# Gamification of Wearable Data Collection: A Tool for both Friend and Foe

Sraddhanjali Acharya
Texas Tech University
Lubbock, Texas, US
sraddhanjali.acharya@ttu.edu

Richard Matovu
Texas Tech University
Lubbock, Texas US
richard.matovu@ttu.edu

Abdul Serwadda
Texas Tech University
Lubbock, Texas, US
abdul.serwadda@ttu.edu

Isaac Griswold-Steiner
Texas Tech University
Lubbock, Texas, US
isaac.griswold-steiner@ttu.edu

## ABSTRACT
Games are designed to build on certain inherently reward centered aspects of our psychology. This encourages the user to keep playing and engaging in the entertainment for just a bit longer. In this paper we explore the way in which gamification can be used to benefit researchers aiming to collect large amounts of data from sometimes less than enthusiastically motivated participants.

We also investigate the risks associated with such mechanisms for data collection and how malicious entities could use these same methods to trick users into exposing private information. Across several experiments used to measure the cross-applicable nature of the data we collected, we demonstrate that a gamified version of a data collection tool could be used to predict the pattern used to unlock a phone.

## CCS Concepts
**• Security and privacy→Mobile and wireless security, Malware and its mitigation, Biometrics.**

## Keywords
Gamification; Wearables;

## 1. INTRODUCTION
Data collection is often one of the most difficult yet unavoidable parts of developing machine learning-driven wearable applications (see common example applications in Table 1). Not only is it time consuming and dependent on rigorous data collection methodologies, but it also requires that a large number of participants execute a set of actions for significant periods of time, often repeatedly. For example, in an experiment on the recognition of gestures, the researcher typically has to recruit a large number of participants and have them execute the target set of gestures multiple times, over multiple sessions that are ideally spread across several days if the dynamics of user behavior variability over time are to be captured.

The challenge with designing these kinds of experiments is that they can be monotonous, making it very difficult to not only attract a large enough number of participants, but to also have them execute the required activity a large enough number of times as would be required for a statistically rigorous experiment. If deep learning-based methods are part of the suite of tools to be used to drive the analytics, the challenge even becomes more amplified since deep learning methods often require orders of magnitude more data than conventional machine learning approaches [29].

To enhance the data collection process through minimization of the monotony seen with traditional experimental protocols, gamification is one of the approaches that are sometimes used (e.g., see [7], [23], [20], [12]). The basic idea behind gamification is to build the data collection process in the form of a game in such a way that a carefully designed application collects relevant data behind the scenes. Since participants are able to enjoy the process of data collection because they are playing a game, it might be possible to recruit large numbers of them, and have them participate for long periods of time, enabling the collection of more data.

Using the case study of hand movement pattern analytics based on wrist-worn, sensor-enabled, devices, this paper examines the potential of gamification in wearable biometrics. Specifically, we address the following question: How closely does pattern execution during a gaming session match with pattern execution in the traditional setting where participants directly execute the patterns without any overarching gaming interaction? The patterns referred to here are those used for authentication on a smart phone screen (i.e., the pattern lock mechanism - see Figure 1 for illustration). If these two forms of pattern execution match closely, this would point to the potential of gamification of data collection for applications related to our work.

While we use authentication patterns as our case study, it is noteworthy that our work directly or indirectly applies to a broad range of applications which involve analytics of hand or wrist movement patterns as captured by a sensor-enabled wrist-worn device (example applications include, gesture recognition [33], air calligraphy [15], gait and handwriting authentication based on wrist movements [9], etc.).


*ICCDA 2019*, March 14–17, 2019, Kahului, HI, USA

ACM ISBN 978-1-4503-6634-2/19/03...$15.
https://doi.org/10.1145/3314545.3314572

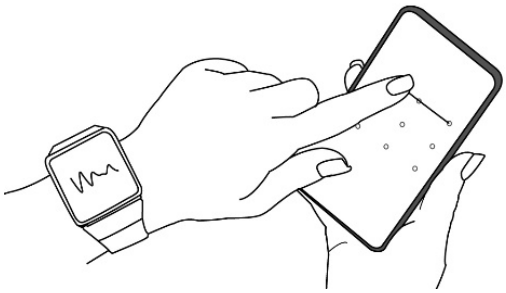


**Figure 1. Illustration of a user wearing a smartwatch entering her pattern.**

Although gamification could be used for benevolent data collection, we also explore these same techniques and experimental setup in the context of malicious intent. Could gamification help an attacker avoid suspicion - e.g., could a gaming app similar to one used for gaming data collection in our pattern matching experiment be used to learn a user's (secret) authentication pattern? To demonstrate these risks, we use the same experimental setup for both lines of research except for slight variations in the app design, underlying assumptions, and performance evaluation specifics (see Section 3.1 for additional details).

The contributions of this paper are summarized below:

1. **Studying gamification, as a means to ease data collection for wearable applications:** Taking the case of pattern execution at the authentication screen, we design an experiment that gamifies the data collection process and study the similarity between the patterns executed in these two cases. At the heart of this experiment is a gaming app that we built in such a way to mimic elements of the popular Flow Free app [36] that lend themselves to the pattern entry process. We find that depending on the specific scenario evaluated, gaming might hold promise as a means for collection of wearable sensor data in experiments involving wrist movement dynamics similarly to our experiment.
2. **Gamification as a tool for an adversary:** We take the case of a "bad guy" who posts a malicious gaming app on the app market (e.g., on Google Play) to evaluate how well such an app could perform at predicting a user's authentication pattern. The assumption in this case is that the app advertises a benign (gaming) functionality, but yet has underlying behavior that captures gaming patterns and compares them with the user's pattern at login time. Through the application of machine learning to the sensor data, we find that the adversary could reliably infer the user's authentication pattern using the game. To our knowledge we are the first to study the question of a gaming app which uses gaming patterns as training data that is later used to decode the user's pattern.

The rest of the paper is organized as follows: We discuss related work in Section 2 and the data collection experiments and machine learning design in Section 3. We then present our results and conclusions in Sections 4 and 5 respectively.

