My(o) Armband Leaks Passwords: An EMG and IMU Based Keylogging Side-Channel Attack

MATTHIAS GAZZARI*, Technical University of Darmstadt, Germany
ANNEMARIE MATTMANN*, Technical University of Darmstadt, Germany
MAX MAASS, Technical University of Darmstadt, Germany
MATTHIAS HOLLICK, Technical University of Darmstadt, Germany

Wearables that constantly collect various sensor data of their users increase the chances for inferences of unintentional and sensitive information such as passwords typed on a physical keyboard. We take a thorough look at the potential of using electromyographic (EMG) data, a sensor modality which is new to the market but has lately gained attention in the context of wearables for augmented reality (AR), for a keylogging side-channel attack. Our approach is based on neural networks for a between-subject attack in a realistic scenario using the Myo Armband to collect the sensor data. In our approach, the EMG data has proven to be the most prominent source of information compared to the accelerometer and gyroscope, increasing the keystroke detection performance. For our end-to-end approach on raw data, we report a mean balanced accuracy of about 76 % for the keystroke detection and a mean top-3 key accuracy of about 32 % on 52 classes for the key identification on passwords of varying strengths. We have created an extensive dataset including more than 310 000 keystrokes recorded from 37 volunteers, which is available as open access along with the source code used to create the given results.

CCS Concepts: • **Security and privacy**; • **Human-centered computing** → **Ubiquitous and mobile devices**; • **Computing methodologies** → *Neural networks*; *Supervised learning*;

Additional Key Words and Phrases: Keylogging, Keystroke Inference, Side-channel Attacks, Privacy, Electromyography, EMG, Wearables, Deep Learning, Time Series Classification

ACM Reference Format:
Matthias Gazzari, Annemarie Mattmann, Max Maass, and Matthias Hollick. 2021. My(o) Armband Leaks Passwords: An EMG and IMU Based Keylogging Side-Channel Attack. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 5, 4, Article 157 (December 2021), 24 pages. https://doi.org/10.1145/3494986

1 INTRODUCTION

Wearables are designed to gather and process information for the user's benefit. They are also designed to be worn for most if not all of the day. Thus, they will inevitably be present when the wearers engage in sensitive activities, gathering data that may contain traces of these activities. Human-centric sensors embedded in ubiquitous wearable devices such as smartwatches have been shown to pose risks for keylogging side-channel attacks to snoop on human input on physical keyboards [16, 18, 26, 36], keypads [16, 17, 35], or virtual keyboards on touch screens [19, 20, 28]. Successful attacks based on this data can leak confidential information such as text typed on a keyboard by matching it against English words from dictionaries [16, 18, 36]. Preliminary work indicates that such approaches could also be used to infer unstructured text such as passwords [26].

*Both authors contributed equally to this research.

Authors' addresses: Matthias Gazzari, Technical University of Darmstadt, Secure Mobile Networking Lab, Darmstadt, Germany, mgazzari@seemoo.tu-darmstadt.de; Annemarie Mattmann, Technical University of Darmstadt, Secure Mobile Networking Lab, Darmstadt, Germany, amattmann@seemoo.tu-darmstadt.de; Max Maass, Technical University of Darmstadt, Secure Mobile Networking Lab, Darmstadt, Germany, mmaass@seemoo.tu-darmstadt.de; Matthias Hollick, mhollick@seemoo.tu-darmstadt.de, Technical University of Darmstadt, Secure Mobile Networking Lab, Darmstadt, Germany.

Most prior work focuses on inertial measurement unit (IMU) sensor data, particularly on the accelerometer and the gyroscope. However, over the past years, more and more wearables have entered the consumer market equipped with a multitude of different sensors like EMG sensors. Devices that record EMG data can capture the muscle activity responsible for moving individual fingers and thus offer a novel attack vector by providing opportunities for a more fine-grained detection and differentiation of typing different keys on the keyboard. Typically, those wearables also include an IMU, providing the opportunity to complement attacks based on EMG with information about the coarse movements of the arm [39]. Given these considerations and the results of prior research, we suggest an approach utilizing both IMU and EMG sensor data both in isolation and combination to evaluate the importance of the EMG data, which may imply a greater security risk of using wearables equipped with both sensors compared with those that only offer an IMU sensor.

Facebook recently proposed a personalized virtual keyboard based on a custom EMG wrist band for extended reality (XR) applications [7], using the sensor data to interpret typing of fingers on an imaginary keyboard. Their work is similar to prior research of using the EMG sensors of the Myo armband for a virtual T9 keyboard by inferring nine different finger gestures [9], as well as to prior work of using a custom EMG sensor exploring text entry with 16 keys of the left hand on a keyboard printed on a paper and a physical keyboard. These kinds of approaches open up new possibilities for user interaction, but also highlight the possible impacts of abusing EMG as well as IMU data as a potential side-channel for snooping on user input.

Compared to a keylogger installed on a single computing device by a malicious actor, inferring keystrokes from sensor data collected by wearables opens up opportunities to attack not only a single computer but as many as a victim is interacting with while wearing these devices. Like smartwatches, such wearables may be connected to mobile devices or operate independently, for example, in an AR context. For reasons of convenience or simply unawareness of potential consequences, these wearables may not be taken off when engaging in sensitive activities such as typing on a physical keyboard. Thus, sensor data of such wearables may reveal everything that has been typed on a personal, work or any other device. This is especially the case for wearables based on EMG sensors, as those typically involve a more time-consuming setup. Furthermore, this poses an issue in scenarios in which users depend on their wearables, as is the case for prostheses based on EMG [34] and medical monitoring [6, 21].

In order to gain access to the sensor data, an attacker could use a malicious app installed on a single device connected to the wearable, enabling the adversary to infer keystrokes from every device the victim is typing on. This app could either be installed by the adversary or by the victims themselves, potentially disguised as a harmless application for using the wearable in a legitimate use-case. Alternatively, with sensor data stored in the cloud, an adversary could access the sensor data of multiple victims directly, without having to interact with the victims. Such cloud storage may be legitimately used by a service providers of the wearable to improve their service, but offers a larger attack surface, possibly exposing the sensor data of all users of their service. Furthermore, users may also willingly share their data with third parties, unaware what could be inferred from sensors like EMG. They may underestimate the risk due to these devices being advertised for more coarse types of input like controlling certain applications with gestures.

To the best of our knowledge, we are among the first to look into keylogging side-channel attacks based on forearm EMG and IMU sensor data for inferring unstructured text like passwords. One of the EMG sensors that have been sold on the consumer market is the *Myo armband*, a wearable device worn on the lower arm and connected to a computer via Bluetooth. It records electromyographic data, i.e. muscle data, and IMU, i.e. accelerometer, gyroscope and magnetometer data, to recognize gestures and finger movement. Prior to us, Zhang et al. [39] have used the EMG data of the Myo armband for keystroke detection and finger differentiation on individual users (*within-subject*). However, their approach is limited to touch typists in a controlled setting and does not support generalization between users.

In our work, we use the raw EMG and IMU (accelerometer and gyroscope) data from two Myo armbands worn on both lower forearms to infer keystrokes typed on a physical keyboard in a *between-subject* scenario to infer keystrokes from previously unknown users. We assume an adversary who has access to either the EMG, accelerometer or gyroscope data (or a combination thereof) of the victim, for example, through the use of a malicious application installed on the device to which the Myo armband is connected. The adversary trains a neural network in advance with a custom dataset of labeled sensor data, based on the input of typists other than the victim. In contrast to within-subject approaches, the adversary does not need to create an individual training set by observing the victim prior to executing the attack. Instead, the adversary can use the trained model to infer the typing of the victim based only on the gathered sensor data, leveraging the generalization capabilities of the neural network approach. In doing so, the adversary seeks to gain information on what has been typed on a physical keyboard by the victim, including text and passwords. By hiring as many different typists as possible for the original dataset, the adversary can further improve the performance.

We compare the influence of each individual sensor with the performance of a combination of all sensors to evaluate the potential security impact of adding EMG sensors to wearable devices. We try to mimic the natural environment of the typists, including its changes and the typists' natural behavior, by setting minimal constraints on the typists to gain representative results and evaluate a realistic scenario for a keylogging side-channel attack. Furthermore, we include touch and hybrid typists in our analysis. In order to investigate the influence of different passwords, we include test data for passwords of different strengths in our evaluation, ranging from popular but insecure to randomly generated passwords and long passwords based on multiple words. To infer the keystrokes, we apply and compare four different supervised end-to-end machine learning models for (1) detecting the presence of keystrokes and (2) identifying the key which has been typed. One of these four models is the state of the art end-to-end neural network model for time series classification taken from Ismail Fawaz et al. [14]. As part of this work, we publish the first multimodal EMG and IMU dataset, including the timestamped keystroke events, as well as the source code for training the machine learning models along with the respective analyses to support further research in this area.

Thus, our main contributions are the following:

- We collect and release the first large open access dataset composed of 310 566 keystrokes (about 34 h of typing) in total with the corresponding EMG and IMU data of two Myo armbands, recorded from 37 participants in a realistic scenario.
- We are the first to implement a between-subject approach to generalize between users for keystroke inference of words as well as unstructured text like passwords, based on EMG sensor data, and to evaluate the importance of EMG compared to IMU data in this context.
- We evaluate four different, end-to-end trained neural network architectures for keystroke detection and identification in order to compare their performance and avoid a bias introduced by a certain model structure. This is the first application of end-to-end neural networks to implement a human-centric sensor-based keylogging side-channel attack.

This paper is structured as follows: In Section 2, we outline the study design and describe the resulting dataset. Section 3 provides details on our classification approach, including the involved data pre-processing, the model descriptions, the training process and the post-processing. We also include the results of the keystroke detection and key identification classifiers. Section 4 includes a short discussion of the results and lists the limitations of our approach. In Section 5, we give a comparison to related work. Section 6 provides a short conclusion for our work.

