (summa cum laude) in electronic engineering and the Ph.D. degree in electronic and computer engineering from the University of Bologna, Italy, in 1989 and 1993, respectively.

He is currently a Full Professor of telecommunications with the University of Bologna. Since 2003, he has been a frequent visitor at the Massachusetts Institute of Technology (MIT), Cambridge, where he currently holds a Research Affiliate appointment. His research interests include the areas of information theory, wireless systems, statistical signal processing, and quantum information.

Dr. Chiani served as the Elected Chair for the Radio Communications Committee of the IEEE Communication Society, from 2002 to 2004. He served as an Editor of wireless communication for IEEE TRANSACTIONS ON COMMUNICATIONS from 2000 to 2007.

Open Access funding provided by 'Alma Mater Studiorum - Università di Bologna' within the CRUI CARE Agreement