**Table 1. A selection of recent publications that studied wearable applications. UA stands for User Authentication, GR stands for Gesture Recognition, ST stands for Sleep Tracking and HAR stands for Human Activity Recognition. Observe that several of these papers use very small study populations, and by extension small datasets. Through gamification, such studies could be extended in scale while subjecting the participants and researchers to minimal load.**

| Publication | Application | # of Users |
|---|---|---|
| Lee et al. [13] | UA | 20 |
| Lewis et al. [14] | UA | 5 |
| Wen et al. [32] | GR | 10 |
| Zhang et al. [35] | GR | 5 |
| Chang et al. [5] | ST | 15 |
| Sun et al. [30] | ST | 16 |
| Khalifa et al. [11] | HAR | 10 |
| Roggen et al. [25] | HAR | 12 |

# 2. RELATED WORK

Our work is related to mainly two broad categories of previous research (1) gamification of cybersecurity training and experiment data collection, (2) mining user inputs using mobile and wearable sensor data. In this section, we discuss our related work in these two broad categories.

## 2.1 Gamification

Gamification involves use of game design elements in an activity to motivate users to participate and/or keep them engaged. For the research on using game design to motivate engagement during activities, the following cybersecurity training [6, 10, 27] and data collection [7] papers are among those distantly related to our own research. Games have also been built into commercial applications and used for encouraging users to share personal information, such as location. In De Nadai *et al.* [22] the authors used the Foursquare API to get information about user behavior across Italian cities. Without gamification, some of the data which facilitates research and commercial applications would be nearly impossible to gather.

Most closely related to our work in this category are two studies reported in [4, 7]. In order to motivate users to participate voluntarily in research studies and keep data collection tasks enjoyable, Dergous *et al.* [7] proposed gamifying data collection experiments and using in-game powerups in those gamified experiments. The authors designed an experiment involving Fitts reciprocal tapping tasks and conducted the experiment in three different ways i.e., using a gamified version deployed on Android Store, using a gamified version in the laboratory and using a non-gamified version in the laboratory. The data collected from the three versions was then compared and the results didn't show any significant difference in the quality of data collected. Similar to works in [7], Cechanowicz *et al.* [4] also designed three gamified versions of market research survey and conducted them along with the traditional version of the same surveys. The authors reported that there were no significant differences found in the response quality except in two situations where response data could have changed because of use of certain game elements. Although our work also used gamification, it significantly differs

from past research in the following ways: (1) Rather than using general methods of evaluating the quality of the data we obtain from these experiments, we built machine learning models to predict user behavior as a way of measuring and comparing the performance of different datasets. This allows us to compare in concrete terms the impact of differences in the data we collect. (2) Our work is centered on the use of sensor data rather than the results of games or easy to measure responses from participants. One concern might be that sensor data is potentially more unstable when compared between a game and a non-game version of a data collection experiment. (3) We explore the security risks associated with malicious actors who could use games to surreptitiously collect data on their users with the purpose of leveraging it directly or selling the information.

Next, we explore some of the previous security research which has used phone behavior to extract personal or secure information from users.

## 2.2 Mining User Inputs Using Sensor Data

Sensor data from mobile and wearable devices has been extensively used to learn patterns that breach privacy of users. Malicious applications installed on either smartphone or wearable device stealthy collect accelerometer and/or gyroscope data to make inference for user inputs such as keystrokes [3, 16, 19, 24], PINs [2, 26, 34], or pattern locks [2, 17, 18]. In this subsection, we discuss previous studies that use sensor data to mine user inputs categorized based on the source (or device) of the data used.

### *2.2.1 Mining user inputs using smartphone sensor data*

The first category of research reported in literature that mined user inputs from sensor data was based on data from smartphones. Cai *et al.* [3] used orientation sensor data collected from smartphone to learn patterns inferring keys on a number keypad. In this study, the authors collected sensor data while users pressed number keys, extracted features pertaining to the angular displacements made by keys when pressed and then inferred the keys on the number keypad with an accuracy of over 70%. Owusu *et al.* [24] advanced this area of research by inferring complete 6-character passwords rather than individual characters. Owusu *et al.* showed that accelerometer data collected from the smartphone could be used to infer the screen area pressed by user and 6-character passwords entered by the user. The authors interpolated the collected accelerometer data, extracted 46 statistical features and then used a Random Forest classification algorithm to make inference of the pressed screen-area and entered passwords. The authors obtained a prediction accuracy of 24.5% with 60 screen partitions and predicted the correct passwords within a median of 4.5 trials.

Xu *et al.* [34] provided a more practical implementation of how sensor data could be used to learn motion change patterns of tap events for inference of keys on a number pad and PINs. The authors presented how the key press detection and inference would be done in both the training and real-world scenarios. The authors used k-means to make key press inference. They obtained an average accuracy of over 62% for each key prediction in the first 4 attempts and over 80% for PINs in the first 3 attempts. Other past research in this category include works in [2, 21] which analyzed more complex privacy leakage scenarios like inferring pattern locks, letter taps.

An obvious difference between our work and the works presented in this category is that our work infers user sensitive information (pattern locks) entered on the smartphone using sensor data from the smartwatch paired with that phone and not directly using sensor data from that smartphone on which the user types.

### *2.2.2 Mining user inputs using smartwatch sensor data*

With the proliferation of smartwatches, new researches showing privacy leakages using sensor data from the smartwatch coped up such as [17, 18, 26, 31]. Similar to works in the previous category, past research in [16, 17, 19, 31] used accelerometer and gyroscope data to predict keys on a number keypad, PINs and words.

Most recent and closely related to our paper in this category are works in [18]. Lu *et al.* [18] used several traditional machine learning techniques and deep learning techniques to infer PINs and Android Pattern Locks (APL) drawn on the smartwatch. The authors achieved an accuracy of over 95% and 98% for PINs and APLs respectively using the traditional machine learning techniques and APL prediction accuracy of 39% for the first guess using the deep learning technique. A notable difference between this Lu *et al.*'s work and our work is the pattern locks being inferred are drawn on the smartwatch itself while the pattern locks being inferred in our work are drawn on the smartphone.

In this category, our work is more closely related to last subcategory of research described above in that we mine user inputs based on sensor data collected from the smartwatch. However, our work is different from all works described in this category in the following ways: (1) we designed a malicious gaming app derived from a popular game (Flow Free [36]) that uses user behavioral data to learn hidden sensor patterns corresponding to predetermined patterns of the user. Previous studies collected training data from custom-designed applications similar to the pattern locks or PIN interfaces and not a completely different application with embedded patterns implicitly. (2) we incorporate several training scenarios (i.e., training using one user game data and using all user game data) and make comparison between these scenarios. Previous studies only use one scenario with both training and testing data obtained from the same application. (3) we use sensor data from smartwatch to infer user authentication pattern lock used on the smartphone.