2 DATA STUDY

Inferring keystrokes from the data recorded by wearable devices involves two steps: *Keystroke detection* and *key identification* [23]. Keystroke detection refers to detecting the presence and timing of any keystroke by

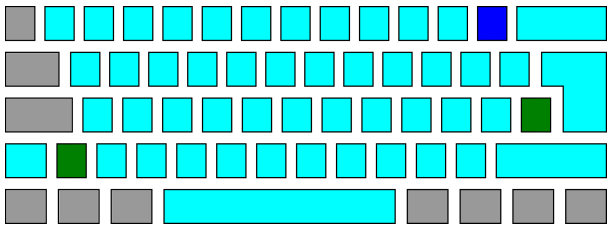

Fig. 1. The 52 keys of an ISO keyboard considered for the classification. Grey keys are not considered, green keys are only included in recordings with a German layout and the blue key is only used in recordings with a US layout.

determining the time of either a press event or a release event, or by inferring the key state at any given time. Key identification refers to determining which physical key was pressed given a detected keystroke. The keystroke detection could be preceded by a typing activity recognition, i.e. detecting those sequences within the constant stream of data recorded by a wearable device that actually contain keystrokes. However, this would require recording multiple daily activities along with typing on a keyboard, which is out of scope for this project.

In this work, we focus on keystroke detection and key identification. Thus, for recording the data we need in order to train and test our classifiers, we assume a typing scenario in which the participants either sit at or stand in front of a table, ready to write on the keyboard. At the same time, we do not prohibit movements by the participants that would usually occur while typing on a keyboard, such as drinking or moving the body. In doing so, we require our keystroke detection models to distinguish between typing and non-typing movements within a typing scenario.

2.1 Data

The data we record includes the *sensor data* recorded by the two Myo armbands, the *ground truth* recorded by the laptop while typing on the external physical keyboard and the *meta data* associated with each recording. The dataset is available as open access at https://doi.org/10.5281/zenodo.5594651 and the source code used for recording this dataset is available as free software at https://github.com/seemoo-lab/myo-keylogging.

The sensor data includes both the raw EMG and IMU (accelerometer, gyroscope and magnetometer) data, recorded at 200 Hz and 50 Hz respectively. As the Myo armband does not attach timestamps to the data, these are added as soon as the Bluetooth packets arrive on the laptop. The ground truth consists of timestamped keystrokes recorded by the laptop while typing on the physical keyboard. For our work we consider a total of 52 keys of an ISO keyboard, with either a German (51 keys) or a US (50 keys) logical keyboard layout as shown in Fig. 1. Both, the sensor and the keystroke data, are used to train and evaluate the classifiers in a supervised learning fashion. The keystrokes refer to physical keys (and not the assigned characters) to ensure that our approach is invariant against logical keyboard layout changes. The meta data includes information such as a participant's unique identifier and the type of the recording.

One *recording session* is comprised of multiple *recordings*, each of which is associated with a certain task. This allows for a simple rerun of a task if a problem occurs during the recording and for easily switching between task types to make the recording more varied for the participants. One participant may or may not take part in multiple recording sessions. All recording sessions are represented by unique identifiers in the meta data and all recordings have unique identifiers in regard to the respective recording session.

Fig. 2. The setup for the data collection showing the two Myo armbands in the front.

2.2 Experimental Setup

As part of our attempt to create realistic and diverse recording settings, the data study is carried out in multiple locations (including different offices, personal working spaces and some participant's living rooms), but the setup as shown in Fig. 2 is the same in all cases.

The white Myo armband is worn on the left and the black one on the right arm, adjusted according to the official guidelines[1]. The participants are encouraged to sit (or stand) and move as they usually do while typing. The keyboard used for the data collection is a TADA68 with mechanical switches. We support both the German and US international keyboard layout, in both cases as an ISO variant, asking the participants prior to the recording which layout they prefer.

2.3 Study Design

We design two distinctive data recording schemes, the first of which provides the *training data* for the classifiers and the second of which provides the *test data*. The training and test data are recorded on completely separate occasions to account for biases associated with position changes of the armband. In addition to that, both datasets contain data from participants who have only taken part in the training or test data recording respectively, in order to test the applicability of our approach on previously unknown persons. Each recording scheme consists of multiple different tasks of different types, all but one of which prompt the user with groups of characters or words to copy. During all recording sessions, we encourage the participants on multiple occasions to move or take a break and continue typing in a different position for diversification.

The training data scheme includes the following task types:

(1) *Text*: This contains extracts copied from Wikipedia articles, selected such that they contain each key in consideration, as shown in Fig. 1, at least 10 times in total.
(2) *Pangram*: Also copied from Wikipedia, each of these pangrams contain each letter of the alphabet (i.e. a subset of the keys in consideration) at least once.

[1]Instructions by the manufacturer state that the Myo armband is supposed to be worn on the thickest part of the lower arm (below the elbow), with the USB port facing the wrist and the LED positioned on top of the arm (when stretched out).

(3) *Random*: This contains pseudo-random groups of two, three or five different characters, generated by shuffling the set of characters corresponding to the keys in consideration such that each key appears exactly twice throughout this task. The rationale behind this task is to ensure that we get enough samples for each possible key, uniformly distributed within this task.
(4) *Random (memorized)*: The same as the above with groups of two, three or four characters. However, this time the characters disappear once the participant starts typing, requiring them to remember the characters and preventing them from copying the characters one after the other. This design is inspired by the observation that users tend to type passwords in chunks of three to four quickly typed characters with a small pause in between [29].
(5) *Game*: A console-based game which involves a small subset of the characters in consideration, included for diversification. This is the only task which does not require copying text.

These tasks are meant to cover most of the types of input made on a physical keyboard in order to be able to infer text as well as passwords, the latter of which can either be closer to a random combination of characters or to text. For all tasks except the game the participants see what they have typed and are able to correct mistakes using the backspace key.

For the test data scheme, we develop four different task types:

(1) *Insecure*: Passwords which are randomly selected from 250 of the top 1000 passwords used [22], e.g. *iloveyou*.
(2) *xkcd*: Passphrases inspired by xkcd [24] and consisting of six randomly chosen words from the Electronic Frontier Foundation list [1], e.g. *hatchery gratified drinking tiling precinct anywhere*.
(3) *pwgen*: Passwords with a length of eight characters generated by pwgen, a password generator aimed at creating easy to type and easy to remember semi-random passwords, e.g. *ei9Aemac*.
(4) *Random*: Randomly generated passwords with a length of eight characters, e.g. *p5nkmq'y*.

These tasks are meant to cover both frequently used and insecure passwords, as well as passwords generated according to different guidelines such as securing a password by length while making it memorable (xkcd), using a password which is pseudo-random but easier to type and memorable (pwgen) and using a random password. Thus, we can evaluate and compare the performance of our classifiers on common but inherently different types of passwords of different strengths. To better mimic normal password input behavior, the participants receive no feedback of what they type during entry (feedback is only given after hitting return) but they are allowed to correct mistakes using the backspace key.

We require the participants to type each password correctly four times (if they do make a mistake the entry does not count towards this) to familiarize them with the password and to simulate the entry of a known password to a certain extent. Note that correcting mistakes while typing is possible without triggering an additional repetition, as the check for correctness is only made after the password entry is confirmed by pressing return, as it would be the case for a real password entry. We set this limit of four repetitions because early tests have shown that requiring a higher number of correct entries can be upsetting for the participants. For future studies, we suggest including repetitions of the same password spread across the whole recording to circumvent this problem.

The data study was approved by the ethics commission of TU Darmstadt (EK 24/2018). Vouchers were raffled off among the participants taking part in the longer recording of the training data.

2.4 Evaluation

A total of 37 volunteers took part in our data study (about one third female, two third male). Most were students or employees of the university, ranging in age between 18 and 74, with two thirds of the participants between the age of 25 to 34.

We distinguish between two typing styles: Touch typing and hybrid typing. The latter includes hunt-and-peck typists as well as typists with an incomplete touch typing training (e.g. who use less than ten fingers).

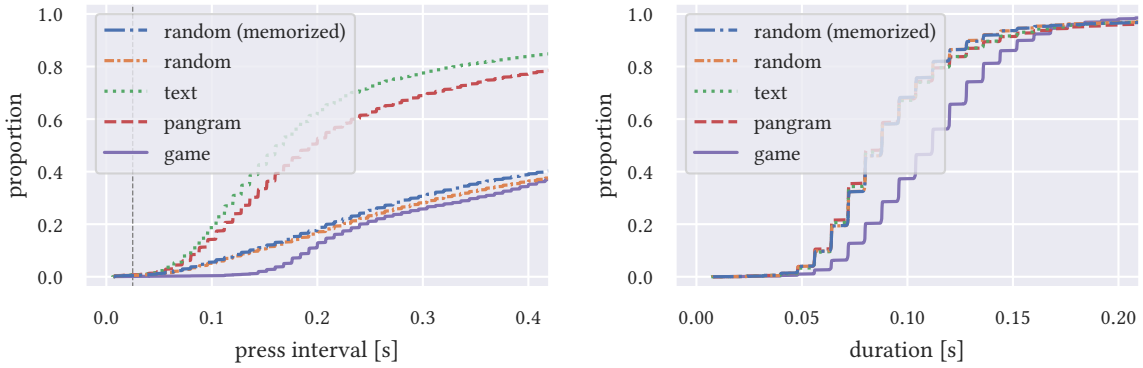

Fig. 3. Cumulative distribution of press-press intervals for different task types (left) and keystroke duration, i.e. times between press and release of a key (right), in the training data. The vertical line on the left marks the minimum peak distance for the peak detection.

The categorization is based on self-reporting in combination with our observations during the recordings. We discerned that even trained touch typists often diverge from the guidelines for the finger-to-key mapping, an observation already recorded in the literature [8].

Typing speeds reported by the participants range from about 86 to 500 keys per minute. Most participants use a keyboard on a daily basis. The mean typing speed per person for the training data is 65 to 231 keys per minute. For the test data, it is 125 to 235 keys per minute. This is significantly lower than the reported typing speed, which is mostly due to the random character tasks for which the average typing speed is 71 keys per minute across all participants (compared to 241 keys per minute for the text-based tasks). Other influences reducing the typing speed may be the unfamiliar keyboard or the reduced look-ahead due to the limited number of characters presented to the participant at a time (one sentence or pangram for the text-based tasks, one chunk of characters for the random tasks and one password for each task type in the test data scheme). According to prior research, look-ahead and motor preparation generally have a significant effect on typing speed [8, 27]. We also observe a difference in the average typing speed for touch typists with 161 keys per minute and hybrid typists with 134 keys per minute for the training data with an even larger difference for the test data of 184 keys per minute for touch typists and 164 keys per minute for hybrid typists.

Fig. 3 shows the difference in typing patterns between the different task types in the training data regarding the press-press intervals between the press events of successive keystrokes (left) and the duration of the keystrokes from press to release (right). In particular, it shows the diversification gained by the game task. We added the random (memorized) task hoping to get a more fluent typing pattern if the participants had to remember the characters, but the results show no significant difference in typing speed between the two random task types. This is most likely due to some participants having difficulty remembering the characters while most participants did not copy them character by character even if they remained visible. In fact, disabling the feedback of what the participants type seems to have a larger effect in this case, as many participants tend to check if they copied the characters correctly, which disrupts the typing flow.

The *class skew* refers to the number of samples of a given class as compared to the number of samples of the other classes representing the imbalance inherent to the dataset. For a keystroke detection on keypress events, the positive class refers to the single point in time of an actual keypress and the negative class covers every step

in time that does not contain a keypress. In our training data, the mean class skew is 98.7 % for the negative class compared to the positive class. For the test data, the mean class skew is 98.3 %. Thus, the dataset is highly imbalanced towards the negative class. Typing speed and class skew negatively correlate with each other, but the changes are insignificant given the overall bias. The mean class skew for the fastest typist is 99.5 % on the training data and 98.8 % on the test data. For the slowest typist it is 98.1 % on the training data and 97.7 % on the test data. The class skew for a key identification depends on the number of samples given for each key. In our dataset, this is uniform for the random tasks but mostly underlies language characteristics for the text-based tasks. Space is the most frequent key.

Each participant of the data study took part in either the test or the training data recording scheme or in both of them, resulting in a total of 29 training and 17 test data recording sessions, six of which were recorded with the same participants for training as well as test data. Furthermore, the training data contains three additional recordings by two participants. Thus, we have a total of 37 participants for a total of 46 recording sessions.

For the training data scheme, we include a total of 24 recordings of separate tasks (one game, five pangram and six of every other task type) with more than 8000 keys recorded from each participant on average. The training data includes 261 962 keystrokes in total. For the test data scheme, we have 12 tasks (three for each type, each including two passwords repeated at least four times). Two thirds of the tasks include the same passwords for multiple participants, one third includes a random selection of passwords different for each participant. The dataset includes 48 604 keystrokes in total. On average, a single data recording took one hour for the training data and 30 minutes for the test data scheme. This includes small breaks and movements made by the participant during a task recording. In total, this amounts to 310 566 keystrokes and about 34 h of typing.

In order to compare the feasibility of a within-subject and a between-subject approach, we apply dynamic time warping (DTW) to repeating sequences representing the same, correctly typed words within the training data recordings and compare the resulting distances across different recordings. Fig. 4 shows the results for one word which appears up to five times in the tasks generated for the German keyboard layout during one recording session. It highlights that the distances between two recordings of the same session are more similar than those between different recording sessions, which show a similar distance to the recordings of different participants. This implies that a within-subject approach may only work well if the model is retrained whenever the Myo armband is taken off and put on again, which is impractical for an unsupervised approach and infeasible for a supervised approach. Thus, a between-subject approach seems best even when focusing on single users.

3 CLASSIFICATION

We formulate the problem of inferring keystrokes on sensor data (recorded within a typing scenario) as a two-step *supervised classification* in a between-subject scenario, enabling us to infer keystrokes belonging to previously unknown persons. Our first step is the keystroke detection, a binary classifier predicting whether a sample belongs to a keystroke or not by encoding the keypress event. Due to the high class skew inherent to this encoding, as shown in Section 2.4, we considered predicting the key state instead, i.e. each moment in time in which a key is pressed. However, this approach has its own disadvantages, most notably that it would require a multi-label approach to predict overlapping keystrokes, e.g. as required for modifier keys like the shift key.

The second step, the key identification, is a multi-class classifier which predicts the actual physical key on the samples predicted to belong to a keypress. We have 52 classes in total representing the keys on the physical keyboard as depicted in Fig. 1.

The input to both classifiers is a segment of sensor data with a size of 150 ms before and 100 ms after the sample, which coincides with the actual keypress. The overall segment size of 250 ms is similar to the one used in [39] and the size and position of the keypress within the segment are supported by the findings of [5]. They report that bursts of EMG activity in the forearm during a keystroke occur about 150 ms before and about 50 ms after

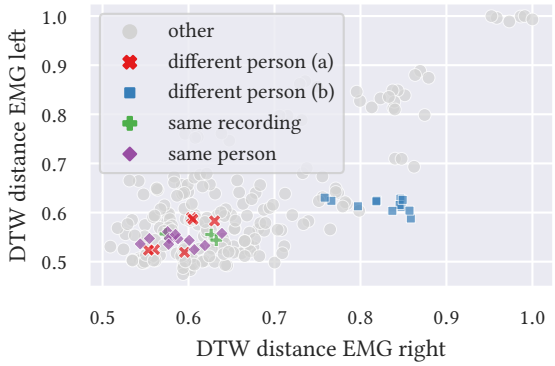

Fig. 4. The DTW distance between different recordings in the training data of the word *Quetzalcoatl* of the same participant recorded in the same (green) and a different recording session (violet) and of different participants (red and blue). The recordings made during the same session are closest to each other but the recordings made in a different session partly overlap those by the participant marked in red. On the other hand, these recordings do not overlap at all with those of the participant marked in blue. These participant's data is highlighted as an example, the other recordings are shown in grey.

bottoming out the key switch, which is very close to the time point of the key switch actuation point. Similarly, they report that the downward movement of the raised finger starts about 83 ms before fully depressing the key switch.

As a consequence of our approach, both classifiers are state-less and therefore classify segments without considering past or future predictions. This is necessary for the key identification as we are aiming at classifying unstructured text like passwords and therefore must not consider the likelihood of certain character sequences. However, for the output of our keystroke detection, we apply a post-processing step, which takes the timing of keystrokes into account to eliminate false positives by enforcing a minimum time distance between consecutive keystrokes.

In summary, our proposed system consists of a simple pre-processing of the sensor data, which is described in the following, followed by the binary classification for keystroke detection, the post-processing of the binary results, and finally the multi-class classification for key identification. Every design choice, including the hyperparameter optimization of the classifiers, as well as the choice of the parameters for the pre- and post-processing steps are based on the training data. The source code for training the proposed models and reproducing our final results is available as free software at https://github.com/seemoo-lab/myo-keylogging.

3.1 Pre-processing

Each armband contains eight EMG sensors sampled at 200 Hz and an IMU containing an accelerometer, gyroscope and a magnetometer sampled at 50 Hz. For this work, we do not consider the magnetometer as the provided rotation quaternion is deemed unsuitable for our task. Since the armband does not provide timestamps for sensor values, we determine those by estimating the sampling frequency for each modality separately.

We tested filtering the EMG sensor data by applying a second-order Butterworth high-pass filter with a cutoff frequency of 20 Hz following best practices [4]. However, we decided to exclude this filter as we observed a performance degradation. Similarly, the performance degraded slightly when changing the cutoff frequency to 10 Hz as in similar work with the Myo armband [9], as well as when testing a second-order infinite impulse response (IIR) notch filter at 50 Hz ($Q = 25$) to remove the power-line noise (PLI). Thus, we do not apply any

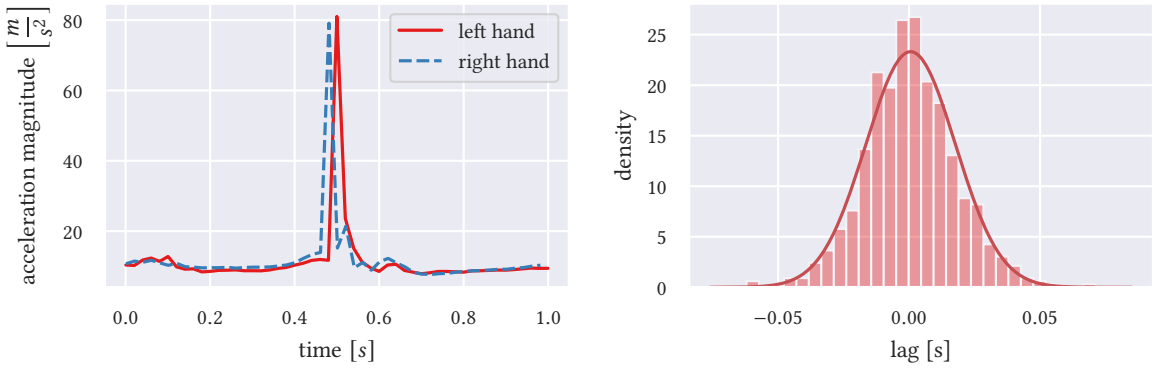

Fig. 5. Left shows an example of the accelerometer magnitude values of both Myo armbands during the joint high acceleration event (clapping the hands). Right shows the distribution of the peak-to-peak time differences between those values for all training samples. The curve in the right picture is a normal distribution fitted to the data points.

filtering on the data. Instead, we rely on the neural networks to discern relevant parts of the raw data, enabling an end-to-end solution.

We fuse the sensor data of a single armband by resampling the IMU data to match the higher EMG sample rate. An offset between the armbands' sensor values is calculated by recording a joint high acceleration event, namely the clapping of both hands as exemplarily shown on the left in Fig. 5. This latency value is then used to align the data of both armbands, followed by a resampling of both data streams to a common sampling rate. Note that this alignment cannot be applied when considering a realistic attack scenario. Though the absolute lag between both devices is low, being considerably less than 50 ms for almost every sample, as can be seen in the histogram on the right in Fig. 5. We assume that in practice, this makes no difference, as we did not observe a performance degradation when disabling this sensor alignment.