# 3. DATA COLLECTION AND MACHINE LEARNING EXPERIMENTS

In this section we describe our data collection experiments and how we implemented our machine learning mechanism. Note that for our research, both the benign scenario (i.e., gamification of data collection) and the attack scenario (i.e., using gamification to stealthily learn a user's pattern) are based on the same data collection experiment since the experiment is adequate to collect all the required information. In a real world setting however, these two scenarios would have some variations in assumptions of what and how certain data is accessed, and by extension the design of the application. We first describe these assumptions before describing our data collection experiments.

## 3.1 Assumptions: Threat Scenario vs Benign Scenario

**Threat Scenario - Using gamification for attack:** The idea behind this scenario is that a malicious entity puts a game on the app market such that users download and play it. Unbeknownst to the users, the game is instrumented to elicit user moves or actions that enable the capture of security-sensitive data such as behavioral biometric data (e.g., swipe patterns that could later be used to drive attacks such as that in [28]) or even more traditional authentication credentials (e.g., an unlock pattern). For a more

focused exposition, we tailor the rest of the description to the pattern lock mechanism since this is the case studied in our experiments. A gaming app built to "steal" an authentication pattern used for a smart phone pattern lock might be designed in such a way to have the user execute patterns or pattern-like shapes during regular game play. We have built an example of this kind of app for this research (see Section 3.3.2). As the user makes moves during gaming operations, the game is methodically getting him to execute certain patterns to build a training set tailored to the user (in our case the relevant training data is motion and orientation sensor data collected from a smart watch). Meanwhile, this gaming app also has a background service component that runs at all times in the background and logs all those times when the user executes a pattern to log in (Mobile OS'es provide broadcasts that apps can use to determine when the user just logged in - e.g., see [37] for Android).

The combination of data collected by both the regular gaming app and the background service means that the attacker has access to: (1) training data of the user executing a wide range of patterns, and, (2) actual (unlabeled) data of the user's authentication pattern that could be labelled using data from (1) if the pattern exists in the training set. The prediction performance obtained from this case of a malicious gaming app will be captured in Scenario II during our performance evaluation (see Table 3 for summary of our scenarios).

**Benign Scenario - Using gamification for data collection:** The main requirement in this case is that participants interact with a gaming app that prompts them to execute the activity required for the experiments. Depending on the experiment design objectives, participants may, or may not be told in advance what "actual experiment" the game represents. Again, tailoring our description to the pattern lock, the gaming process would involve people executing patterns selected by the experimenter, and the collected data labelled accordingly to correspond to the relevant patterns. In this paper, we measure how well these gaming patterns match with the real patterns by also having the users execute the real patterns and making a comparison (see Scenario III in Table 3).

## 3.2 Data Collection Experiments

After getting approval from our university's Institutional Review Board (IRB), we conducted two kinds of data collection experiments: (1) Experiments in which participants interacted with a gaming app (i.e., played a game) on a smart phone, and, (2) Experiments in which participants entered patterns into the Android pattern lock screen (i.e., experiments which simulated the user authentication process on a pattern lock screen). Going forward we refer to these experiments as G (for gaming) and P (for pattern). For both experiments, participants wore a smart watch whose sensors recorded motion and orientation dynamics (i.e., linear acceleration and gyroscope readings) during the process of each user's interaction with the phone. These sensor measurements enabled us to capture the hand movement patterns while users interacted with the phones.

For each of the experiments G and P, each user participated in two sessions that were at least one day apart. Collection of each user's data on two separate days allowed us to capture some of the user behavior variability during the gaming and (or) pattern entry process. A total of 24 users participated in our experiments of whom 15 were male. All participants were students, faculty or staff at our university. The smart watch used in our experiments was the LG Urbane [38], while the smart phone used was a Samsung Galaxy S8+ [39]. Data from the watch was transmitted to the phone via a blue tooth connection.

## 3.3 Design of Applications used for Data Collection

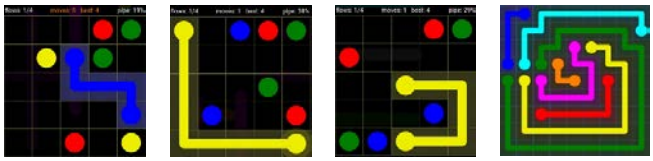
### *3.3.1 App for Experiment P*

Since Experiment P is the standard pattern entry process on an Android phone, we built for it an application that emulates the Android pattern lock for GUI of our app. The only variation from the standard Android pattern lock was that our application recorded a time stamp at each touch point, which enabled us to delimit each pattern. Note that in practice the attacker could delimit a pattern in many different ways (e.g., through broadcasts sent by the OS during login [8], or through fingerprinting the behavioral quirks of pattern entry [1]).

### *3.3.2 App for Experiment G*

Recall that the idea behind experiment G is to have participants play a game whose operations mimic the entry of different kinds of patterns. As users play moves on the app (e.g., connecting objects, tracing paths, etc.), they are executing different kinds of patterns similar to those that people typically use for authentication.

After exploring a wide range of gaming apps on the Google play app market, we identified the Flow Free puzzle game (Figure (2)) as a good model for this kind of application. In the Flow Free puzzle game, a user connects pairs of similar colored dots using links that must not intersect. Since these dots are located on a grid shaped similarly to the Android pattern lock grid, the process of connecting these dots is like the pattern entry process. Figure 2 shows example screenshots of the Flow Free game during typical game play.

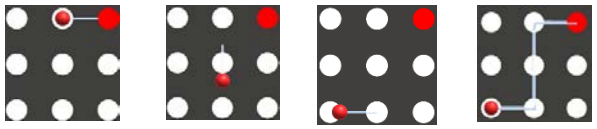


(a) (b) (c) (d)

**Figure 2. Screenshots of Flow Free puzzle game levels. Figure 2a and 2b and 2c show first pipes drawn in different levels. Figure 2d shows the screenshot of when a level is completed.**

Our game follows the same general idea as Flow Free, except that it: (1) permits the intersection of paths, (2) involves a finger trailing, or tracing out the path of, a moving ball (as opposed to joining dots), and, (3) only shows the most recent segment of the path (as opposed to the full path) traversed by the finger.

The idea behind (1) is to provide support for all sorts of arbitrary patterns while the idea behind (2) is to force the user to execute a path determined by the app since the movement of the ball determines the user's finger trajectory. The combination of (2) and (3) forces the user to execute at a speed determined by the gaming app (i.e., if the ball moves fast, the user will move fast in order to traverse the path while it is still visible, otherwise the user might get the path wrong and get a low score). Figure 3 shows an example execution of the game. As the ball moves from one point to the next (Figure 3a is the first step while Figure 3d is the last step), the finger follows it and traverses its full path. Figure 3d shows the pattern resulting from the combination of operations shown in Figures 3a, 3b and 3c. Note that in practice Figure 3d is not shown to the user. We only show it here for illustration purposes. During gameplay our app had users execute the patterns shown in Figure 4, which are adopted from [40]. The majority of these are frequently used for authentication.



(a) First (b) Second (c) Third (d) Fourth

**Figure 3. Screenshots of our game in action from the first step to fourth. Our game level finishes at the fourth step (Figure 3d). Users do not see the whole pattern in the end as shown in Figure 3d and is only presented for illustration purpose.**

## 3.4 Building Machine Learning Models

### *3.4.1 Preprocessing and Feature Extraction*

In this section, we describe our data preprocessing and feature extraction process before we apply the data to various classification models. The linear acceleration gives the acceleration of the device along the three axes (x, y and z) and gyroscope measures the rate of rotation around the device's axes. We use all three dimensions of these two sensors in our study. The first basic step of data preprocessing is removing the noise and outliers from sensor readings. We undertake this by applying a median filter algorithm (from *Scikit-Learn*) to all the axes.

Having filtered the data, we then build 4 data vectors for each sensor and each axis, creating a total of 24 (=4 *vectors* x 2 *sensors* x 3 *axes*) different data vectors that we later use to compute the features. Here, we first describe these 4 data vectors before proceeding to describe the feature computation.

[1] **Vector #1 - Raw Time Series Data:** Let the notation $S_1$ denote the sensor readings of linear acceleration. Then a sequence of readings of $S_1$ can be denoted by $S_1 = a_1, a_2, ....a_n$ of size n. This sequence represents the first data vector.

[2] **Vector #2 - Derivative of Raw Time Series Data:** From $S_1$, we calculate its derivative and get $S_1^d = [a_1^d, a_2^d, a_3^d, ....a_n^d]$, the second data vector.

[3] **Vector #3 - Fast Fourier Transform (FFT):** Calculate the FFT of $S_1$ and get $S_1^f = [a_1^f, a_2^f, a_3^f, ....a_n^f]$, the third data vector.

[4] **Vector #4 -Derivative of FFT:** Perform a derivative on data vector $S_1^f$ and obtain $S_1^{df} = [a_1^{df}, a_2^{df}, a_3^{df}, ....a_n^{df}]$, the fourth data vector.

After computing the above 4 vectors on each axis of each sensor, we then compute the features according to the feature types described below and presented in Table 2.

**Feature Type Description:** The feature set used in our study are classified into Type A, B, C and D depending on the nature of vectors (time and frequency domain) they operate on and the axis (x, y, z) over which they are computed.

The vectors 1 (raw time series data) and 2 (derivative of raw time series data) are in time domain (i.e. values in the vector are known with respect to time) whereas vectors 3 (FFT of raw time series data) and 4 (derivative of FFT) are in frequency domain (i.e. values are known with respect to frequency). To present how these feature types differ from one another, we discuss the differences in their computation.

The feature types – Type A, C and D are applied to both time domain (1, 2) and frequency domain vectors (3, 4), however Type A features are computed from each axis separately. These feature types result in different lengths of feature. Type A feature results

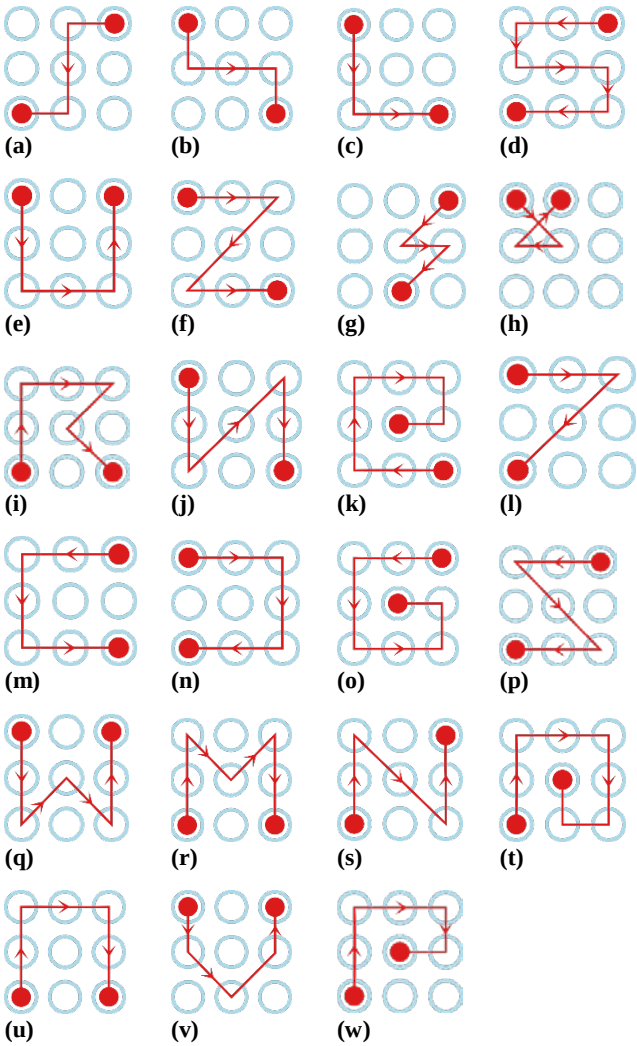



**Figure 4. The 23 patterns used in our experiments. For Experiment G, the user simply follows a ball around the screen but ends up executing each of these patterns multiple times in the process. The game only shows part of the path to the user in order to minimize the likelihood that the user keeps track of any patterns being executed during the game. In Experiment P, users are shown the patterns and hence execute them in just the same way they would execute them while logging into a phone. Data collected in P is used to measure how well the patterns collected in G match with real patterns.**

in 8 features per vector on each axis for each sensor, and Type C and D feature types result in 3 and 1 features from three axes respectively. Feature Type C (inter-axis) uses pairings of axis (*xy*, *yz* and *yz*) to compute 3 features. Feature Type D uses all the axis (x, y, and z) to compute 1 feature. The feature `*correlation*' (Type C (inter-axis)) is calculated between axes *x*, *y*; *y*, *z* and *x*, *z* giving three features namely- $corr_{xy}$, $corr_{yz}$ and $corr_{xz}$ respectively and feature "*signal magnitude area*" (Type D) is computed as the sum of absolute value of x, y and z giving a single feature.