3.2 Neural Networks

We train four different neural networks, each of which has shown promising results when applied to problems that are similar to ours in one way or another while being based on a different approach than the other models. All models are trained once for binary and once for multi-class classification using the training data in our dataset. All classifiers are designed as end-to-end neural networks: We skip feature engineering, training the classifiers on raw data and thus relying on the neural networks to find good feature representations relevant for the given task. However, in order to connect the binary with the multi-class classifiers, we require an intermediate step of post-processing the binary output with a peak detector in order to remove candidates of predictions which likely belong to the same keystroke.

Each model contains a final *output layer*, linearly mapping the output of the previous stages onto either a single neuron (binary classification) or on a number of neurons matching the number of required classes (multi-class). For binary classification, we apply the sigmoid function on the output layer, whereas for multi-class classification we use the softmax function. Through hyperparameter optimization on both types of classification problems, we did not find notable differences justifying different sets of hyperparameters. Note that simpler prototypes based on convolutional neural network (CNN) showed greater specialization towards a single type of classification when comparing the performance with a single hyperparameter set, though these attempts were consistently

outperformed by the four models described below. More details for the hyperparameters of each model are given in Appendix B.

3.2.1 TSC ResNet11. This neural network is a close recreation of an 11-layer ResNet [11] adaption proposed [37] and reviewed [14] as a strong off-the-shelf choice in univariate, as well as multivariate time series classification (TSC) and is, therefore, a natural fit to be applied to our problem. Our main change to the original architecture is the introduction of so-called grouped convolutions [13], essentially providing independent paths through the network for each of the 28 sensor input channels. In our experience, this single architectural change boosted the performance significantly.

3.2.2 CWT ResNet18. This neural network is a combination of a batch normalization layer, followed by a continuous wavelet transformation (CWT) and an 18-layer ResNet network [11] with full pre-activation residual units [12] inspired by a similar approach for hand gesture recognition based on the EMG sensor data from a Myo armband [3]. The basic idea of this approach is to use the CWT ($\delta_j = 0.125$, $\delta_t = 0.005$) with a Morlet wavelet ($\omega_0 = 6$ as suggested in [31]) to transform the one-dimensional time series representation of each individual sensor channel into a two-dimensional representation, namely the wavelet power spectrum [31], and apply a popular image recognition model like ResNet to perform the actual classification. Our main change to the basic ResNet architecture is the removal of the early max pooling layer, which could be explained by having a significantly smaller input size: The wavelet power spectrum for each segment consists of 38x50 data points compared to the original input of a 224x224 pixel-sized image [11].

3.2.3 CRNN. The convolutional recurrent neural network (CRNN) is a neural network architecture consisting of one or more convolutional layers followed by one or more recurrent layers. This particular network is inspired by works in the related field of human activity recognition (HAR) [2, 25], dealing with a related task of classifying multivariate sensor data. The main difference compared to our task of inferring keystrokes is the relatively long time span of human activity classes compared to single keystrokes. Our final architecture consists of a batch normalization layer, a single convolution layer with grouped convolutions, followed by a rectified linear unit (ReLU) function and batch normalization, and a series of two long short-term memory (LSTM) layers.

3.2.4 TSC WaveNet. This neural network is inspired by the WaveNet architecture [32] and the Gated PixelCNN architecture using dilated convolutions [33]. Although being designed for audio synthesis, the authors already suggested using the WaveNet architecture for classification tasks. To cope with the multivariate nature of our sensor data, we introduce grouped convolutions like explained earlier for the CRNN and TSC ResNet11 architectures. Aside from this change, our implementation closely follows the structure of the original work with two residual block layers.

3.3 Training Process

We perform a 3-fold cross-validated hyperparameter random search on the training data in order to assess the mean performance and stability of a hyperparameter set across all folds. The folds are created such that the data from a single participant is only present in a single fold in order to find hyperparameters which generalize well between different persons. The f1 score is used to rank the hyperparameter sets.

We use Adam ($\beta_1 = 0.9$, $\beta_2 = 0.999$) as the gradient descent optimizer for the TSC ResNet11, CWT+ResNet18 and TSC WaveNet models and RMSprop ($\alpha = 0.99$) for the CRNN as indicated in Table 1 together with the cross-entropy loss. During training, the last 20 % of the training data are used as validation data to apply early stopping.

Our data exhibits two types of class skew, one relevant for the binary and one for the multi-class classification as described in Section 2.4. When training a binary classifier, we apply random subsampling of the majority class

Table 1. Optimizer configuration and batch size per neural network architecture.

Model	Optimizer	Learning Rate	L2 Weight Decay	Batch Size
TSC ResNet11	Adam	0.001	0	16
CWT+ResNet18	Adam	0.0001	0.00001	512
CRNN	RMSprop	0.001	0.001	128
TSC WaveNet	Adam	0.005	0	128

(non-press events) after each training epoch to compensate for the comparatively small number of keypress events. For multi-class classification, we counter the imbalance of typing different keys a different amount of time by using class weights which are inversely proportional to their occurrences in the training data.

3.4 Post-processing

Even though our classifiers are trained on keypress events, their prediction of a keypress usually spans multiple timestamps. This is because each prediction made by our classifiers is independent of the prediction on neighboring samples. However, two keypresses rarely appear in two consecutive samples. We apply a simple peak detector on the output of the binary classifiers to refine the output by taking this temporal dependency between consecutive keystrokes into account.

We manually define a minimum peak distance of 25 ms, which is smaller than 99 % of all press-press intervals between keystrokes within the training data as shown on the left in Fig. 3. To further reduce the number of false positives, we require a minimum peak height of 0.5 with a minimum prominence of 0.05.

However, the prediction values resulting from this peak detection do not necessarily match the ground truth exactly but may be shifted by a couple of samples. We argue that a shift by a few samples should not have an effect on the overall results or the multi-class prediction, similar to the temporal tolerance applied in time series segmentation [10] and anomaly detection [15, 30] in which the presence of a prediction is more important than an exact match. Thus, we apply a custom metric that is tolerant given a *temporal tolerance*, i.e. a certain maximum shift in time, between the prediction of the classifier and the encoded truth. We formulate the problem of matching the ground truth keystrokes to the ones in the prediction as a linear assignment problem with the cost linearly increasing with the lag between the predicted and actual keystroke. Additionally, we assign an infinitely high cost to pairs with a distance larger than a given temporal tolerance, in order to avoid matching them. This allows us to create a temporally tolerant confusion matrix as described in Equations 1-4, where the `positives` are samples of keypress events and the `negatives` are samples without keypress events.

$$\texttt{true\_positives} = \texttt{matched\_positives} \tag{1}$$

$$\texttt{false\_positives} = \texttt{number\_of\_predicted\_positives} - \texttt{matched\_positives} \tag{2}$$

$$\texttt{false\_negatives} = \texttt{number\_of\_actual\_positives} - \texttt{matched\_positives} \tag{3}$$

$$\texttt{true\_negatives} = \texttt{number\_of\_predicted\_negatives} - \texttt{false\_negatives} \tag{4}$$

Given the values of this confusion matrix, we evaluate the results on standard metrics to gain a performance estimate of the peak detection.

3.5 Keystroke Detection Results

In the following, we present the performance results of evaluating our trained classifiers on the test data of our dataset described in Section 2.4. In this section, we evaluate the keystroke detection, the key identification results are given in Section 3.6.

Table 2. Mean and Standard Deviation of Keystroke Detection Results

Classifier Type	Balanced Acc. (%)	F1 Score (%)	Precision (%)	Recall (%)
TSC ResNet11	74.6 (9.1)	10.5 (2.3)	5.7 (1.3)	76.3 (18.1)
CWT+ResNet18	74.0 (9.8)	9.1 (2.4)	4.9 (1.6)	81.1 (20.2)
CRNN	76.1 (9.6)	10.1 (2.6)	5.4 (1.4)	82.1 (18.9)
TSC WaveNet	70.7 (9.5)	8.8 (2.8)	4.8 (1.6)	74.5 (23.2)

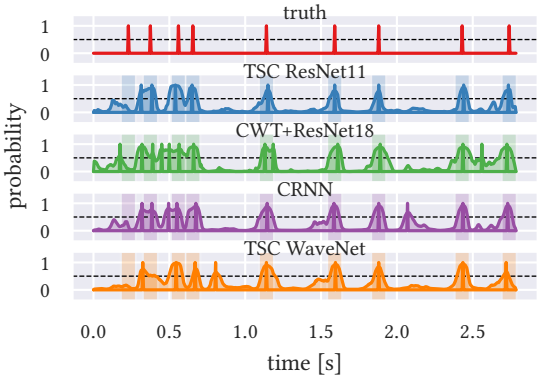

Fig. 6. Sample truth and prediction values for the password *wah/Ri2t* typed on a German keyboard layout with the `shift` key covering two consecutive keys (*/R*). Both the prediction probabilities and the peak predictions of each classifier are given. The semi-transparent background marks a temporal tolerance of 50 ms before and after the truth.

Table 2 shows the average prediction performance of each binary classifier on the entire test data, i.e. evaluated on all participants and all password types, without applying peak detection. The performance of the classifiers is similar with a mean balanced accuracy of about 74 – 76 % except for the TSC WaveNet which performs worse than the others with a mean balanced accuracy of only 70.7 %. CRNN and TSC ResNet11 perform best, the former showing better results on the mean balanced accuracy (76.1 %) and the latter showing better results on the mean f1 score (10.5 %).

The comparatively weak f1 score of about 9 – 10 % and precision of about 5 – 6 % can be explained by the high class skew and the observation that none of the trained classifiers produces a point-wise prediction to match the given truth. Instead, they show a wider peak of increasing and decreasing probabilities as shown in the example result in Fig. 6. This example of a keystroke detection on a pwgen password also allows some observations on the behavior of the classifiers. For example, a small distance between two keypresses often results in two merging peaks while keystrokes further apart from each other show a pronounced peak for the majority of the predictions. Peaks corresponding to false positives are often smaller than those corresponding to true positives. Similarly, false negatives are often still visible as a small peak.