For the last feature Type B, we compare it with feature Type A. Feature Type B is similar to Type A, in that its features are also computed separately for each axis but differs to Type A in that it is only applied on frequency domain vectors (3, 4). The application of Type B features result in 4 features per vector per

**Table 2. The feature types, Type A and Type B are extracted for each dimension of a sensor reading. The features in Type C are extracted from inter axis (xy, yz, xz) and Type D is extracted from all axes of a sensor reading.**

| Feature Type | Features | Data Vectors on which Features are computed | # of Features |
|---|---|---|---|
| **Type A (time-freq domain)** | mean, standard deviation, mean, absolute deviation, minimum, maximum, energy, interquartile range, entropy | Vectors 1, 2, 3, 4 | 8 x 4 x 3 = 96 |
| **Type B (frequency-domain)** | spectral maximum index, spectral mean frequency, spectral skewness, spectral kurtosis | Vectors 3, 4 | 4 x 2 x 3 = 24 |
| **Type C (inter-axis: —xy, yz and xz)** | correlation | Vectors 1, 2, 3, 4 | 3 x 4 = 12 |
| **Type D (all-axis: — x, y and z)** | signal magnitude area | Vectors 1, 2, 3, 4 | 4 |

per axis and per sensor.

**Feature Extraction:** In our study, we extracted 136 features from each sensor. Since the feature extraction process is the same for both the sensors, we only show the process for any one sensor. Recall that, each sensor has vectors 1 through 4 for each axis. For vectors 1 and 2, we compute 8 features from Type A for each of six axes of the vector (8 x 6). Then, using inter axis of each vector, we compute 3 features from Type C (3 x 2) and 1 feature from Type D (1 x 2) from all axes. In total, we get 8 features from Type C and Type D. These 56 features from vectors 1 and 2 are appended to a feature vector, *F* (*no. of features = 56*).

From vectors 3 and 4, for all six axis of these vectors, 8 features from Type A and 4 features from Type B are computed ((8+4) x 6); Similar to above calculation for Type C and Type D, we get 8 additional features. The resulting 80 features are appended to *F* (*no. of features 56+80 = 136*).

The whole process of feature extraction from both sensors gives us 272 features in total. Our preliminary experiments showed that all these features contributed to good classification accuracy, so we did not perform any feature selection.

### *3.4.2 Training and Testing Details*

**Datasize of Experiments G and P:** If we recall our experiment design, we have two experiments G and P where a user either plays or enters pattern and, in both experiments we use the same 24 users. In experiment G, in the guise of a game, a user enters 23 patterns (Figure 4) for ten repetitions, same as in experiment P. From each of these experiments, we get 230 instances of data for a user. The total data size in experiment G can be calculated as *no. of users* x *no. of patterns* x *no. of repetitions* and in a similar manner for experiment P. Given we have 24 users, 23 patterns and the number of repetitions is 10, the datasize of experiments G and P is 5520 each.

**Size of Feature Matrix:** From the above feature set calculation, the number of features computed for each pattern is 272. For both experiments G and P, each with 5520 instances of data, we have a feature matrix of size (5520, 272) with no. of rows = 5520 and no. of columns = 272.

**Scenarios:** Here, we model two potential use cases for our study and create three scenarios (I, II, III) in Table 3. The first use case is user-specific such as gesture recognition, classification of medical problems. The second use case is generic such as when an attacker is trying to infer pattern lock of a user using data corpus of other users. Scenario I-III model these use cases in the way they choose the training and testing data. The application of scenarios I and III falls under generic use case and, II falls under user-specific use case.

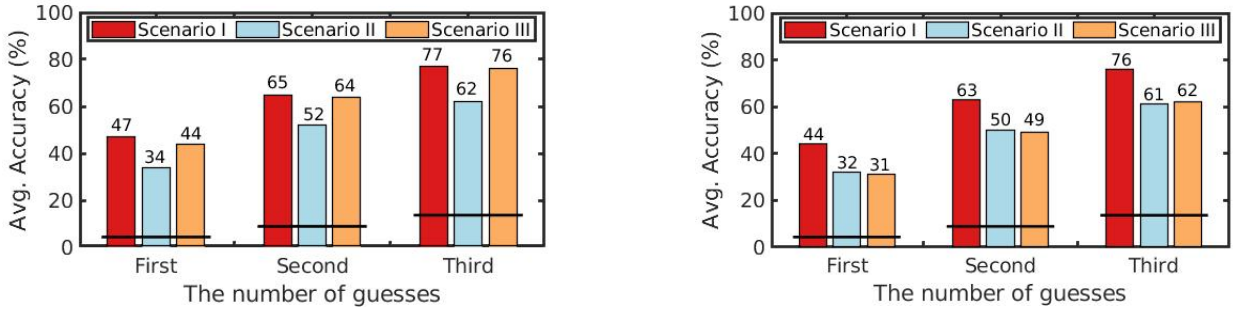
Let's denote the data instances from experiment G to be $d_1$ and data from experiment P to be $d_2$. Depending on the use cases for scenarios I-III, we pick either $d_1$ or $d_2$ as training or testing data, the detail of which is explained below.

1. **Scenario I:** Scenario I uses data from $d_2$ from all users except one user for training the classification model, and tests against that user's $d_2$ data. The size of the training data is computed as (*no. of users - 1* x *no. of patterns* x *no. of repetitions* = 5290) and the testing data size is 230. This scenario is the baseline against which we compare the other two scenarios described below.
2. **Scenario II:** Scenario II combines all users' data $d_1$ except one user's data $d_1$ for training a classification model and tests against the same user's data $d_2$. Like I, the training data size is 5290 and the testing data size is 230. This scenario can be used by an attacker to predict a user's pattern lock from data corpus of other users.
3. **Scenario III:** Scenario III tries predict a user's pattern lock from $d_2$, given we train the model using that user's $d_1$. So, $d_1$ from a user is used for training the classification model, the same user's $d_2$ is used for testing. As discussed above, each pattern is entered 10 times in both experiments G and P, so in each user's experiment, both the training and testing data size is 230. This scenario can be a model for benign data collection games.

To summarize, scenarios I and II do a leave-one-out classification, whereas scenario III does a user-to-user classification using same user's data for training and testing. In leave-one-out, the model is built from training data from all users except one and is evaluated against testing data from the left-out user. The results of classification of these scenarios are averaged for all 24 users. Thus, these three scenarios are ways in which a classification model is built from various pairings of training and testing data of users for potential use cases (Table 3).

**Table 3. Experimental scenarios I, II, and III used to model the study for potential use cases.**

| Scenarios | Description | Relevance to our Study |
|---|---|---|
| **I** | Train a model using other user's data from P and test using the user's data from P. | Baseline experiment where pattern data is used for both training and testing. It provides a reference for how well scenarios II and III perform. |
| **II** | Train a model using a user's data from G, and test against data of the same user from P. | Represents the malicious gaming app scenario. |
| **III** | Train a model using other user's data from G and test using the user's data from P. | Represents the gamification scenario used for regular experimentation. |




**(a) Logistic Regression (LR) classifier** **(b) Support Vector Machine (SVM) classifier**

**Figure 5. The prediction accuracies for Scenarios I, II and III are averaged over guesses of first, second and third for all users. The plots are results for classifiers LR (5a) and SVM (5b) respectively. The horizontal black line drawn over the bar graphs for scenarios I, II and III are prediction accuracies of random guesses (first, second and third).**

**Classification Algorithms:** For our analysis, we use classifiers Support Vector Machine (SVM) and Logistic Regression (LR). The SVM model is set with a linear kernel and a penalty parameter (*C=2*), and the LR model is set with a "*libfgs*" solver, a multinomial loss function and a penalty parameter (*C=2*) using the *Scikit-Learn* library. In addition, we set both the classifiers to give probability estimates.

# 4. PERFORMANCE EVALUATION

In this section, we report the results from three experimental scenarios in Table 3 to evaluate our ability to infer patterns using the sensor data (linear acceleration and gyroscope) collected from a smartwatch. We then explore the user and pattern level results which contribute to the overall success of the classifiers in these experiments. In the following section we provide a high-level look at the performance of each experiment.

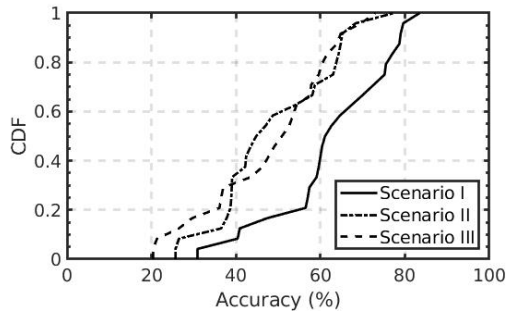
## 4.1 Pattern Lock Inference Accuracy

A smartphone allows a user to unlock their android lock pattern even after three unsuccessful attempts but for our study we report pattern inference results for up to three guesses only. Our study uses a small dataset of 23 patterns. So, it does not make sense to use more than three guesses.

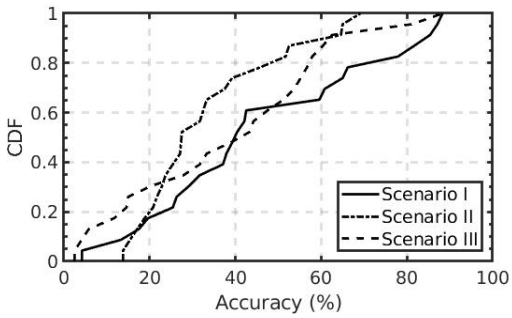
**Overview of Scoring and Figures:** The objective of the three scenarios in Table 3 is to find how accurately the patterns in Figure 4 can be predicted from the sensor data. The results of pattern lock inference are shown in Figure 5 for classifier LR (Figure 5a) and classifier SVM (Figure 5b). In the bar plot, each of the scenarios are put together side-by-side and their average accuracy (y-axis) over number of guesses (x-axis) is shown. For each set of test data (as described in Table 3), the classifiers output the top three labels in order of probability. If the test data is correctly predicted in the first guess, we count this data in the accuracy calculation for first guess. Similarly, we get the accuracy for the second and third guess. Figure 5 reports the accuracy for each of the guesses and classifiers. Note the black line that lays along each set of plots. In all the scenarios, the prediction accuracies of the patterns are over 26%, 40% and 47% better than random guess (black line) for the first, second and third attempts respectively.

**Baseline Experiment:** As we review the performance of the experiments, Scenario I should be thought of as the baseline experiment. If an attacker trying to predict unlock patterns was determined they could gather a large amount of data from their associates or people they pay, to swipe different phone unlock patterns. In both Figures (5a and 5b), Scenario I performs 1-15% better than the other two scenarios. It makes sense that although the game has been designed to be similar, only using the P dataset will provide slightly better results in most cases.

**Comparison of Performance:** However, it is not always true that there is a significant difference between Scenario I and Scenario III. Looking at Figure (5a) shows that Logistic Regression (LR) for Scenario III performs almost as well as Scenario I. Additionally, although it might make sense for Scenario III to perform better than Scenario II if there was enough data to effectively generalize across individual users, it isn't always true. Although LR fits this pattern, the SVM might be overfitting on the




**(a) User-Level Accuracy of SVM classifier**




**(b) Pattern-Level Accuracy of LR classifier**

**Figure 6. The CDF plots are user-level and pattern-level performance of our classification models SVM(6a) and LR(6b). The plot results are User-Level (6a) and Pattern-Level (6b) accuracies averaged over third guess for scenarios in Table 3.**

data from other users. If this is true, it would explain why LR is able to outperform Scenario II with III, even though this does not occur for the SVM.

**Implications of Experimental Performance:** Scenarios II and III were designed to evaluate how a game designed to collect data for a specific experiment would perform for the real task. Although both scenarios demonstrated that the approach is feasible, it seems likely that the more successful case is when the researcher or attacker collects a large amount of data from an app that can infer how you might unlock your phone. After three guesses, the LR classifier is able to predict the unlock pattern 76% of the time, while when using only the target's data from playing the game, the success rate is only 62%.

Next, we explore how the user level performance of the experimental approach impacts the overall performance.