Applying a peak detection to receive the point-wise predictions shown in Fig. 6 poses the problem that most of these predictions do not match the exact timestamps of the keypress event encoded in the truth as discussed in Section 3.4. This can be seen for a distance of zero on the left in Fig. 7 which equals the results of a classifier that almost always predicts the majority class. Thus, to give an estimate of the performance of the binary classifiers with a peak detection, we compare the results on different temporal tolerances as described in Section 3.4. The

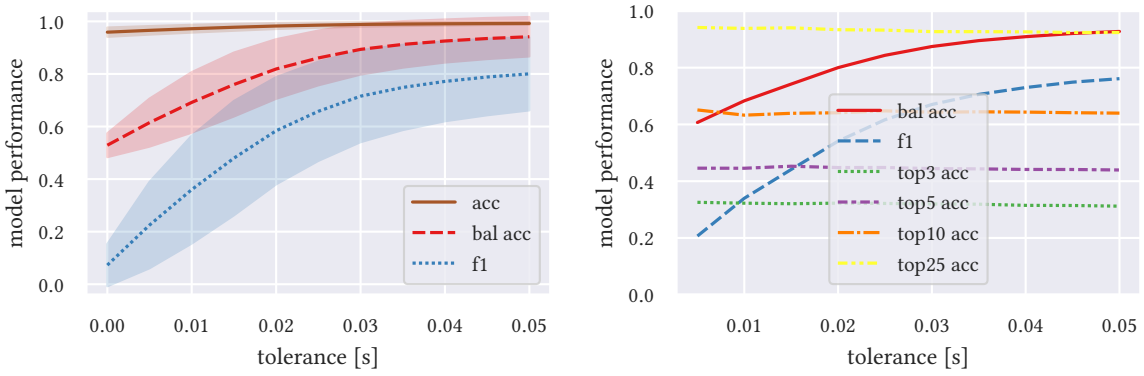

Fig. 7. The CRNN (left) and CWT+ResNet18 result (right) of the mean temporary f1 score and mean temporary balanced accuracy on the peak predictions of each classifier given different temporal tolerances along with the mean accuracy (left) and along with the mean top-n key accuracies (right). The standard deviation is omitted for better visibility on the right. The x axis of both plots depicts the temporal tolerance, i.e. the accepted distances around each keypress encoded in the truth within which a prediction is considered as a correct prediction. A distance of zero equals an evaluation of the peak detection based on the standard metrics without temporal tolerance.

Table 3. Mean and Standard Deviation of Keystroke Detection Results for Touch and Hybrid Typists (CRNN and TSC ResNet11)

Typist	Balanced Accuracy (%)		F1 Score (%)	
	TSC ResNet11	*CRNN*	*TSC ResNet11*	*CRNN*
touch	71.3 (8.6)	73.1 (9.6)	10.3 (2.7)	10.3 (2.8)
hybrid	77.5 (8.6)	78.7 (8.8)	10.6 (2.0)	9.9 (2.4)

left part of Fig. 7 shows the results of this evaluation with both the performance and variance increasing for a temporal tolerance of zero (i.e. the prediction must match the truth exactly) to 50 ms the latter of which is depicted as a semi-transparent window in Fig. 6. For a window of 50 ms, the CRNN achieves a mean balanced accuracy of 94.2 % and a mean f1 score of 80.0 %. We observe that the lags between the true and the predicted keypress events are normally distributed. For a distance of 50 ms to the encoded true keypress the mean lag is −4.1 ms for the CRNN and −2.6 ms for the TSC ResNet11 with a standard deviation of 21.55 ms and 20.65 ms respectively. Thus, most keypresses are predicted slightly before they occur according to the ground truth.

3.5.1 Typing Style. On average, the results of hybrid typists compared to touch typists are better regarding the balanced accuracy but worse or almost equal regarding the f1 score as shown in Table 3. This is likely due to the typing speed difference between the two groups, given that in our dataset touch typists type faster on average than hybrid typists, as described in Section 2.4, and the typing speed correlates negatively with the performance of the binary models, i.e. the mean balanced accuracy drops for higher typing speeds, as shown on the left in Fig. 8. This could be explained by a lower number of positive predictions (both correct and false) due to an increased chance of the classifier to completely miss a keystroke, which is more likely to happen for faster

Table 4. Mean and Standard Deviation of Keystroke Detection Results per Password Type (CRNN and TSC ResNet11)

Password Type	Balanced Accuracy (%)		F1 Score (%)	
	TSC ResNet11	*CRNN*	*TSC ResNet11*	*CRNN*
random	78.5 (9.2)	81.2 (9.2)	10.1 (2.7)	9.2 (2.6)
pwgen	76.8 (7.9)	76.9 (9.3)	10.0 (2.0)	9.0 (2.2)
xkcd	73.0 (6.5)	74.2 (6.2)	10.8 (1.3)	10.7 (1.6)
insecure	69.9 (10.0)	72.0 (10.7)	11.1 (2.8)	11.3 (3.1)

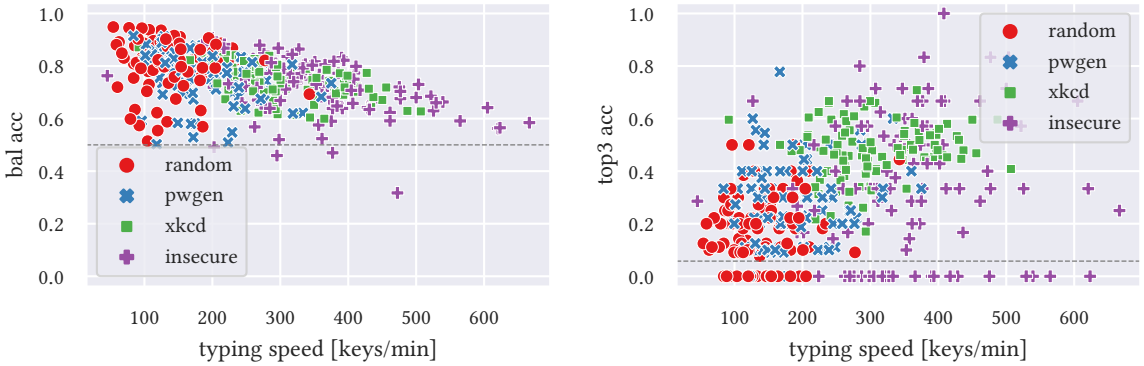

Fig. 8. The mean balanced accuracy for the keystroke detection evaluated on the CRNN model (left) and mean top-3 key accuracy for the key identification evaluated on the CWT+ResNet18 model (right) for each password recording compared to the current typing speed of the respective participant. The horizontal line marks the probability of guessing.

typists. Also, multiple keys in rapid succession leave less samples between keys for false positives. On the other hand, predictions of multiple quickly typed keystrokes will likely merge, such as the first four keys in Fig. 6.

3.5.2 Password Types. A similar trend for performance applies for the different password types shown in Table 4 and on the left in Fig. 8, as (pseudo)random passwords are typed in a slower speed than those resembling text as mentioned in Section 2.4. The slight decrease in the mean balanced accuracy for pwgen passwords shows that these are slightly easier to type than random passwords but harder than passwords resembling text. Interestingly, the insecure passwords show the widest performance spread, likely because most of them are short and rapidly typed.

3.5.3 EMG vs IMU. To compare the performance of each model on the EMG, accelerometer and gyroscope sensor, we train one model each using only the data of the respective sensor. When training the models only on the EMG data, the performance drops slightly (3.2 % in the mean balanced accuracy for the CRNN) compared to training on all data. When training the models only on the gyroscope data, this performance drop increases to 7.7 % in the mean balanced accuracy for the CRNN and the results are less consistent among the four classifiers, with the TSC ResNet11 and the CRNN performing better than the other two, as shown on the left in Fig. 9. This effect is even more prominent for the accelerometer data for which the CRNN performs best with a performance drop of 8.5 % in the mean balanced accuracy compared to using all data, with the other classifiers showing drops as high

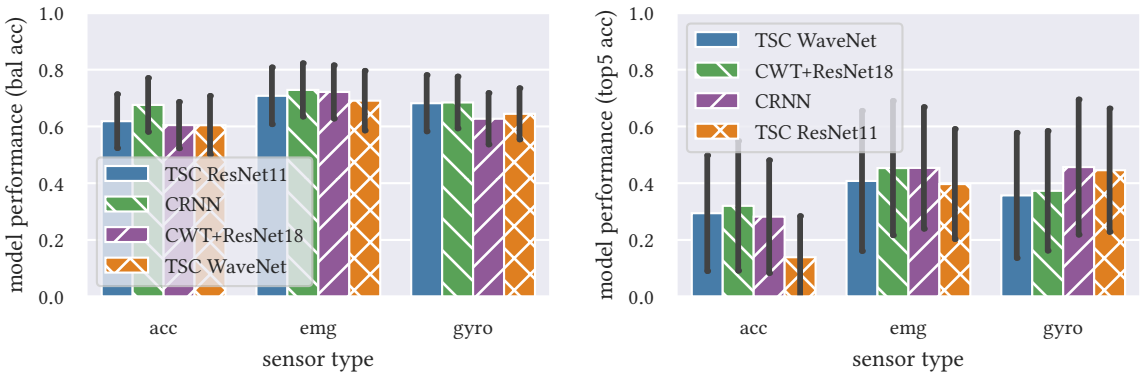

Fig. 9. The mean balanced accuracy of the keystroke detection (left) and the mean top-5 key accuracy of the key identification (right) and the respective standard deviation for different sensor types and classifiers.

as 13.5 % and only reaching performances of about 60 % mean balanced accuracy. This comparison suggests that the accelerometer data has little influence on the results and that the results mostly depend on the EMG data.

3.5.4 Known vs Unknown Participants/Passwords. Comparing the average performances on *known* participants whose data is present in the training data with the average performances on *unknown* participants yields no significant difference compared with the standard deviation of both results. This may be due to the effect described in Section 2.4, concerning recordings of the same participant in a new recording session, but also shows the capability of the models to generalize between participants. The same observation applies when comparing the results on shared data, i.e. passwords typed by multiple participants, with the results on individual data, i.e. passwords typed by only one participant each. However, the generalization does not work equally well for all participants as the mean balanced accuracy for individual participants ranges from 63.3 % to 83.2 %, as shown on the left in Fig. 10. Nevertheless, the results on each of the participants is significantly better than guessing, with the performances for many of them showing similar results. These differences are similar across all considered models.