## 4.2 User-Level Performance

To examine the impact that each user might have had on the overall performance of the algorithm in predicting phone unlock patterns, we plotted a Cumulative Density Function (CDF) for the accuracy of the algorithm on specific users with its third guess. Looking at Figure 6a, we again use Scenario I as the baseline. By looking at the location of the curves in the plot, it is clear that our baseline scenario performs far better than the other two experiments for almost all users except a small subset of them. While the model had an accuracy of at least 60% for nearly 80% of users for Scenario I, 80% of users only had an accuracy of over 40% for the other two experiments. This means that it isn't likely to be just a small subset of users for Scenarios II and III which are dragging down the results. In fact, this is more likely the case for Scenario I.

## 4.3 Pattern-Level Performance

The pattern-level analysis of the results can tell us a lot about the performance of our models and scenarios (Table 3). Figure 6b reports the pattern-level accuracy of individual scenarios for LR classifier. Scenario I performs almost as well as the other two scenarios (II, III) which affirms feasibility of the use cases shown in Table 3. The results demonstrate that the gamification of benign applications such as data collection and of malicious intents such as pattern lock inference, are plausible. Looking at the confusion matrix of LR classifier (Figure 7) for Scenario III, we observe most of the patterns get classified correctly in three attempts. For Scenario III, the average performance of pattern inference of the LR classifier is around 76%. Despite we observe

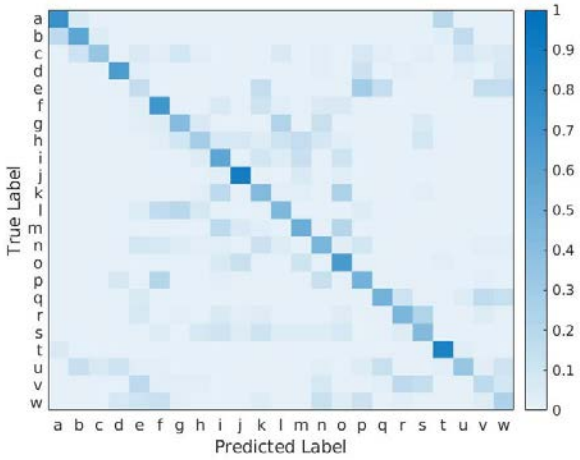



**Figure 7. Confusion matrix of an LR classifier for Scenario III for 23 patterns (Figure 4). The confusion matrix gives an idea of which pattern(s) tends to be misclassified as another pattern(s). Looking at the non-diagonal elements, the figure reveals very few cells with dark blue color, which implies very few misclassifications overall. Note that this confusion matrix corresponds to the case of 3 guesses, where overall accuracies are in the 70's.**

some prominent misclassification of patterns in the non-diagonal areas of the confusion matrix. Patterns which are very similar to one another seem to be harder to distinguish. It is possible to find pairs that are likely to be confused with one another. For example, patterns (r) and (s) have largely the same individual swiping actions, except one of them switches swiping direction near the end. Another example to look at is pattern (f), (f) is a Z-like pattern from the top left to the bottom right and several other patterns get frequently classified as (f) (e.g., (l), (n), (p), and (w)). Several of these falsely classified patterns share parts of their pattern with (f).

With additional data and more sophisticated techniques that require large amounts of data (such as deep learning), we might be able to better distinguish between the relatively small number of patterns that are being mistaken for one another. However, in most cases the classifier performs well and is able to clearly distinguish between patterns that are quite similar. For example, although (r) is very similar to (u), it is rarely mistaken for the similar pattern. These pattern level results explain some of the

source of error for our experiments and they motivate further research.

## 5. CONCLUSION

In this paper, we have explored the idea of gamification of experiments involving wearables data collection. Concurrently with the gamification, we have also studied the case of a malicious entity who leverages the appeal of gaming to bait users into having their information (in our case authentication patterns) stolen. Our results have shown that gamification is able to illicit data that closely matches with data collected in the conventional experimental setup(s). Further, we have shown that the attack exploiting the gamification idea is also able to generate patterns that very closely match user's real patterns. In our future work, we will continue to explore the notion of gamification in other data collection settings and identify and examine potential attacks related to those settings.

## 6. ACKNOWLEDGMENT

This research was supported by National Science Foundation Award Number: 1527795.

## 7. REFERENCES

[1] Angulo, J. and Wästlund, E. 2011. Exploring touch-screen biometrics for user identification on smart phones. *IFIP PrimeLife International Summer School on Privacy and Identity Management for Life* (2011), 130–143.

[2] Aviv, A.J., Sapp, B., Blaze, M. and Smith, J.M. 2012. Practicality of accelerometer side channels on smartphones. *Proceedings of the 28th Annual Computer Security Applications Conference* (2012), 41–50.

[3] Cai, L. and Chen, H. 2011. TouchLogger: Inferring Keystrokes on Touch Screen from Smartphone Motion. *HotSec*. 11, (2011), 9.

[4] Cechanowicz, J., Gutwin, C., Brownell, B. and Goodfellow, L. 2013. Effects of gamification on participation and data quality in a real-world market research domain. *Proceedings of the first international conference on gameful design, research, and applications* (2013), 58–65.

[5] Chang, L., Lu, J., Wang, J., Chen, X., Fang, D., Tang, Z., Nurmi, P. and Wang, Z. 2018. SleepGuard: capturing rich sleep information using smartwatch sensing data. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*. 2, 3 (2018), 98.

[6] Dabrowski, A., Kammerstetter, M., Thamm, E., Weippl, E. and Kastner, W. 2015. Leveraging competitive gamification for sustainable fun and profit in security education. *2015 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 15)*. (2015).

[7] Dergousoff, K. and Mandryk, R.L. 2015. Mobile gamification for crowdsourcing data collection: Leveraging the freemium model. *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (2015), 1065–1074.

[8] Diao, W., Liu, X., Li, Z. and Zhang, K. 2016. No pardon for the interruption: New inference attacks on android through interrupt timing analysis. *Security and Privacy (SP), 2016 IEEE Symposium on* (2016), 414–432.

[9] Griswold-Steiner, I., Matovu, R. and Serwadda, A. 2017. Handwriting watcher: A mechanism for smartwatch-driven handwriting authentication. *Biometrics (IJCB), 2017 IEEE International Joint Conference on* (2017), 216–224.

[10] Jin, G., Tu, M., Kim, T.-H., Heffron, J. and White, J. 2018. Game based Cybersecurity Training for High School Students. *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (2018), 68–73.

[11] Khalifa, S., Lan, G., Hassan, M., Seneviratne, A. and Das, S.K. 2018. Harke: Human activity recognition from kinetic energy harvesting data in wearable devices. *IEEE Transactions on Mobile Computing*. 17, 6 (2018), 1353–1368.

[12] Laso Bayas, J.C., See, L., Fritz, S., Sturn, T., Perger, C., Dürauer, M., Karner, M., Moorthy, I., Schepaschenko, D., Domian, D. and others 2016. Crowdsourcing in-situ data on land cover and land use using gamification and mobile technology. *Remote Sensing*. 8, 11 (2016), 905.

[13] Lee, W.-H., Liu, X., Shen, Y., Jin, H. and Lee, R.B. 2017. Secure pick up: Implicit authentication when you start using the smartphone. *Proceedings of the 22nd ACM on Symposium on Access Control Models and Technologies* (2017), 67–78.

[14] Lewis, A., Li, Y. and Xie, M. 2016. Real time motion-based authentication for smartwatch. *2016 IEEE Conference on Communications and Network Security (CNS)* (2016), 380–381.

[15] Li, Q., Cao, H., Lu, Y., Yan, H. and Li, T. 2016. Controlling Non-Touch Screens as Touch Screens Using Airpen, a Writing Tool with In-Air Gesturing Mode. *System and Software Reliability (ISSSR), International Symposium on* (2016), 68–76.

[16] Liu, X., Zhou, Z., Diao, W., Li, Z. and Zhang, K. 2015. When good becomes evil: Keystroke inference with smartwatch. *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security* (2015), 1273–1285.

[17] Liu, Y. and Li, Z. 2018. aleak: Privacy leakage through context-free wearable side-channel. *IEEE INFOCOM 2018-IEEE Conference on Computer Communications* (2018), 1232–1240.

[18] Lu, C.X., Du, B., Wen, H., Wang, S., Markham, A., Martinovic, I., Shen, Y. and Trigoni, N. 2018. Snoopy: Sniffing your smartwatch passwords via deep sequence learning. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*. 1, 4 (2018), 152.

[19] Maiti, A., Jadliwala, M., He, J. and Bilogrevic, I. 2015. (Smart) watch your taps: side-channel keystroke inference attacks using smartwatches. *Proceedings of the 2015 ACM International Symposium on Wearable Computers* (2015), 27–30.

[20] McKenzie, G. 2011. Gamification and location-based services. *Workshop on Cognitive Engineering for Mobile GIS* (2011).

[21] Miluzzo, E., Varshavsky, A., Balakrishnan, S. and Choudhury, R.R. 2012. Tapprints: your finger taps have fingerprints. *Proceedings of the 10th international conference on Mobile systems, applications, and services* (2012), 323–336.

[22] De Nadai, M., Staiano, J., Larcher, R., Sebe, N., Quercia, D. and Lepri, B. 2016. The death and life of great Italian cities:

a mobile phone data perspective. *Proceedings of the 25th international conference on world wide web* (2016), 413–423.

[23] Odobašić, D., Medak, D. and Miler, M. 2013. Gamification of geographic data collection. (2013).

[24] Owusu, E., Han, J., Das, S., Perrig, A. and Zhang, J. 2012. ACCessory: password inference using accelerometers on smartphones. *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications* (2012), 9.

[25] Roggen, D., Calatroni, A., Rossi, M., Holleczek, T., Förster, K., Tröster, G., Lukowicz, P., Bannach, D., Pirkl, G., Ferscha, A. and others 2010. Collecting complex activity datasets in highly rich networked sensor environments. *Networked Sensing Systems (INSS), 2010 Seventh International Conference on* (2010), 233–240.

[26] Sarkisyan, A., Debbiny, R. and Nahapetian, A. 2015. WristSnoop: Smartphone PINs prediction using smartwatch motion sensors. *Information Forensics and Security (WIFS), 2015 IEEE International Workshop on* (2015), 1–6.

[27] Schreuders, Z.C. and Butterfield, E.M. 2016. Gamification for teaching and learning computer security in higher education. *2016 USENIX Workshop on Advances in Security Education (ASE 16)* (2016).

[28] Serwadda, A., Phoha, V. V, Wang, Z., Kumar, R. and Shukla, D. 2016. Toward robotic robbery on the touch screen. *ACM Transactions on Information and System Security (*TISSEC*)*. 18, 4 (2016), 14.

[29] Sun, C., Shrivastava, A., Singh, S. and Gupta, A. 2017. Revisiting unreasonable effectiveness of data in deep learning era. *Computer Vision (ICCV), 2017 IEEE International Conference on* (2017), 843–852.

[30] Sun, X., Qiu, L., Wu, Y., Tang, Y. and Cao, G. 2017. Sleepmonitor: Monitoring respiratory rate and body position during sleep using smartwatch. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*. 1, 3 (2017), 104.

[31] Wang, H., Lai, T.T.-T. and Roy Choudhury, R. 2015. Mole: Motion leaks through smartwatch sensors. *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking* (2015), 155–166.

[32] Wen, H., Ramos Rojas, J. and Dey, A.K. 2016. Serendipity: Finger gesture recognition using an off-the-shelf smartwatch. *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (2016), 3847–3851.

[33] Xu, C., Pathak, P.H. and Mohapatra, P. 2015. Finger-writing with smartwatch: A case for finger and hand gesture recognition using smartwatch. *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications* (2015), 9–14.

[34] Xu, Z., Bai, K. and Zhu, S. 2012. Taplogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors. *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks* (2012), 113–124.

[35] Zhang, Y. and Harrison, C. 2015. Tomo: Wearable, low-cost electrical impedance tomography for hand gesture recognition. *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology* (2015), 167–173.

[36] Google Play. https://play.google.com/store/apps/details?id=com.bigduckgames.flow

[37] Intent | Android Developers. https://developer.android.com/reference/android/content/Intent#ACTION_USER_UNLOCKED

[38] LG W150: Watch Urbane - Sleek, Stylish Smartwatch | LG USA. https://www.lg.com/us/smart-watches/lg-W150-lg-watch-urbane

[39] Samsung Galaxy S8 and S8+ - The Official Samsung Galaxy Site. https://www.samsung.com/global/galaxy/galaxy-s8/

[40] Trickytricks. http://mytrickytricks.blogspot.com/2013/07/commonlockpattern.html