3.6 Key Identification Results

Table 5 shows the average performance of each classifier on the entire test data, i.e. evaluated on all participants and all password types. Note that, if not stated otherwise, the key identification is evaluated separately from the keystroke detection on samples containing a valid keystroke. We thus give an estimation of the performance independent of our keystroke detection and possible improvements thereof, e.g. by incorporating further modalities or choosing different temporal tolerances for the post-processing step.

The CWT+ResNet18 shows the best performance of about 32.1 % for the mean top-3 key accuracy, followed by the TSC WaveNet with a mean top-3 key accuracy of 27.8 %. The TSC ResNet11 and CRNN model only reach a mean top-3 key accuracy of about 24 %. All four models achieve a mean performance value significantly higher than random guessing ($3/52 \approx 6$ % for the mean top-3 key accuracy, considering 52 classes) and for all models, the majority of measured performance values is higher than random guessing. However, all of the performance values exhibit comparatively large variations, especially when compared to the keystroke detection.

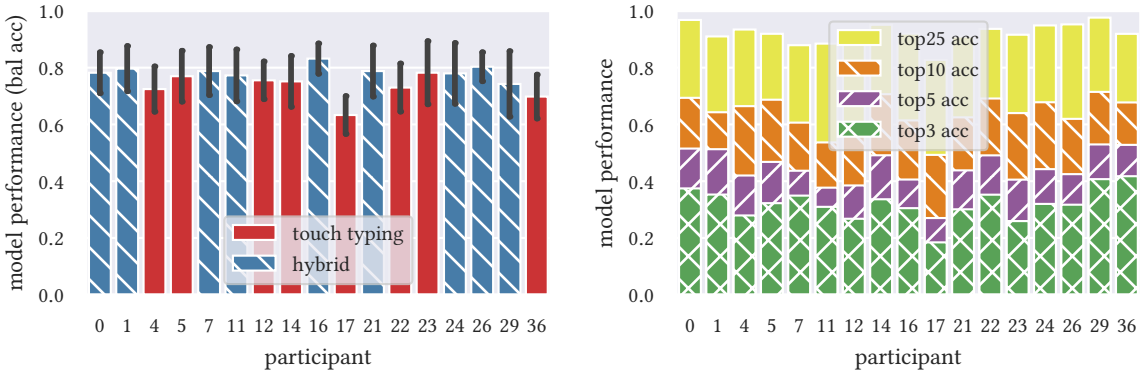

Fig. 10. The mean balanced accuracy of the keystroke detection of all participants of the CRNN model (left) and the mean top-n key accuracies of the key identification of all participants of the CWT+ResNet18 model (right). The standard deviation is omitted for better visibility on the right.

Table 5. Mean and Standard Deviation of Top-n Key Accuracy Key Identification Results

Classifier Type	Top-3 Acc. (%)	Top-5 Acc. (%)	Top-10 Acc. (%)	Top-25 Acc. (%)
TSC ResNet11	24.0 (15.8)	33.9 (18.7)	53.4 (19.9)	84.8 (14.0)
CWT+ResNet18	32.1 (20.3)	44.4 (23.4)	64.1 (25.3)	92.2 (11.8)
CRNN	24.3 (17.6)	33.8 (21.4)	52.5 (23.5)	81.7 (17.1)
TSC WaveNet	27.8 (20.4)	39.4 (23.7)	58.2 (26.0)	87.4 (14.2)

The right part of Fig. 7 shows the impact of our binary classification with a peak detection for different temporal tolerances on the results of our multi-class classifier, compared to the improved keystroke detection rates when increasing the temporal tolerance. We observe that the mean performance of our second stage is very similar for temporal tolerances of $0 - 50$ ms, which indicates that our assumption made in Section 3.4 is correct and that the multi-class performance results are invariant to a shift of a few samples between a true keypress event and its prediction. Note that we only consider the true positives for this test of our second stage, as using the false positives would not allow a meaningful evaluation of the robustness against classifying keystrokes with slightly miss-aligned samples.

The right part of Fig. 10 shows the performance of the key identification on each participant, assuming a perfect keystroke detection. For all of the participants, the mean top-n key accuracy is significantly above random guessing but the results vary more strongly between different participants than for the binary model on the left of Fig. 10. Notably, both the binary and multi-class models perform worst on participant 17 for both the mean balanced accuracy as well as across all mean top-n key accuracies, but the best performance is not consistent between the binary and multi-class classifiers. Nevertheless, the results show that a generalization between users is possible, though the results will vary for some users.

3.6.1 Password Types. The right part of Fig. 8 shows the performance of the multi-class CWT+ResNet18 on different password types, an average of which is given in Table 6. While the insecure passwords show a very high variation and a high number of samples for which the classifier has a mean top-3 key accuracy of 0 %, the

Table 6. Mean and Standard Deviation of Key Identification Results per Password Type (CWT+ResNet18)

Password Type	Top-3 Acc.	Top-5 Acc.	Top-10 Acc.	Top-25 Acc.
random	17.7 (12.8)	27.9 (14.5)	47.3 (17.9)	84.7 (13.6)
pwgen	28.7 (15.0)	41.3 (16.6)	62.3 (14.0)	90.9 (9.2)
xkcd	47.5 (10.4)	64.0 (10.5)	84.4 (8.2)	98.6 (2.0)
insecure	34.6 (26.4)	44.3 (30.6)	62.4 (36.0)	94.7 (13.6)

results on xkcd passwords are comparatively good and show little variation, even though these are typed faster than the random and pwgen passwords which show a worse performance on average. This is contrary to the results for the binary classifier, which works best on slowly typed data, as shown on the left in Fig. 8. This may indicate that a consistency in the typing pattern, which is more likely for longer text than for random character combinations (and also for the entry of a well-known password), has more influence on the key identification results than the typing speed.

3.6.2 EMG vs IMU. To compare the performance of each model on the EMG, accelerometer and gyroscope sensor, we train one model each using only the data of the respective sensor. The models trained only on the EMG data perform slightly better overall, being more consistent for different models than the ones trained only on gyroscope data, as shown on the right in Fig. 9. However, the best results based on the gyroscope by the TSC ResNet11 and the CRNN are similar to the best result on the EMG data by the CWT+ResNet18 and the CRNN. Both results are significantly better than the results for the accelerometer, for which the TSC ResNet11 comes close to guessing. All results show a comparatively high variation, in particular compared to the binary results on the left of Fig. 9. These results show that the EMG and gyroscope sensor data are about equally important for the key identification. However, depending on the classifier used, the accelerometer negatively influences the performance when combining all sensors, with the EMG and gyroscope data individually yielding a better result for the TSC ResNet11 and the CRNN.

3.7 Result Overview

In summary, we apply four different end-to-end machine learning models for (1) keystroke detection and for (2) key identification. The CRNN and the TSC ResNet11 perform best for keystroke detection with a mean balanced accuracy of 76.1 % and 74.6 % respectively. Applying a peak-detector in combination with a temporal tolerance allows us to boost this performance even further depending on the chosen tolerance. For example, a conservative choice of 25 ms, which is smaller than 99 % of all keystroke intervals observed in our training data, boosts the mean balanced accuracy to 86.1 % and the f1 score to 65.8 %. Depending on the speed of the typist, one could safely choose even larger values, which would synergize with our insights from Fig. 8, showing a higher vulnerability of slower typist. As shown on the right in Fig. 7, the choice of the temporal tolerance does not influence the key identification.

For key identification the CWT+ResNet18 performs significantly better than the other networks. Assuming a perfect keystroke detection, it achieves a mean top-3 key accuracy of about 32.1 % when classifying 52 different keys as part of our key identification. In contrast to the keystroke detection, the key identification performance is mostly independent of the typing speed but may be influenced strongly by the typing pattern and its consistency. Passwords such as the xkcd password based on multiple, randomly selected words to achieve a high entropy seem to be among the most vulnerable for our approach.

In comparison, our TSC WaveNet model is consistently outperformed in both keystroke detection and key identification, though for the latter it performs better than the CRNN and TSC ResNet11 on all data. In general, all

four models proved to be able to learn patterns from the raw data and yield results better than guessing. Taking a look at real-world password entry, a well-known password will usually by typed fast, which is disadvantageous for our keystroke detection, but it will presumably also always be typed in a similar way due to muscle memory, for which in turn our key identification performs well, as shown for the xkcd passwords on the right in Fig. 8.

For the keystroke detection, we observe that the EMG data is the most important sensor, closely followed by the gyroscope, both of which are better than the accelerometer. For the key identification the EMG data is similarly important to the gyroscope data and more so regarding cross-model consistency of results, whereas the accelerometer data can negatively influence the performance depending on the choice of classifier. This supports our intuition that the EMG data provides a more fine-grained insight into the movements of the finger compared to the IMU data of the Myo armband and suggests that the EMG and gyroscope sensors should be preferred when investigating the impact of a keylogging side-channel attack on a wearable worn on the lower arm.

Furthermore, we observe that for both the keystroke detection and the key identification, a generalization between users is possible.

4 LIMITATIONS AND DISCUSSION

Though we tried to replicate a realistic scenario, there are some limitations to our data and approach.

First, we have used only one keyboard for all recordings, thus we cannot provide sufficient data about the generalization capabilities of our approach concerning different models of keyboards.

Second, the participants wore the Myo armband always in a specific position and orientation on the lower arm. Since this position is predefined by the vendor, users of the Myo armband will most likely use the same positioning. But creating a more extensive dataset with multiple Myo armband positions may result in a more powerful model, in particular when regarding possible generalizations across devices such as prostheses. Including a wrist-worn wearable to directly compare our findings regarding the performance influence of the different sensors would also be interesting in the future.

Third, as mentioned in Section 2, wearables constantly generate data which is not directly linked to the action of typing on a keyboard. Thus, an additional typing activity detection should be explored to reduce the number of false predictions of keystrokes during different daily activities. As this also provides a possible mitigation for the presented attack by cutting out the detected sequences of typing from the sensor data, this would be a worthwhile addition for future work.

Forth, while we include the `shift` modifiers in our dataset, our classification approach is unable to detect if a pressed `shift` modifier applies to one or multiple keys, as we do not consider key releases. An example of this is shown in Fig. 6 where `shift` is pressed to type both / and *R* on a German keyboard layout. A multi-label approach with a key state encoding or a combination of classifiers predicting press and release events could solve this.

Fifth, our classifiers lack knowledge about predictions on neighboring samples and thus are unable to learn that keypresses usually do not appear in consecutive samples. Including the prior truth into the models or choosing a sequence-to-sequence approach may improve the results and make the peak detection obsolete.

Apart from the restrictions described above, we have tried to simulate natural typing behavior by including `backspace` and the `shift` modifier, including different postures occurring in everyday keyboard usage, as well as allowing non-typing movements, small breaks and individual typing styles. We have included an extensive set of keys of a physical keyboard, focusing on the physical keys rather than their character representation in order to be independent of a logical keyboard layout. Furthermore, we present a dataset of diverse training and test data meant to mimic realistic password entry in order to evaluate keylogging side-channel attacks based on EMG and IMU data recorded from the lower arm.

Our results show that even under these difficult conditions, significant information can be gained regarding the inference of keystrokes from sensor data. According to our experiments, the EMG data in particular can raise the performance of such inferences and could be used without requiring additional sensor data or as a surrogate for the accelerometer or for microphones in approaches that combine sensor data for key identification with audio data for keystroke detection. Compared to previous attacks focusing on IMU devices, typically worn on the wrist, EMG devices offer an additional attack vector, but also a potentially better way to infer keystrokes as indicated by our experiments.

Furthermore, it may be interesting to further explore the potential of using neural networks based on end-to-end learning for between-subject approaches. Creating more open access datasets would be a next step towards enhancing and comparing such approaches. If each new sensor modality opens up new side channels, this possibility should be explored in realistic and replicable settings to be able to explore feasible mitigations.

5 RELATED WORK

Multiple papers have investigated wrist-worn smartwatches or fitness trackers to infer (pass)words typed on a physical keyboard [16, 18, 26, 36] or personal identification numbers (PINs) typed on a number pad like a payment terminal used in stores [16, 17, 35]. All this work relies on motion data captured solely by an accelerometer and a gyroscope [17, 26, 36], an accelerometer and a magnetometer [35], or an accelerometer while using the microphone to detect keystrokes [16, 18].

Similar research has been done to infer passwords typed on the touchscreen-based virtual keyboards of smartphones [19, 20, 28], but the motion when typing on a smartphone is very different from that on a physical keyboard and often dominated by the thumb. The latter is used mostly for typing the space key on a physical keyboard and is often disregarded by some of the work related closely with ours because the authors only consider words or passwords (e.g. [36, 39]).

Of those targeting a physical keyboard, most include a language analysis and focus on recovering words or sentences from the English language [16, 18, 36]. Only Pandelea and Chiroiu [26] mention passwords, including alphabetic characters and digits. But they do not reach a competitive accuracy for the full keyboard, only on the number pad. Most enforce or assume the user to use touch typing [16, 26, 36], only Maiti et al. [18] do not.

Pandelea and Chiroiu [26] are the first to apply a convolutional neural network to the problem of inferring keystrokes. But the authors use it on features and have not tried end-to-end learning with neural networks.

Yang et al. [38] implement a partial text-entry system using five channels of EMG data acquired from the left forearm with a custom system for predicting 16 different keys with feature-based classifiers. The authors use a custom algorithm to determine the onset and offset of keystrokes by comparing the output value to a user-dependent threshold value. Similar to the basic idea of our peak detector, the authors revise their prediction by taking the typical timing of a keystroke into account. For doing so, they discard recognized keystrokes with a duration smaller than 300 ms. However, this is a larger value than encountered for the majority of keystrokes in our training data as shown on the right in Fig. 3.

Zhang et al. [39] are closest to our work as they also use the Myo armband and are the only ones apart from us using EMG data for a keylogging side-channel attack on a physical keyboard. The authors describe both an attack to infer PINs from typing on a number pad as well as inferring passwords from typing on a physical keyboard. They use a custom algorithm for detecting keystrokes using the EMG data collected from the Myo armband. For key identification, the authors use a feature-based finger classification based on the EMG data in combination with a hand position tracking derived from the accelerometer data to identify keystrokes in passwords consisting of four to eight characters recorded as test data.

The authors assume that the target of the described attack uses the Ratatype typing style, a variant of touch typing, and that the data is recorded in a strict environment, i.e. the participants in their experiments were told

to avoid movements and to keep their wrists above the table while typing [39]. Both are strong assumptions which limit the generalizability of the approach, as many typists frequently change their posture and move their hands or arms away from the keyboard or table. One could argue that this is more likely to happen given a natural break, i.e. at the end of a sentence, and less likely to happen while the users type a word, in particular a password. But taking a different posture may have an impact on the classification itself, as does the typing style, and given that the authors assume perfect Ratatype typing, their classifier will not be able to classify other typing styles correctly. However, many other typing styles exist and even most touch typists do not display a perfect finger-to-key mapping [8].

In our work, we have tried to minimize assumptions and assume a realistic scenario in order to evaluate the feasibility of an actual attack on a user. The results show that our approach is barely influenced by a person's typing style with our classifiers performing slightly better on hybrid typists. We have gathered an extensive dataset and tested our classifiers on different types of passwords of varying lengths and strengths. Furthermore, we have compared our approach on both the EMG and IMU data and found that it works best for the EMG and worst for the accelerometer data, which also makes it a valid option to only rely on the EMG data.

Furthermore, Zhang et al. [39] use a within-subject approach, which requires recording of and training a classifier on a victim's data to launch the attack. This approach requires either to extract both the sensor data and the ground truth (supervised learning) or to extract only the sensor data and to apply an unsupervised learning model. In case of the key identification, this could be achieved by using a language model to leverage known structures for predicting the most likely keys. However, it does not allow generalizing between users, i.e. training a classifier on a set of users before targeting victims outside this set. In fact, the authors claim that they have tried a generalizing approach but have not succeeded due to the differences in different user's muscle structures.

We have analyzed the data in our dataset and found that even under the assumption that participants wear the Myo armband in the same way on different days, the recordings of different sessions differ significantly, being in part more similar to those of other participants, as shown in Fig. 4. This may render a within-user approach useless if the classifier has to be retrained each time the Myo armband is taken off. We have further shown that a between-subject approach works well on most users, making it a powerful and feasible option for further exploration of using EMG data in keylogging side-channel attacks.

In general, it is hard to compare different approaches, as, for example, smartwatches are usually worn on the wrist while the Myo armband is worn on the lower arm. Furthermore, metrics such as accuracy and recall convey little meaning by themselves in a keystroke detection scenario which is heavily influenced by class skew, as shown in Section 2.4 and as illustrated for the accuracy on the left in Fig. 7. Similarly, metrics such as key accuracy, word accuracy or metrics that rely on the look-up of a word in a given dictionary cannot be directly translated to each other. In order to improve comparability, multiple metrics need to be reported and open access datasets have to be created, against which different approaches can be measured. With this work, we have attempted this while investigating a new sensor modality and bridging the gap to existing keylogging side-channel attacks by including and comparing the IMU data of lower arm movement in our dataset and evaluation.

6 CONCLUSION

In this work, we have explored the potential of using EMG and IMU data for a keylogging side-channel attack. We have implemented the first between-subject feature-less approach based on neural networks and EMG data and evaluated it on a dataset of sensor and key data recorded in a realistic scenario using the Myo armband. We have shown the feasibility of such an approach and the advantage that can be gained by using the EMG data to improve the performance of keylogging side-channel attacks, in particular compared with the accelerometer data. Furthermore, we have created an extensive dataset which is available as open access along with the source code of this research.

ACKNOWLEDGMENTS

We thank our shepherd and our anonymous reviewers for their insightful comments that helped to improve this paper. Calculations for this research were conducted partially on the Lichtenberg high performance computer of the TU Darmstadt. This work has been co-funded by the German Research Foundation as part of project C.1 within the RTG 2050 "Privacy and Trust for Mobile Users" as well as by the German Federal Ministry of Education and Research and the Hessian Ministry of Higher Education, Research, Science and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE.

REFERENCES

[1] Joseph Bonneau. 2016. *Deep Dive: EFF's New Wordlists for Random Passphrases.* Retrieved 2021-10-21 from https://www.eff.org/deeplinks/2016/07/new-wordlists-random-passphrases

[2] David M. Burns, Nathan Leung, Michael Hardisty, Cari M. Whyne, Patrick Henry, and Stewart McLachlin. 2018. Shoulder physiotherapy exercise recognition: machine learning the inertial signals from a smartwatch. *Physiological Measurement* 39, 7 (2018). https://doi.org/10.1088/1361-6579/aacfd9

[3] Ulysse Côté-Allard, Cheikh Latyr Fall, Alexandre Drouin, Alexandre Campeau-Lecours, Clément Gosselin, Kyrre Glette, François Laviolette, and Benoit Gosselin. 2019. Deep Learning for Electromyographic Hand Gesture Signal Classification Using Transfer Learning. *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 27, 4 (2019), 760–771. https://doi.org/10.1109/TNSRE.2019.2896269

[4] Carlo J. De Luca, L. Donald Gilmore, Mikhail Kuznetsov, and Serge H. Roy. 2010. Filtering the surface EMG signal: Movement artifact and baseline noise contamination. *Journal of Biomechanics* 43, 8 (2010), 1573–1579. https://doi.org/10.1016/j.jbiomech.2010.01.027

[5] Jack T. Dennerlein, C. D. Mote Jr., and David M. Rempel. 1998. Control strategies for finger movement during touch-typing The role of the extrinsic muscles during a keystroke. *Experimental Brain Research* 121, 1 (1998), 1–6. https://doi.org/10.1007/s002210050430

[6] Jessilyn Dunn, Ryan Runge, and Michael Snyder. 2018. Wearables and the medical revolution. *Personalized Medicine* 15, 5 (2018), 429–448. https://doi.org/10.2217/pme-2018-0044

[7] Facebook 2021. *Inside Facebook Reality Labs: Wrist-based interaction for the next computing platform.* Retrieved 2021-10-21 from https://tech.fb.com/inside-facebook-reality-labs-wrist-based-interaction-for-the-next-computing-platform/

[8] Anna Maria Feit, Daryl Weir, and Antti Oulasvirta. 2016. How We Type: Movement Strategies and Performance in Everyday Typing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. Association for Computing Machinery, 4262–4273. https://doi.org/10.1145/2858036.2858233

[9] Zongkai Fu, Huiyong Li, Zhenchao Ouyang, Xuefeng Liu, and Jianwei Niu. 2020. Typing Everywhere with an EMG Keyboard: A Novel Myo Armband-Based HCI Tool. In *Algorithms and Architectures for Parallel Processing (Lecture Notes in Computer Science)*. Springer International Publishing, 247–261. https://doi.org/10.1007/978-3-030-60245-1_17

[10] Andre Gensler and Bernhard Sick. 2014. Novel Criteria to Measure Performance of Time Series Segmentation Techniques. In *Proceedings of the 16th LWA Workshops: KDML, IR and FGWM (CEUR Workshop Proceedings, Vol. 1226)*. CEUR-WS.org, 193–204.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 770–778. https://doi.org/10.1109/CVPR.2016.90

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity Mappings in Deep Residual Networks. In *Computer Vision – ECCV 2016 (Lecture Notes in Computer Science)*. Springer International Publishing, 630–645. https://doi.org/10.1007/978-3-319-46493-0_38

[13] Yani Ioannou, Duncan Robertson, Roberto Cipolla, and Antonio Criminisi. 2017. Deep Roots: Improving CNN Efficiency with Hierarchical Filter Groups. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 5977–5986. https://doi.org/10.1109/CVPR.2017.633

[14] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. 2019. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery* 33, 4 (2019), 917–963. https://doi.org/10.1007/s10618-019-00619-1

[15] Alexander Lavin and Subutai Ahmad. 2015. Evaluating Real-time Anomaly Detection Algorithms – the Numenta Anomaly Benchmark. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*. 38–44. https://doi.org/10.1109/ICMLA.2015.141

[16] Xiangyu Liu, Zhe Zhou, Wenrui Diao, Zhou Li, and Kehuan Zhang. 2015. When Good Becomes Evil: Keystroke Inference with Smartwatch. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS '15)*. Association for Computing Machinery, 1273–1285. https://doi.org/10.1145/2810103.2813668

[17] Yang Liu and Zhenjiang Li. 2018. aLeak: Privacy Leakage through Context-Free Wearable Side-Channel. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*. 1232–1240. https://doi.org/10.1109/INFOCOM.2018.8485958

[18] Anindya Maiti, Oscar Armbruster, Murtuza Jadliwala, and Jibo He. 2016. Smartwatch-Based Keystroke Inference Attacks and Context-Aware Protection Mechanisms. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security (ASIA CCS '16)*. Association for Computing Machinery, 795–806. https://doi.org/10.1145/2897845.2897905

[19] Anindya Maiti, Murtuza Jadliwala, Jibo He, and Igor Bilogrevic. 2015. (Smart)Watch Your Taps: Side-Channel Keystroke Inference Attacks using Smartwatches. In *Proceedings of the 2015 ACM International Symposium on Wearable Computers (ISWC '15)*. Association for Computing Machinery, 27–30. https://doi.org/10.1145/2802083.2808397

[20] Anindya Maiti, Murtuza Jadliwala, Jibo He, and Igor Bilogrevic. 2018. Side-Channel Inference Attacks on Mobile Keypads Using Smartwatches. *IEEE Transactions on Mobile Computing* 17, 9 (2018), 2180–2194. https://doi.org/10.1109/TMC.2018.2794984

[21] Sumit Majumder and M. Jamal Deen. 2019. Smartphone Sensors for Health Monitoring and Diagnosis. *Sensors* 19, 9 (2019). https://doi.org/10.3390/s19092164

[22] Daniel Miessler, Jason Haddix, and g0tmi1k. 2018. *SecLists*. Retrieved 2021-10-21 from https://github.com/danielmiessler/SecLists/blob/master/Passwords/Common-Credentials/10-million-password-list-top-1000.txt

[23] John V. Monaco. 2018. SoK: Keylogging Side Channels. In *2018 IEEE Symposium on Security and Privacy (SP)*. 211–228. https://doi.org/10.1109/SP.2018.00026

[24] Randall Munroe. 2011. *Password Strength*. Retrieved 2021-10-21 from https://xkcd.com/936/

[25] Francisco Javier Ordóñez and Daniel Roggen. 2016. Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition. *Sensors* 16, 1 (2016). https://doi.org/10.3390/s16010115

[26] Alexandru-Ionut Pandelea and Mihai-Daniel Chiroiu. 2019. Password guessing using machine learning on wearables. In *2019 22nd International Conference on Control Systems and Computer Science (CSCS)*. 304–311. https://doi.org/10.1109/CSCS.2019.00055

[27] Timothy A. Salthouse. 1984. Effects of Age and Skill in Typing. *Journal of Experimental Psychology: General* 113, 3 (1984), 345–371. https://doi.org/10.1037/0096-3445.113.3.345

[28] Allen Sarkisyan, Ryan Debbiny, and Ani Nahapetian. 2015. WristSnoop: Smartphone PINs Prediction Using Smartwatch Motion Sensors. In *2015 IEEE International Workshop on Information Forensics and Security (WIFS)*. https://doi.org/10.1109/WIFS.2015.7368569

[29] Dawn Xiaodong Song, David Wagner, and Xuqing Tian. 2001. Timing Analysis of Keystrokes and Timing Attacks on SSH. In *Proceedings of the 10th conference on USENIX Security Symposium*.

[30] Nesime Tatbul, Tae Jun Lee, Stan Zdonik, Mejbah Alam, and Justin Gottschlich. 2018. Precision and Recall for Time Series. In *Advances in Neural Information Processing Systems*, Vol. 31. Curran Associates, Inc., 1920–1930. arXiv:1803.03639 [cs.LG]

[31] Christopher Torrence and Gilbert P. Compo. 1998. A Practical Guide to Wavelet Analysis. *Bulletin of the American Meteorological Society* 79, 1 (1998), 61–78. https://doi.org/10.1175/1520-0477(1998)079<0061:APGTWA>2.0.CO;2

[32] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. 2016. WaveNet: A Generative Model for Raw Audio. *CoRR* (2016). arXiv:1609.03499 [cs.SD]

[33] Aäron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. 2016. Conditional Image Generation with PixelCNN Decoders. In *Advances in Neural Information Processing Systems*, Vol. 29. Curran Associates Inc., 4790–4798. arXiv:1606.05328 [cs.CV]

[34] Paolo Visconti, Federico Gaetani, Giovanni Antonio Zappatore, and Patrizio Primiceri. 2018. Technical Features and Functionalities of Myo Armband: An Overview on Related Literature and Advanced Applications of Myoelectric Armbands Mainly Focused on Arm Prostheses. *International Journal on Smart Sensing and Intelligent Systems* 11, 1 (2018), 1–25. https://doi.org/10.21307/ijssis-2018-005

[35] Chen Wang, Xiaonan Guo, Yan Wang, Yingying Chen, and Bo Liu. 2016. Friend or Foe? Your Wearable Devices Reveal Your Personal PIN. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security (ASIA CCS '16)*. Association for Computing Machinery, 189–200. https://doi.org/10.1145/2897845.2897847

[36] He Wang, Ted Tsung-Te Lai, and Romit Roy Choudhury. 2015. MoLe: Motion Leaks through Smartwatch Sensors. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking (MobiCom '15)*. Association for Computing Machinery, 155–166. https://doi.org/10.1145/2789168.2790121

[37] Zhiguang Wang, Weizhong Yan, and Tim Oates. 2017. Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline. In *2017 International Joint Conference on Neural Networks (IJCNN)*. 1578–1585. https://doi.org/10.1109/IJCNN.2017.7966039

[38] Qiang Yang, Yongpan Zou, Meng Zhao, Jiawei Lin, and Kaishun Wu. 2018. ArmIn: Explore the Feasibility of Designing a Text-entry Application Using EMG Signals. In *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous '18)*. Association for Computing Machinery, 117–126. https://doi.org/10.1145/3286978.3287030

[39] Ruide Zhang, Ning Zhang, Changlai Du, Wenjing Lou, Y. Thomas Hou, and Yuichi Kawamoto. 2017. From Electromyogram to Password: Exploring the Privacy Impact of Wearables in Augmented Reality. *ACM Transactions on Intelligent Systems and Technology* 9, 1 (2017). https://doi.org/10.1145/3078844

A DATASET AND SOURCE CODE

The dataset created as part of this work is available as open access at https://doi.org/10.5281/zenodo.5594651. The source code to record the data, train the models and reproduce our final results is available as free software at https://github.com/seemoo-lab/myo-keylogging.

B CLASSIFIER CONFIGURATION DETAILS

The following describes the neural networks in more detail.

B.1 TSC ResNet11

Compared to the original architecture, due to using grouped convolutions, we use $\{8, 16, 16\}$ filters per input channel, totaling in $\{224, 448, 448\}$ filters based on our hyperparameter optimization, compared to $\{64, 128, 128\}$ in the original work.

B.2 CWT ResNet18

Our main changes to the original ResNet18 architecture are the removal of the max pooling layer and the reduction of the initial number of convolution filters from 64 to 32.

B.3 CRNN

We use grouped convolutions to apply 16 individual filters per sensor channels (a total of 448 filters) with a kernel size of two. Both LSTM layers consist of 64 units and we apply a dropout of 0.4 for regularization.

B.4 TSC WaveNet

The core architecture can be divided into three parts: The first part consist of a batch normalization layer followed by a convolution with kernel size one for adapting the number of input channels to the second part. The second part of the model consists of a series of residual blocks as described in the work of van den Oord et al. [32]. In contrast to the models based on the ResNet architecture, the output of each residual block, which contain the dilated convolutions, is not only fed into the next block, but added together before being fed into the third part of the network. The third part consists of a simple series of a ReLU, a convolution with kernel size one and another ReLU, followed by a convolution with kernel size one, which is connected to the output layer.